The impact of data-heavy, post-quantum TLS 1.3 on the Time-To-Last-Byte of real-world connections

Panos Kampanakis
Amazon Web Services
kpanos@amazon.com

Will Childs-Klein
Amazon Web Services
childw@amazon.com

Abstract*—It has been shown that post-quantum key exchange and authentication with ML-KEM and ML-DSA, NIST's post-quantum algorithm picks, will have an impact on TLS 1.3 performance used in the Web or other applications. Studies so far have focused on the overhead of quantum-resistant algorithms on TLS time-to-first-byte (handshake time). Although these works have been important in quantifying the slowdown in connection establishment, they do not capture the full picture regarding real-world TLS 1.3 connections which carry sizable amounts of data. Intuitively, the introduction of an extra 10KB of ML-KEM and ML-DSA exchanges in the connection negotiation will inflate the connection establishment time proportionally more than it will increase the total connection time of a Web connection carrying 200KB of data. In this work, we quantify the impact of ML-KEM and ML-DSA on typical TLS 1.3 connections which transfer a few hundreds of KB from the server to the client. We study the slowdown in the time-to-last-byte of post-quantum connections under normal network conditions and in more unstable environments with high packet delay variability and loss probabilities. We show that the impact of ML-KEM and ML-DSA on the TLS 1.3 time-to-last-byte under stable network conditions is lower than the impact on the time-to-first-byte and diminishes as the transferred data increases. The time-to-last-byte increase stays below 5% for high-bandwidth, stable networks. It goes from 32% increase of the time-to-first-byte to under 15% increase of the time-to-last-byte when transferring 50KiB of data or more under low-bandwidth, stable network conditions. Even when congestion control affects connection establishment, the additional slowdown drops below 10% as the connection data increases to 200KiB. We also show that connections in lossy or volatile networks could see higher impact from post-quantum handshakes, but these connections' time-to-last-byte degradation still drops as the transferred data increases. Finally, we show that such connections are already significantly slow and volatile regardless of the TLS handshake.

I. Introduction

It is imperative for today's digital communications to be properly secured and for data to be protected. One widely used protocol which establishes such secure communication channels is TLS. TLS 1.3 [32], the latest TLS version, is used to negotiate and establish secure channels which encrypt and authenticate all exchanged data between a client and server, authenticate peer identities, and protect communication from active and passive attackers. TLS 1.3 is used in numerous applications like Web, e-banking, streaming, and more.

TLS runs over TCP. TLS 1.3 was designed for security and speed. In its most typical use, it can start encrypting data from the server to the client after two round-trips (RTT), one for the TCP handshake and one for the TLS 1.3 key exchange and authentication between client and server. Figure 1 shows a typical, simplified TLS 1.3 handshake from [32]. The protocol offers some even faster options that can send data after only the TCP handshake which are less commonly used. Even in the general case, sending data after two RTTs enables fast communications.

Most TLS 1.3 message exchanges today are small in size. The key exchange `ClientHello` and `ServerHello` messages carrying (Elliptic Curve) Diffie-Hellman ((EC)DH) public keys are just a few hundreds of bytes. The largest messages are the `Certificate` and the `CertificateVerify` which carry a certificate chain and a signature of the TLS transcript in order for the client to authenticate the server. The `Certificate` and the `CertificateVerify` amount to a few (2-5) KB when using RSA2048 or even less when using ECDSA certificates. Thus, the TLS 1.3 handshake amounts to just a few KB of data or a few packets. If the key exchange, `Certificate`, and `CertificateVerify` message sizes were to grow significantly, handshake speed would be impacted. There have been a lot of works which study the impact of larger key exchange and authentication messages in TLS 1.3. Most of these stem from the recent industry interest in migrating to quantum-resistant algorithms.

A concern for asymmetric cryptographic algorithms used today, including in TLS 1.3, is quantum computing. A cryptanalytically-relevant quantum computer (CRQC), if it were to become a reality, could implement quantum algorithms that would break (EC)DH key exchange and RSA or ECDSA signatures used in TLS. That is the reason why academia and industry have been working on new algorithms which are not known to be vulnerable against quantum algorithms. The US National Institute of Standards and Technology (NIST) has been working on standardizing some of these algorithms in its Post-Quantum (PQ) Project [27]. At the end of the Project's Round 3, it published new post-quantum algorithm draft standards [25], [24], [26] which include ML-KEM [25] as a Key Encapsulation Mechanism (KEM) for key exchange, ML-DSA [24] as the preferred, general-use signature scheme, and SLH-DSA [26] as a hash-based signature. FN-DSA will be the third signature to be published later. ML-KEM and ML-DSA are the general-use schemes which offer good perfor-

Workshop on Measurements, Attacks, and Defenses for the Web (MADWeb) 2024
1 March 2024, San Diego, CA, USA
ISBN 979-8-9894372-2-1
https://dx.doi.org/10.14722/madweb.2024.23xxx
www.ndss-symposium.org

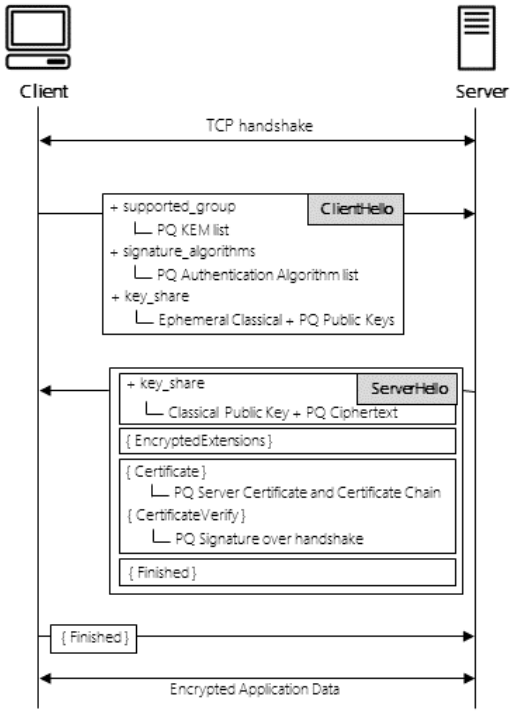


Fig. 1. A typical TLS 1.3 handshake [32] with application data flowing after two RTTs, one for the TCP handshake and one for the TLS key exchange and authentication. "{}" indicates encrypted messages protected using keys derived from the key exchange.

Parameter	Quantum-Resistant	Public Key (KB)	Ciphertext / Signature (KB)
ECDH P-384	✗	0.05	0.05
ML-KEM-512	✓	0.8	0.8
ML-KEM-768	✓	1.2	1.1
ML-KEM-1024	✓	1.6	1.6
ECDSA P-384	✗	0.05	0.1
RSA-3072	✗	0.4	0.4
ML-DSA-44	✓	1.3	2.4
ML-DSA-65	✓	2	3.3
ML-DSA-87	✓	1,6	4.6

TABLE I. ECDH, RSA, ECDSA, ML-KEM AND ML-DSA PUBLIC KEY, CIPHERTEXT AND SIGNATURE SIZES.

mance and relatively small key, ciphertext and signature sizes. They each offer three security levels: ML-KEM-512, ML-KEM-768, ML-KEM-1024 and ML-DSA-44, ML-DSA-65, ML-DSA-87 respectively. Table I shows the respective sizes for each parameter against the classical equivalents for ECDH, RSA, and ECDSA and demonstrates that using the quantum-resistant schemes in TLS 1.3 would significantly increase the `ClientHello`, `ServerHello`, `Certificate`, and `CertificateVerify` messages in a TLS 1.3 handshake.

A lot of works have been investigating the impact of new post-quantum algorithms on TLS 1.3 [20], [21], [19], [29], [35], [34], [3], [36]. Essentially all of these studies compared the TLS 1.3 handshake time of classical key exchanges and authentication against post-quantum ones and used it as an indication of the degradation the new algorithms will bring to TLS connections. Most post-quantum key exchanges in these works used post-quantum hybrid (PQ-hybrid) key exchanges which combine classical ECDH with a quantum-resistant KEM. This methodology ensures that if there is an unknown issue with the new KEM, the classical key exchange still ensures the security of the shared secret.

The handshake time measured in previous studies can be seen as the Time-To-First-Byte (TTFB) which corresponds to the time the application takes until it can start sending data over the secure tunnel. These comparisons were useful to quantify the overhead of each new algorithm introduced to the handshake. TLS 1.3 handshake time was the right metric to identify the best performing quantum-resistant schemes and how much overhead they were adding to classical handshakes. Previous analyses, showed that ML-KEM and ML-DSA or FN-DSA were the best performing KEM and signatures respectively. Unfortunately, the previously measured handshake performance degradation ignores the data transfer time over the secure connection which, together with the handshake time, is the total delay before the application can start processing data. As we now have reached a small set of quantum-resistant KEMs and signatures to be standardized (NIST Round 3), it is time to consider a metric which could more accurately capture the total impact of the new algorithms on the applications running over TLS 1.3. That metric is the total time from the connection start to the end of the data transfer, or Time-To-Last-Byte (TTLB).

Google's PageSpeed Insights [12] uses a set of metrics to measure the user experience and webpage performance. The First Contentful Paint (FCP), Largest Contentful Paint (LCP), First Input Delay (FID), Interaction to Next Paint (INP), Total Blocking Time (TBT), and Cumulative Layout Shift (CLS) metrics include this work's TTLB along with other client-side, browser application-specific execution delays. The PageSpeed Insights TTFB metric measures the total time up to the point the first byte of data makes it to the client. So, PageSpeed Insights TTFB is like this work's TTFB/TLS handshake time with additional network delays like DNS lookup, redirect, service worker startup, and request time.

As an example, the TTLB of a Web connection would correspond to the time from the connection initiation to the `load` event which fires after all the contents of an HTML page have been loaded. Arguably, the time from the connection initiation to an `DOMContentLoaded` event could also be used as a performance metric. A `DOMContentLoaded` event fires when the HTML page has been loaded without waiting for stylesheets, images, and subframes. For the purposes of our experiments we chose to use TTLB as a general metric for use-cases that go beyond the Web. Some applications could start processing data before the whole content has been transferred which means that these applications could use part of the TTLB as a performance benchmark.

How much TTLB slowdown introduced by a TLS algorithm change is acceptable highly depends on the application. [1] reported that 100ms affects customer conversion rates in retail sites and 2-3 extra seconds drive users away. Google's PageSpeed Insights [12] categorizes TTFBs $<$800ms as Good and $>$1.8s as Poor. Browsers like Chrome and Firefox, on the other hand, include performance regression tests for various page load tests for any new code changes. Chrome states that if you are a developer who "makes a change that regresses measured performance, you will be required to fix it or revert" [7]. They also offer an exception process for justified performance regressions. The Firefox performance sheriffing process triggers an alert when the performance regressions increase is $\geq$2% [11]. We can't tell with certainty what slowdowns are acceptable for all use-cases. Different applications will certainly be more forgiving than others.

In this manuscript, we analyze the impact of ML-KEM-768 and ML-DSA-44 or ML-DSA-65 on the TTLB of real-world TLS 1.3 connections which transfer hundreds of KB of data under fast, slow, stable and unstable networks. The rest of this document is organized as follows: Section I-A summarizes the new contributions of this work. Section II describes related work on the topic of post-quantum TLS 1.3 performance. Section III goes over our experimental setup and our testing methodology, Section IV presents the results and analysis of our experiments and Section V concludes the paper and summarizes future work.

A. Our Contributions

This work makes a few new contributions on the topic of TLS 1.3 performance with data-heavy handshakes:

1) It introduces the TTLB as a more accurate metric for the impact of new post-quantum TLS 1.3 handshakes on real-world TLS connections. Intuitively, the more data a connection transfers, the less impactful post-quantum handshake message sizes will be.
2) It experimentally shows that the impact of heavy post-quantum handshakes diminishes as the amount of data transferred over the tunnel increases.
3) It experimentally evaluates the impact of ML-KEM-768 and ML-DSA-44 or ML-DSA-65 in TLS 1.3 connections using TTLB under different network conditions and shows that although the handshake slowdown is higher under adverse network conditions, the overall connection time is less sensitive to such conditions. The TTLB impact in fast, stable networks remains below 5%. Slow, stable connections see their TTLB drop from 32% to $<$10% when transferring 100KiB of data or more. Even TCP related congestion control slowdowns are amortized to $<$5% when the transferred data goes to 200KiB.
4) It experimentally shows that under unstable or lossy conditions where the impact of ML-KEM-768 and ML-DSA-44 or ML-DSA-65 on the TLS 1.3 handshake is high, the absolute time of classical TLS 1.3 handshakes is already suffering and showing high volatility. Put differently, the new data-heavy TLS handshakes will not be "the deal breaker" for connections in unstable or lossy networks.

II. Related Work

There have been many works which evaluate the impact of NIST quantum-resistant algorithms on TLS 1.3. All of these studies look into the impact introduced by the increased key exchange and authentication information exchanged and the CPU overhead. Early experimental deployments by Google and Cloudflare [20], [21], [19] showed that new KEMs can perform well under normal conditions but there will be more volatility and higher impact at the tails of these connection distributions. They also observed that network middleboxes can sometimes cause failures for larger TLS 1.3 `keyshare`s. Another early experimental study [29] showed that large keys and signatures affect the TLS 1.3 handshake more, especially at the 90-th percentile under higher network loss probabilities. An intuitive explanation for this is that more packets in the handshake means higher total loss probability, resulting in handshake slowdown. The authors also showed that the impact of data-heavy KEMs and signatures on webpage load time diminishes as the size of the webpage increases. Load time is a metric similar to the TTLB, but it may include application-specific delays the cryptographic algorithms are not responsible for.

[34], on the other hand, evaluated the combination of post-quantum hybrid key exchanges and post-quantum signatures in TLS 1.3 and SSH. It showed that well-performing quantum-resistant KEMs and signature schemes can be used in the protocols without significantly degrading their handshake performance. The authors also observed that large or CPU-intensive quantum-resistant schemes have higher impact on the handshake and explored tweaking the TCP initial congestion window `initcwnd` to speed it up.

Other works focused solely on authentication. [35] showed that data or CPU-heavy post-quantum signatures slow down the TLS 1.3 handshake and could even introduce an extra RTT due to the TCP `initcwnd`. Sikeridis et al. proposed using different algorithms throughout the certificate chain in order to control the size and speed of the handshake. [30] explored the same premise; combining different signatures in the certificate chain to trim the chain size and showed promising results Subsequently, Cloudflare experimentally tested [3] various TLS 1.3 certificate chain sizes and showed that for their traffic profiles, more than 9-10KB of authentication data from the server will slow down the handshake more than 10-20% at high percentiles. To address the quantum-resistant TLS authentication data size issue, various works have proposed to slim it down [17], [33]. Similar concepts have been proposed

in the IETF for cutting down the authentication data size [16], [4], [15].

[5] and [22] evaluated post-quantum algorithms for TLS in constrained environments and showed that although promising, they would introduce performance degradation for some constrained use-cases.

More recently, other studies [14] have confirmed the results in previous works. [10] experimentally evaluated the NIST Round 3 algorithm picks' CPU performance and their impact on the mean TLS handshake time. The results were in-line with previous studies but did not investigate the higher latency percentiles. [36] demonstrated that well-performing KEMs and signatures lead to mean TLS handshakes times which are close to classical TLS 1.3 handshakes even under lossy network conditions. Sosnowski et al. did not investigate higher latency percentiles which would have seen higher impact by the post-quantum handshakes.

NIST National Cybersecurity Center of Excellence's (NCCOE) Post-quantum Migration Project [23] recently published a report draft [28] which confirms some of the results of the aforementioned studies and points out the need for evaluating the TTLB as a performance metric. The NIST NCCOE report also includes results of the impact of PQ KEMs and signatures on QUIC [37], a widely-used transport protocol for the Web which leverages TLS 1.3 for session establishment. It shows that QUIC congestion control, amplification protection, and packet pacing can introduce significant slowdowns to a QUIC handshake. Finally, [18] discussed some potential challenges and changes needed to address issues introduced by the new post-quantum algorithms. It also proposed the time-to-last-byte performance metric. **Full-disclosure**, one co-author of [28] and [18] which propose the use of the TTLB metric is also a co-author of this manuscript.

III. Experiments

We designed our experiments to simulate various network conditions and measure the TTFB and TTLB of the TLS 1.3 connection. The TTFB allows us to compare our results with existing studies. The TTLB represents a more realistic measurement of impact of data-heavy key exchange and authentication on TLS 1.3 connections. Both the TTFB and TTLB in our measurements included the initial TCP handshake time (one round-trip). For example, a typical TLS 1.3 handshake time (TTFB) is equivalent to the TTLB of a 0-Byte data transfer which amounts to two RTTs under normal network conditions. We felt that including the TCP handshake time was a more accurate measurement of the time it takes for an application to start sending or receiving data over the secure connection. The amount of data transferred in our experiments consisted of typical client-server scenarios where the client makes a small request and the server responds with hundreds of Kilobytes of data. We did not test client-to-server transfers; their performance can be extrapolated from the server-to-client experimental results below.

The testing methodology we used follows the same rationale as in [29, §3]. We used Linux `namespaces` in a Ubuntu 22.04 virtual machine instance. The `namespaces` were interconnected using `virtual ethernet` interfaces. The "network" between the `namespaces` was emulated using

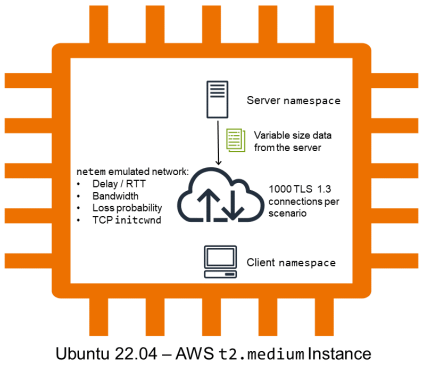


Fig. 2. The experimental setup between the client and server Linux `namespaces` and `netem`-emulated network conditions. Each scenario included 1000 TLS 1.3 connections for different data size downloads from the server. The collected data per connection included the connection TTLB and various packet statistics.

Linux kernel's `netem` utility which can introduce variable network delay, interface bandwidth, and packet loss probabilities between the client and server. The Ubuntu 22.04 EC2 instance was a `t2.medium` (2x64-bit Intel(R) Xeon(R) CPU E5-2686 v4 @ 2.30GHz, 4GB RAM) in the AWS Cloud. Figure 2 summarizes the setup.

Our experiments included sevaral parameters:

➜ Key exchange mechanism. That was classical ECDH with curve P256 for the classical connections or PQ-hybrid with ECDH using P256 and ML-KEM-768 for the quantum-resistant ones.

➜ Certificate chain size from the server. That was set to a typical 2.5KB (on the wire) classical chain, a medium simulated ML-DSA-44 post-quantum chain (8KB on the wire), and a large (16KB on the wire) simulated ML-DSA-65 chain. These certificate chains were constructed using RSA, not ML-DSA certificates, as there was no ML-DSA support in our testing toolset at the time of our experiments. The 8KB chain was only used to compare our results with previous studies. The reason we picked 16KB for the post-quantum chain is to investigate if ML-DSA certificates would significantly affect the total connection time. A typical TLS 1.3 handshake today (with a certificate chain of one end-entity, one intermediate Certificate Authority (CA), and one Root CA cert) includes three Signed Certificate Timestamps, one `CertificateVerify` and two X.509 signatures and two public keys.That would amount to $\sim$14 and $\sim$19KB of "TLS 1.3 authentication data" on the wire for ML-DSA-44 and ML-DSA-65 respectively. If we used one of the simple approaches [17], [16], [15] proposed to slim down the authentication data from the server by omitting the intermediate CA certificate, the "authentication data" on the wire would become $\sim$11 and $\sim$14KB respectively. With ML-DSA-87, it would be even higher ($\sim$20KB). So, we picked an inflated certificate chain which amounted to 16KB on the wire as a representative "TLS authentication data" size with ML-DSA-44 or ML-DSA-65 certificates. This value slightly exceeds the default TCP `initcwnd` of 10 times the Maximum Segment Size (MSS) [8] and the critical 9-10KB size identified as a performance turning point for TLS 1.3 Web connections in [3], so it is a good representative size to test with. We did not test Mutual TLS connections with large client PQ chains; their performance can be extrapolated from the regular TLS authentication results below.

Note that due to their high compute performance (exceed-

ing or equivalent to their classical counterparts), ML-KEM and ML-DSA's impact on the TLS 1.3 handshake will mostly come from their larger public key and signature sizes. That has been proven in previous works [35], [30], [36] and is generally confirmed by our testing.

➜ TCP `initcwnd`. As this value could introduce an extra round-trip for the large certificate chains [35], [34], we picked initial window values of 10 or 20 times the server's MSS. The former was only used when evaluating the impact of the extra RTT due to TCP congestion control to the total connection time. The latter was enough to eliminate the RTT so we can investigate other experimental parameters.

➜ Network delay between client and server. That was set to 35, 70, or 200ms. These RTTs do not represent typical datacenter round-trips, which tend to be much smaller. Packet size MTUs are typically much higher in datacenter environments, so we chose to not consider such scenarios where post-quantum key, ciphertext, and signature sizes will all fit in larger frames and connections will complete much faster. 35ms is a typical RTT for today's Internet where services are often localized. We chose 200ms as the RTT to emulate constrained cellular networks or long distance connections. Based on [9, §V], 200ms would be the RTT for LTE-M cellular networks for distances of 15Km. According to [39, §4.4], 4G and 5G mobile network round-trips can vary between 40-120ms for up to 2.5Km distances. For our "volatile network" condition tests we used `netem` to generate a `Paretonormal`-distributed network delay distribution with a mean of 35 or 70ms and jitter of 35/4 or 70/4 ms respectively. These were designed to emulate highly volatile network conditions with high variability.

➜ Bandwidth between client and server. That was set to 1Mbps or 1Gbps. 1Gbps is typical "fast" bandwidth for modern networks. 1Mbps was chosen to emulate more constrained environments, legacy, or cellular networks. Based on [9, §V], 1-2Mbps would be the bandwidth for LTE-M cellular networks for distances up to 20Km. According to [39, §5.2], 4G and 5G mobile network speeds vary from 10-200Mbps or higher depending on TCP congestion control.

➜ Loss probability per packet. This was set to uncorrelated 0, 1, 3, 5 or 10% . The higher loss probabilities would be for more unstable network conditions. We chose 10% to emulate an LTE-M cellular network for up to a distance of 15Km [9, §V]. According to [39, §4.1], 4G and 5G mobile network loss can vary between 0-4%.

➜ Data size transferred from the server to the client. That was set to 0, 50, 100, 200 or 300 Kibibytes (KiB, i.e. 2^{10} bytes). [13] shows that the average webpage transfers upwards of 2MB of data over 10-11 connections which amounts to 150-200KB per connection. Additionally, TLS connections to cloud services transfer highly variable sizes of data from the server depending on the use-case. The transfer sizes commonly surveyed start from 30-50KB from the server and can reach hundreds of Kilobytes sent over ∼5-100 requests reusing the same connection. So, we chose 0-Byte transfers to measure TTFB and 50, 100, 200, 300KiB to measure the TTLB for real-world connections which transfer sizable, but not very large, amounts of data. We expect that the impact of the PQ-hybrid ML-KEM / ML-DSA handshakes on higher size transfers will be significantly minimized.

We used `s2n-tls` [2], AWS' open-source implementation of TLS, for our experiments [6]. All our tested scenarios ran 1000 serial connections between the client and server. For each scenario, we calculated the 50-th, 75-th and 90-th percentile of the TTLB or TTFB (handshake) times. Google's PageSpeed Insights [12] considers the 75-th as appropriate to cover the majority of user experiences. We added the 90-th percentile to include even more of the tail-end of the device and network conditions.

IV. Results and Analysis

Our experiments implemented the methodology described in Section III. Our goals were to:

➥ Evaluate the RTT impact introduced by TCP congestion control with large certificate chains. We wanted to quantify how much the RTT impact is amortized over the amount of data sent by the server.

➥ Compare our experimental results against other studies to confirm that our experiments align previous work and its results.

➥ Assess the impact of the PQ-hybrid handshake on the TTLB and confirm or debunk the intuition that a few extra KB of handshake data would be unnoticeable in the total connection time especially for stable, non-lossy networks.

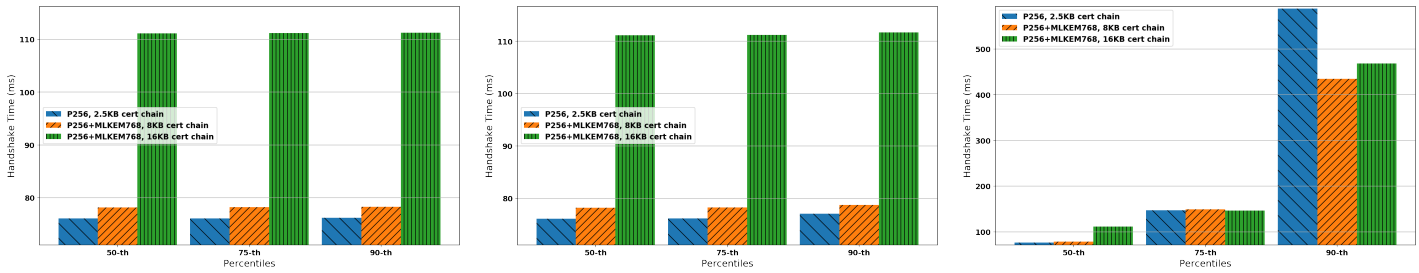
➥ Evaluate the impact of the PQ handshakes on the total connection time in fast, slow, stable, unstable, or volatile network conditions. We wanted to isolate each testing scenario so we focus on each use-case separately. Sometimes aggregating many traffic profiles in one experiment can hide specific traffic profile behaviors in the aggregate percentiles.

A. Impact on the TLS 1.3 handshake

We first investigated the impact of the new post-quantum algorithms on the TLS 1.3 handshake. We wanted to confirm that our testing methodology leads to the same results as in previous studies. We looked into the impact of the new algorithms with both default [8] and increased TCP `initcwnd`.

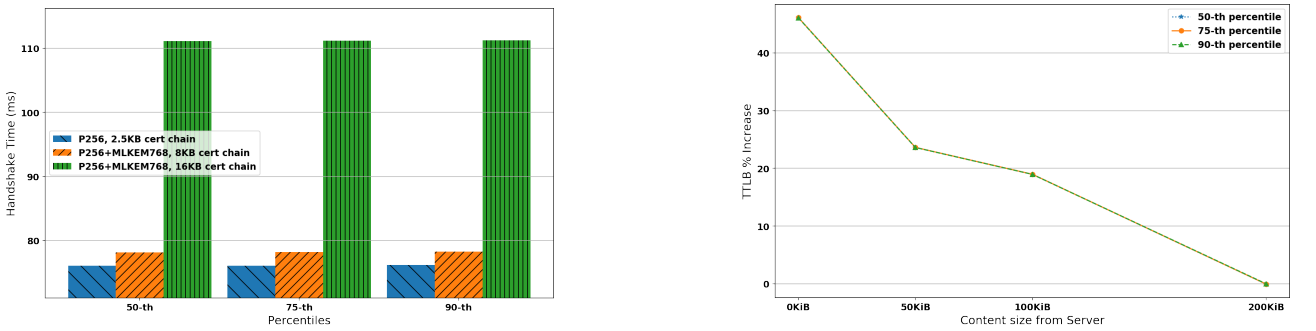
1) TCP `initcwnd`=10: The impact of the `initcwnd` has already been analyzed in other works [35], [34]. In summary, if the server `Certificate`+`CertificateVerify` message sizes exceed the `initcwnd`, the server has to wait for a TCP `ACK` after sending `initcwnd` of data. Given the size of ML-DSA-44 and 65, this could happen for servers using the default [8] `initcwnd`=10MSS.

We tested `initcwnd`=10MSS to evaluate if the cost of an extra round-trip diminishes as connections transfer more data. Figure 3 shows the TLS 1.3 handshake time for connections with various loss probabilities at 1Gbps bandwidth and 35ms RTT. Figure 3a and Figure 3b show that the "authentication data"-heavy connections (16KB ML-DSA chain) include an extra round-trip for low loss probabilities which makes them significantly slower. These results are in-line with previous works [35], [34]. Figure 3c shows that as the loss probability increases, the volatility and unpredictability of these connections can sometimes even lead to slower classical handshakes. Re-running our experiments for the 5% loss scenario produced volatile results. Note that the handshake time for all of these lossy handshakes are significantly (6-8×) slower than the classical one with 0% loss at ∼76ms. This indicates that lossy networks in themselves have a substantial impact on handshake times.



(a) Loss probability=0% (b) Loss probability=1% (c) Loss probability=5%

Fig. 3. TLS 1.3 handshake time (ms) for classical and PQ connections and different loss probabilities (%). Bandwidth=1Gbps. RTT=35ms. TCP `initcwnd`=10.



(a) Handshake time (ms)

(b) TTLB % increase between classical (P256, 2.5KB chain) & PQ (P256+MLKEM768, 16KB chain)

Fig. 4. TLS 1.3 performance for classical and PQ TLS 1.3 connections, different percentiles, and transferred data sizes from the server. Loss probability=0%. Bandwidth=1Gbps. RTT=35ms. TCP `initcwnd`=10.

We then collected the TLS 1.3 performance for the same connections for different data sizes transferred from the server (Figure 4). Figure 4a shows the 50-th, 75-th, 90-th percentiles for the TLS handshake time for the classical and post-quantum connections for 0% loss probability, 1Gbps bandwidth and 35ms RTT. We see that all connection latencies are roughly aligned, except for the post-quantum connections with the large 16KB chain. These connections include an extra round-trip due to TCP congestion control's interaction with the large certificates. Figure 4b shows the TTLB % increase between the classical (P256 key exchange, 2.5KB chain) and the PQ connection (P256+MLKEM768 key exchange, 16KB chain) for different application data transfer sizes at each percentile. We see the extra RTT at 0KiB, but it gets amortized as the server data increases and the % increase goes to zero when the RTTs are the same at 200KiB. The slowdown disappears because the TCP congestion window grows to complete the 200KiB transfer. Sending the 16KB certificate chain in the handshake increases the congestion window early and leads to the same amount of RTTs for the classical and post-quantum connections. Depending on the size of that data, the classical connection could be one RTT faster or not. At higher loss probabilities, higher latency percentiles would naturally be more volatile than what we measured at 0% loss.

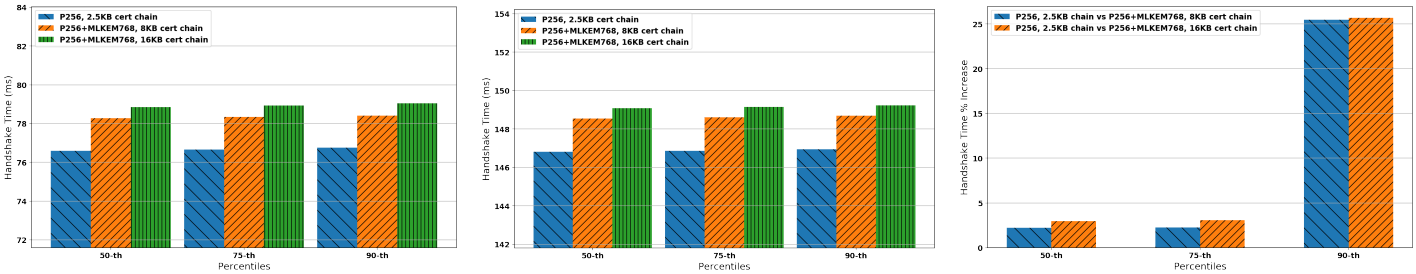
In summary, using the default TCP `initcwnd`=10 with data-heavy TLS 1.3 authentication would slow down the handshakes by one RTT. The slowdown diminishes as the data transfer size increases and could drop to zero depending on how the TCP window aligns with the data size.

2) TCP `initcwnd`=20: Content providers today often increase their TCP `initcwnd` to speed up connections [31]. For example, [3] discusses how Cloudflare uses `initcwnd`=30. This would prevent the extra round trip introduced by TCP congestion control due to ML-DSA authentication as shown in in Section IV-A1.

In Figure 5, we tested the classical (P256 key exchange, 2.5KB chain) and post-quantum (P256+MLKEM768 key exchange, 8KB and 16KB cert chain) TLS 1.3 handshakes for a 35 and 70ms RTT under 1Gbps bandwidth and various loss probabilities. We wanted to see if our results would match previous studies. Figure 5a and Figure 5b show the handshake times for all percentiles for 35 and 70ms RTTs respectively at 0% loss probability. We can see that the the post-quantum handshakes are slightly slower than the classical ones but only by a few milliseconds. These results match previous works [35], [30], [36].

Figure 5c shows the handshake time % increase for each percentile for the aggregate measurements (all RTTs, and all loss probabilities 0, 1, 3, 5%). They show the % handshake time increase between the classical and the post-quantum handshakes. We observe that the latter ones are slower especially at the 90-th percentile. Intuitively, as these handshakes send more packets, network packet loss would affect them more. Note that we expected the 8KB chain to be affected significantly less than the 16KB one which is not reflected in these results.

Generally, the results in Figure 5c did not match the findings in [3] which combined various traffic profiles. [3] showed that the handshake time was affected $<$10% at the 90-th percentile for $<$10KB chains which was not the case in our experiments. We believe this inconsistency was due to the high network bandwidth (1 Gbps) of our traffic profile which was likely higher than traffic profiles measured in [3].

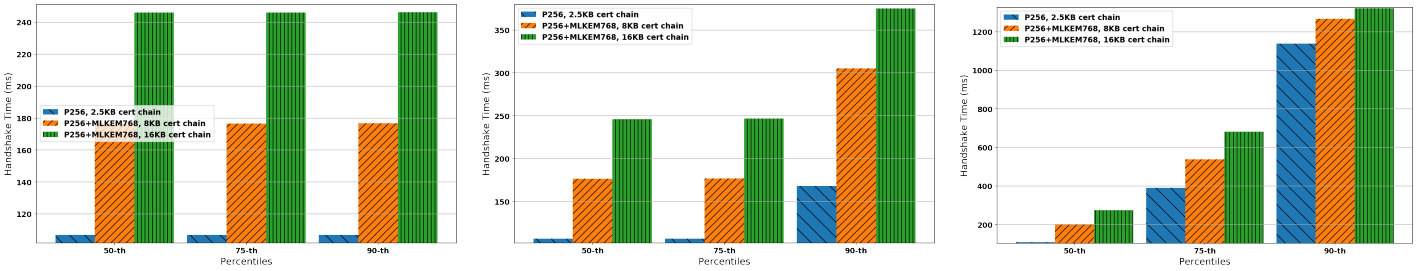


(a) Handshake time (ms).
RTT=35ms.
Loss probability=0%

(b) Handshake time (ms).
RTT=70ms.
Loss probability=0%

(c) Handshake time % increase.
RTT=35, 70ms
Loss probability=0, 1, 3, 5%

Fig. 5. TLS 1.3 handshake for classical and PQ connections and different loss probabilities (%). Bandwidth=1Gbps. RTT=35, 70ms. TCP `initcwnd`=20.



(a) Loss probability=0%

(b) Loss probability=3%

(c) Loss probability=10%

Fig. 6. TLS 1.3 handshake time (ms) for classical and PQ connections and different loss probabilities (%). Bandwidth=1Mbps. RTT=35ms. TCP `initcwnd`=20.

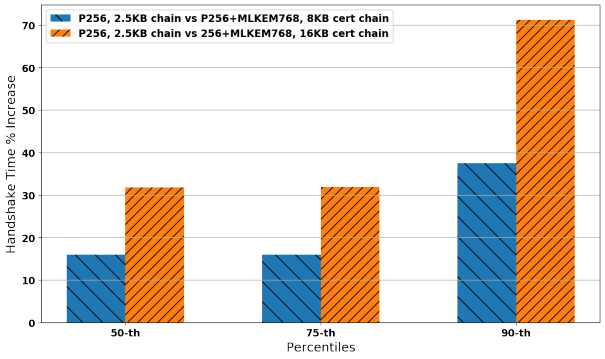


Fig. 7. TLS 1.3 handshake time % increase between classical & PQ TLS 1.3 connections. Bandwidth=1Mbps. Loss probability=0, 1, 3, 10%. RTT=35, 200ms. TCP `initcwnd`=20.

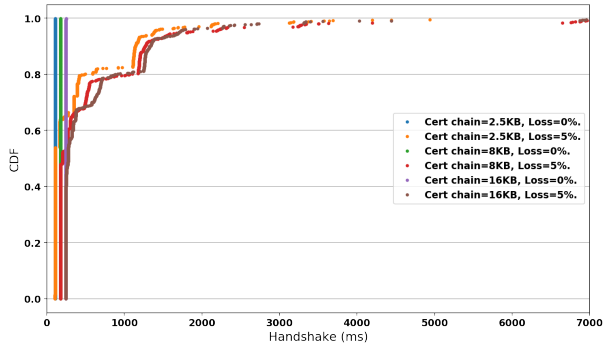
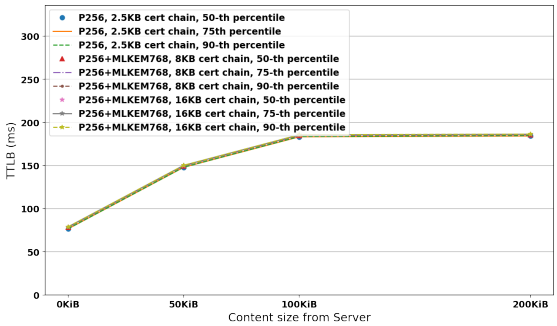


Fig. 8. Classical (P256, 2.5KB chain) & PQ (P256+MLKEM768) TLS 1.3 handshake time (ms) Cumulative Distribution Function (CDF) for different certificate chain sizes. 35ms RTT. TCP `initcwnd`=20.

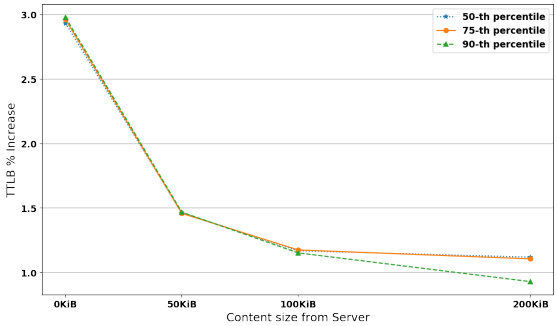
To test that theory, we experimented with a lower bandwidth of 1Mbps to represent constrained networks. Figure 6 shows the classical (P256 key exchange, 2.5KB chain) and post-quantum (P256+MLKEM768 key exchange, 8KB and 16KB chain) handshake time for different loss probabilities at 1Mbps bandwidth and 35ms RTT. We can see that a slower network shows more relative slowdown for the 16KB chain than the 8KB one, especially for low loss probabilities. Intuitively, with slower network speeds, the transfer time for 8-10KB of post-quantum handshake data will be more noticeable. At higher loss probabilities, the slowdown is less pronounced due to the instability of these connections.

To confirm the volatility of the handshake time at higher loss probabilities, we plotted the Cumulative Distribution Function (CDF) of the TLS 1.3 handshake time at 1Mbps bandwidth and 35ms RTT. Figure 8 shows that the classical and post-quantum handshakes have little volatility at 0% loss, but at 5% they start to increase after 50% of the sample. We notice that the classical handshakes are less volatile at 5% loss, but not significantly compared to the PQ ones with 8KB or 16KB chains.

Figure 7 shows the percentiles of the aggregate datasets collected for 1Mbps bandwidth, 0, 1, 3, 10% loss probability, and 35ms and 200ms RTT. 1Mbps, 200ms RTT with 10% loss probability could correspond to LTE-M cellular networks for up to 15Km distances [9, §V]. We can see that the 16KB chain is almost twice slower than the 8KB chain. Generally, this matches the results in [3]. Without knowing the specific traffic profiles of experiments in [3], we estimate that they included a mix of fast, slow, stable and unstable connections. Naturally, below (Section IV-B) we investigate the TTLB beyond the



(a) TTLB (ms)



(b) TTLB % increase between classical (P256, 2.5KB chain) & PQ (P256+MLKEM768, 16KB chain)

Fig. 9. TTLB for classical and PQ TLS 1.3 connections at 0% loss probability. Bandwidth=1Gbps. RTT=35ms. TCP `initcwnd`=20.

handshake for TLS 1.3 connections under different network conditions.

In summary, under "fast and stable" network conditions and a high TCP `initcwnd`, a few more KB in the handshake will not slow it down significantly. The slowdown will be more noticeable in slower networks especially with large ML-DSA size certificate authentication.

As a note specifically about cellular networks, TCP itself has been shown to affect network performance similarly to how new PQ algorithms will affect it. For example, [38] proposed bandwidth probing in LTE instead of TCP's slow-start to improve TCP performance. Such changes are not straightforward to deploy on the Internet. TCP, TLS, and other fundamental protocols are likely to remain universal regardless of their use-case.

B. Impact on the TLS 1.3 Connection

After investigating the impact of data-heavy cryptographic algorithms on the TLS 1.3 handshake and confirming observations in previous works, we proceeded with evaluating the impact on the total connection time (TTLB) for common connection data sizes on the Internet today. For the rest of this document, we use TCP `initcwnd`=20 as we already showed (Section IV-A1) that if congestion control interferes with these connections, we can assume some slowdown (one RTT) which drops as the connection data increases.

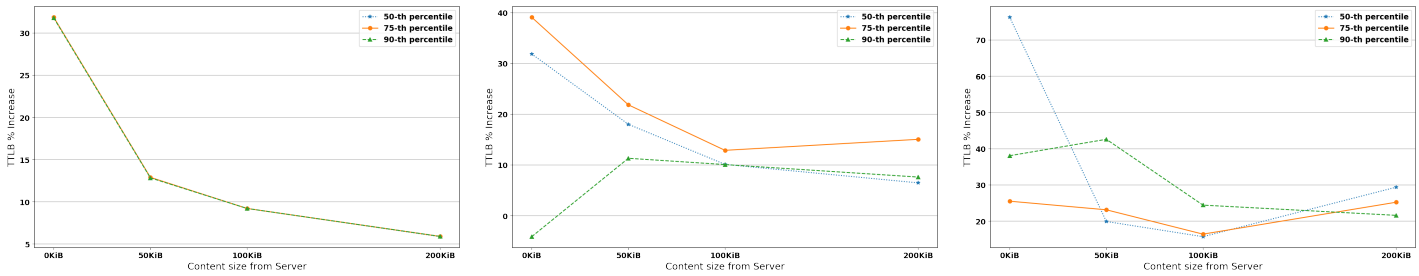
1) Under "good" network conditions: We initially focused on the TTLB under "good" network conditions at high speeds and low loss probabilities. Figure 9 shows the TTLB for classical (P256 key exchange, 2.5KB cert chain) and post-quantum (P256+MLKEM768 key exchange, 16KB chain) TLS 1.3 connections carrying 0-200KiB of data from the server, at 1Gbps bandwidth, 0% loss, 35ms RTT. In Figure 9a, we can see that under ideal conditions, the TTLBs for all connections are almost identical. Similarly, Figure 9b shows the TTLB % increase of the post-quantum connection over the classical one for all percentiles. We can observe that although the slowdown is low (∼3%) at 0KiB from the server (equivalent to the handshake), it drops even more (∼1%) as the data from the server increases. At the 90-th percentile the slowdown is slightly lower. These slowdowns would also be considered low for some online retail users and certain browser requirements [1], [7], [11]. The results generally match [29, §5.2] which demonstrated that as the downloaded data increases, data-heavy key exchanges end up leading to similar load times.

In summary, under "fast and stable" network conditions, a data-heavy (ML-KEM, ML-DSA) TLS 1.3 handshake will not impact the TTLB for most common connections that transfer tens of KBs of data.

2) Under low bandwidth network conditions: Section IV-A2 discussed TLS handshake performance in low bandwidth networks and showed that it can be significantly slower depending on the size of the handshake. We now look into the overall impact (TTLB) on real-world, low-bandwidth connections.

Figure 10 shows the % increase of the TTLB between the classical (P256 key exchange, 2.5KB chain) & PQ (P256+MLKEM768 key exchange, 16KB chain) TLS 1.3 connections carrying 0-200KiB of data from the server for each percentile at 1Mbps bandwidth, 200ms RTT and 0, 3, 10% loss probability. We modeled this as a typical low bandwidth network with "some instability". 1Mbps, 200ms RTT with 10% loss probability could correspond to LTE-M cellular networks for up to 15Km distances [9, §V]. We can see that at low loss probabilities (Figure 10a), all three percentiles are almost identical. The TTLB increase is high (∼33%) at 0KiB from the server. This is equivalent to just the TLS handshake increase shown in Figure 6a. As the data size from the server increases, the TTLB increase drops to ∼6% because the handshake data size is amortized over the connection. As the loss probability increases, we notice volatility in the TTLBs. At 3% loss (Figure 10b), we observe that the percentiles converge between 10-20%. Note that as the loss probability increases the volatility of the experimental measurements go up. This is also the case for the TTFB in Figure 8.

Figure 10c shows that at 10% loss, the TTLB increase settles between 20-30% for all percentiles. The same experiments for 35ms RTT produced similar results. Although, 20-30% increase may seem high, we note that re-running the experiments could sometimes lead to smaller or higher percentile increases because of the general "network instability" of the scenario. Also, bear in mind that classical connection percentile TTLBs for 200KiB from the server at 200ms RTT and 10% loss were [4644, 7093, 10178]ms, whereas their post-quantum connection equivalents were [6010, 8883, 12378]ms. At 0% loss they were [2364, 2364, 2364]ms. So, although the



(a) Loss probability=0% (b) Loss probability=3% (c) Loss probability=10%

Fig. 10. TTLB % increase between classical (P256, 2.5KB chain) & PQ (P256+MLKEM768, 16KB chain) TLS 1.3 connections for different loss probabilities (%). Bandwidth=1Mbps. RTT=200ms. TCP `initcwnd`=20.

percentiles increased by 20-30% in the post-quantum connections compared to the classical ones, the classical connections are already impaired (by 97-331%) due to network loss. An extra 20-30% is not likely to be impactful in an already highly-degraded connection time.

We also wanted to confirm that the TTLB increase would continue its downward trend for higher data transfers from the server as observed in Figure 10. To achieve that, we ran the experiments for 300KiB of server data for 35 and 200ms RTTs. Table II summarizes the results. We can see that the TTLB increase remains under 10% for low loss probabilities. It increases for higher loss rates but stays within 10-20% even at the 90-th percentile. Note that, as with the 200KiB case, a packet loss rate of 10% had already introduced 37%, 71%, 115% total connection slowdown for the 50-th, 75-th and 90-th percentiles respectively when the RTT was 35ms. The equivalent slowdowns with an RTT of 200ms were 170%, 292%, and 423%. So, even the 90-th percentile PQ TTLB slowdown in the table is minimal compared to the slowdown already introduced by the network's "lossiness".

RTT (ms)	Loss probability (%)	Percentiles		
		50-th	75-th	90-th
35	0	5.09	5.09	5.08
	1	5.09	5.08	4.27
	3	5.10	5.01	5.79
	10	13.69	16.51	15.82
200	0	4.31	4.31	4.31
	1	4.31	5.28	5.47
	3	6.19	11.73	12.98
	10	20.81	14.90	8.03

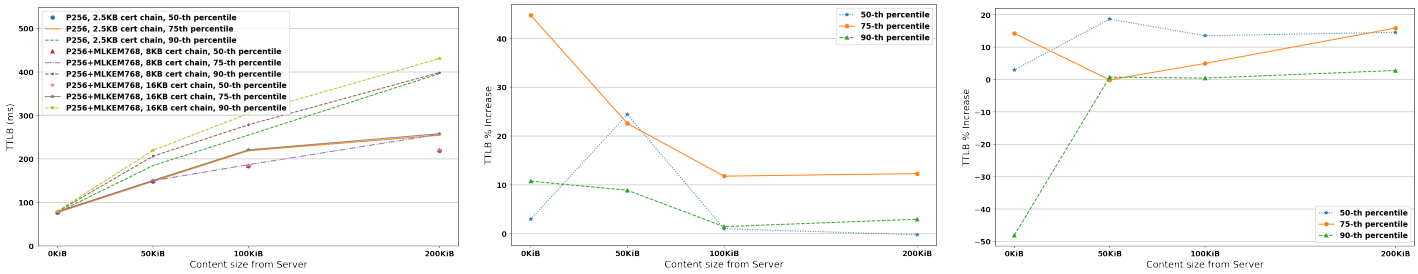
TABLE II. TTLB % INCREASE BETWEEN CLASSICAL (P256, 2.5KB CHAIN) & PQ (P256+MLKEM768, 16KB CHAIN) TLS 1.3 CONNECTIONS TRANSFERRING 300KIB FROM THE SERVER FOR DIFFERENT LOSS PROBABILITIES (%) AND RTTS. BANDWIDTH=1MBPS. TCP `INITCWND`=20.

In summary, low bandwidth TLS 1.3 connections will see some impact by post-quantum algorithms on the total connection time. This impact decreases as the transferred data increases. Low bandwidth connections in lossy conditions will be affected more, especially at higher percentiles, but the impact on total connection time will be more moderate than on the handshake alone. We should also not disregard that lossy, low bandwidth connections which transfer sizable amounts of data have already been slowed down by the network conditions much more than by the data-heavy algorithms.

3) Under unstable network conditions: After evaluating stable and fast, and low bandwidth networks, we wanted to investigate more unstable network conditions to assess if data-heavy handshakes would be more impactful on their TTLB. Our intuition was that an unstable network would see more impact when carrying more data in the handshake, but that it may not be noticeable when transferring sizable amounts of application data over the connection. We experimented with different loss probabilities and delay distributions to confirm or debunk this theory.

Figure 11 shows the TTLB for classical (P256 key exchange, 2.5KB chain) and post-quantum (P256+MLKEM768 key exchange, 8KB or 16KB chain) TLS 1.3 connections carrying 0-200KiB of data from the server for various loss probabilities at 1Gbps bandwidth and 35ms RTT. At 1% loss probability (Figure 11a), we see that the 50-th and 75-th percentiles overlap for all testing scenarios. The TTLB naturally increases as the transferred data size increases. The 90-th percentile presents some differentiation as the network instability affects the 16KB cert chain connections more than the 8KB one. The 2.5KB chain connections are also faster than the 8KB ones. So, at 1% loss, we do not see significant difference between classical and post-quantum connections except for the 90-th percentile where the ML-DSA certificates impose a 0-10% extra slowdown. As the loss probability increases (Figures 11b and 11c), we see more volatility in the TTLB % increase of the post-quantum connections over the classical ones. We see that the post-quantum connections converge to a 10-20% increase although they start higher for the handshake itself (0KiB). We also note that the measured times at higher loss rates demonstrated high fluctuations. Re-running the experiments could skew the results one way or the other. We note that even the classical connections are already significantly impaired at high loss probabilities which means that a 20% worst-case slowdown introduced due to to the data-heavy handshake will not be detrimental. For example, the classical 200KiB connection TTLB percentiles went to [544, 806, 1332]ms at 5% loss from [275, 295, 313]ms at 0% which is a drastic slowdown. The equivalent post-quantum percentiles were [624, 967, 1500]ms. So, the PQ slowdown is less noticeable compared to one introduced by the network loss probability for the classical connections.

All our experiments so far have shown that as the network loss probability increases, the TTFB and TTLB times fluctuate more which can sometimes end up making the classical connections worse than we expect compared to the post-quantum

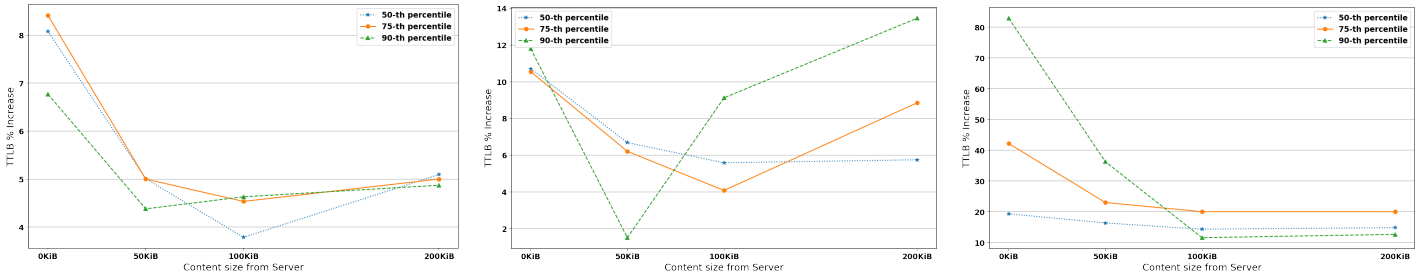


(a) TTLB (ms). Loss probability=1%.

(b) TTLB % increase between 2.5 & 16KB chain. Loss probability=3%.

(c) TTLB % increase between 2.5 & 16KB chain. Loss probability=5%.

Fig. 11. TTLB for (P256 key exchange, 2.5KB chain) and post-quantum (P256+MLKEM768 key exchange, 16KB chain) TLS 1.3 connections and different loss probabilities (%). Bandwidth=1Gbps. RTT=35ms. TCP `initcwnd`=20.



(a) Loss probability=0%

(b) Loss probability=1%

(c) Loss probability=5%

Fig. 12. TTLB % increase between classical (P256, 2.5KB chain) & PQ (P256+MLKEM768, 16KB chain) TLS 1.3 connections for different loss probabilities (%) under "volatile network" conditions. Bandwidth=1Gbps. RTT=35ms. TCP `initcwnd`=20.

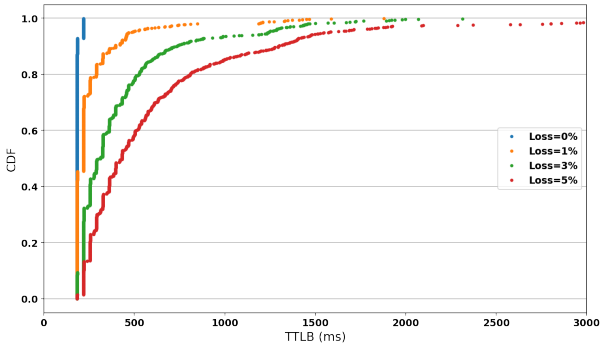


Fig. 13. TTLB CDF for classical TLS 1.3 connections. 200KiB from the server. 1Gbps bandwidth. 35ms RTT.TCP `initcwnd`=20.

ones. We attribute that to the randomness of packet loss which can skew the percentiles one way or another. In order to confirm that, we plotted the CDF for the classical TLS 1.3 connection TTLB carrying 200KiB from the server at 1Gbps bandwidth and 35ms RTT. Figure 13 shows that at 0% loss the TTLB remains pretty stable, but it starts to increase after 50% of the sample at 1% loss. At 3% and 5% loss the TTLB starts to increase even earlier. That explains the volatile TTLB times we have observed throughout our experiments at the 90-th percentile with higher loss probabilities and they match similar observations with the TTFB (Figure 8).

Our last set of experiments introduced variable network delays between the client and the server. We used `Paretonormal`-distributed network delay with a mean of 35 or 70ms and jitter of 35/4 or 70/4 ms respectively. These were chosen to emulate highly volatile network conditions. Figure 12 shows the TTLB % increase between classical (P256 key exchange, 2.5KB chain) & PQ (P256+MLKEM768 key exchange, 16KB chain) TLS 1.3 connections for different loss probabilities (%) and data sizes transferred from the server. We can see that the post-quantum connection TTLB increase starts higher at 0KiB server data and drops to 4-5% for 0% loss and 10-20% for higher loss probabilities. As with previous experiments, the percentiles were more volatile at higher the loss probabilities, but overall the results show that even under "volatile network conditions" the TTLB drops to as the transferred data increases.

We also wanted to confirm that the TTLB increase for higher data transfers from the server would continue its downward trend observed in Figure 12. We ran the `Paretonormal`-distributed network delay experiments for 300KiB of server data. Table III summarizes the results. We can see that the TTLB increase remains under 10% for low loss probabilities. It goes up for higher loss probabilities but stays within 10-20% even at the 90-th percentile. In some cases, the high instability makes the post-quantum total connection time slightly faster that the classical one (negative % in the table). As with the 200KiB case, the network "instability" had already introduced [133, 246, 336]% and [129, 210, 307]% total classical connection slowdown for the 50-th, 75-th and 90-th percentiles for 35 and 70ms RTTs respectively. Additionally, we noticed volatility when re-running the experiments for all measurements (even for the classical TLS connections) due to variable delay distribution.

To observe the volatility of the TTLB for all "unstable network conditions" in our experiments (higher network loss probabilities, `Paretonormal`-distributed network delay), we

RTT (ms)	Loss probability (%)	Percentiles 50-th	75-th	90-th
35	0	-8.45	-8.90	-8.12
	1	-7.03	3.97	4.39
	3	-2.81	-0.97	-1.23
	5	8.713	7.96	14.51
70	0	-5.89	-7.61	-8.25
	1	-6.33	-8.18	-4.58
	3	2.27	0.94	7.05
	5	9.06	8.67	5.00

TABLE III. TTLB % INCREASE BETWEEN CLASSICAL (P256, 2.5KB CHAIN) & PQ TLS 1.3 (P256+MLKEM768, 16KB CHAIN) CONNECTIONS TRANSFERRING 300KIB FROM THE SERVER FOR DIFFERENT LOSS PROBABILITIES (%) UNDER "UNSTABLE NETWORK" CONDITIONS. BANDWIDTH=1GBPS. TCP `INITCWND`=20.

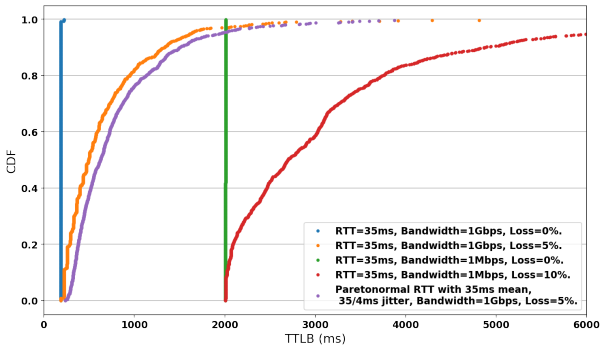


Fig. 14. TTLB CDF for PQ TLS 1.3 connections. 200KiB from the server. 35ms RTT. TCP `initcwnd`=20.

plotted the PQ TLS 1.3 connection (P256+MLKEM768, 16KB chain) for 200KiB from the server, 35ms RTT and different bandwidth and loss probabilities. Figure 14 shows that at 0% loss, fast 1Gbps or slower 1Mbps networks lead to pretty stable TTLB times. As we observed in Section IV-B2, the 1Mbps connections are slower due to low bandwidth. Note that under all types of unstable conditions (1Gbps and 5% loss, 1Mbps and 10% loss, `Paretonormal`-distributed network delay), the TTLB increases very early in the experimental measurement sample which demonstrates that the total connection times are highly unstable.

In summary, we saw that the impact of data-heavy handshakes on the total connection time is lower than on the handshake itself under lossy conditions or unstable network delays. The impact drops as the transferred data size increases, but adverse network conditions introduce high volatility on all connections. Also, notably, high network instability affects the TTLB of connections that transfer sizable amounts of data so much that the additional slowdown introduced by data-heavy handshakes does not make noticeable difference.

V. CONCLUSIONS AND FUTURE WORK

In conclusion, this work demonstrated that the practical impact of data-heavy, post-quantum algorithms on TLS 1.3 connections is lower than the impact on the handshake itself. Low-loss, low- or high-bandwidth connections will see little impact from post-quantum handshakes when transferring sizable amounts of data. The more data the connection transfers, the lower the impact. We also showed that under unstable conditions with higher loss rates or high-variability delays, although the impact of PQ handshakes could vary, it stays within certain limits and drops as the total transferred data increases. Additionally, we saw that unstable connections inherently provide poor completion times; a small latency increase due to post-quantum handshakes would not render them less usable than before. This does not mean that trimming down the handshake data is undesirable, especially if little application data is sent relative to the size of the handshake messages, but a few extra KB will not be noticeable in connections transferring hundreds of KB or more. Connections that transfer <10-20KB of data will probably be more impacted by the new data-heavy handshakes.

In the future, it would be worthwhile to run experiments with actual ML-DSA certificates and use actual TLS 1.3 traffic profiles observed in real-world networks. These could include mobile networks, Web traffic in different use-cases like mobile device browsing, Internet browsing, social media or cloud applications and more. Such experiments would give us a better idea about the TTLB of these applications being noticeable or not. It would also be worth testing real-world QUIC connection traffic profiles (which use TLS 1.3 underneath) under different network conditions to confirm if the post-quantum algorithm impact on their TTLB will be any different.

VI. ACKNOWLEDGEMENT

The authors would like to thank Eric Crocket and Steven Collison from AWS for their valuable feedback.

REFERENCES

[1] Akamai, "Akamai Online Retail Performance Report: Milliseconds Are Critical." [Online]. Available: https://www.akamai.com/newsroom/press-release/akamai-releases-spring-2017-state-of-online-retail-performance-report

[2] AWS, "s2n-tls." [Online]. Available: https://github.com/aws/s2n-tls

[3] C. Bas Westerbaan, "Sizing Up Post-Quantum Signatures," Nov. 2021. [Online]. Available: https://blog.cloudflare.com/sizing-up-post-quantum-signatures/

[4] D. Benjamin, D. O'Brien, and B. Westerbaan, "Merkle Tree Certificates for TLS," Internet Engineering Task Force, Internet-Draft draft-davidben-tls-merkle-tree-certs-01, Sep. 2023, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/draft-davidben-tls-merkle-tree-certs/01/

[5] K. Bürstinghaus-Steinbach, C. Krauß, R. Niederhagen, and M. Schneider, "Post-quantum tls on embedded systems: Integrating and evaluating kyber and sphincs+ with mbed tls," in *Proceedings of the 15th ACM Asia Conference on Computer and Communications Security*, ser. ASIA CCS '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 841–852. [Online]. Available: https://doi.org/10.1145/3320269.3384725

[6] W. Childs-Klein, "s2n-tls pq-tls-benchmark." [Online]. Available: https://github.com/WillChilds-Klein/pq-tls-benchmark/tree/fixed-cert

[7] Chromium, "Core Principles." [Online]. Available: https://www.chromium.org/developers/core-principles/

[8] J. Chu, N. Dukkipati, Y. Cheng, and M. Mathis, "Increasing TCP's Initial Window," RFC 6928, Apr. 2013. [Online]. Available: https://www.rfc-editor.org/info/rfc6928

[9] S. Dawaliby, A. Bradai, and Y. Pousset, "In depth performance evaluation of lte-m for m2m communications," in *2016 IEEE 12th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, 2016, pp. 1–8.

[10] R. Döring and M. Geitz, "Post-quantum cryptography in use: Empirical analysis of the tls handshake performance," in *NOMS 2022-2022 IEEE/IFIP Network Operations and Management Symposium*, 2022, pp. 1–5.

[11] Firefox, "Performance Sheriffing." [Online]. Available: https://firefox-source-docs.mozilla.org/testing/perfdocs/perf-sheriffing.html

[12] Google, "PageSpeed Insights ." [Online]. Available: https://developers.google.com/speed/docs/insights

[13] http archive, "Report: State of the Web," https://httparchive.org/reports/state-of-the-web.

[14] T. Iraklis, K. Limniotis, and N. Kolokotronis, "Evaluating the performance of post-quantum secure algorithms in the TLS protocol," *Journal of Surveillance, Security and Safety*, vol. 3, no. 3, pp. 101–27, 2022.

[15] D. Jackson, "Abridged Compression for WebPKI Certificates," Internet Engineering Task Force, Internet-Draft draft-ietf-tls-cert-abridge-00, Sep. 2023, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/draft-ietf-tls-cert-abridge/00/

[16] P. Kampanakis, C. Bytheway, B. Westerbaan, and M. Thomson, "Suppressing CA Certificates in TLS 1.3," Internet Engineering Task Force, Internet-Draft draft-kampanakis-tls-scas-latest-03, Jan. 2023, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/draft-kampanakis-tls-scas-latest/03/

[17] P. Kampanakis and M. Kallitsis, "Faster post-quantum TLS handshakes without intermediate CA certificates," in *Cyber Security, Cryptology, and Machine Learning*, S. Dolev, J. Katz, and A. Meisels, Eds. Cham: Springer International Publishing, 2022, pp. 337–355.

[18] P. Kampanakis and T. Lepoint, "Vision paper: Do we need to change some things?" in *Security Standardisation Research*, F. Günther and J. Hesse, Eds. Cham: Springer Nature Switzerland, 2023, pp. 78–102.

[19] L. V. Kris Kwiatkowski, "The TLS Post-Quantum Experiment," Oct. 2020. [Online]. Available: https://blog.cloudflare.com/the-tls-post-quantum-experiment/

[20] A. Langley, "CECPQ1 results," Nov. 2016. [Online]. Available: https://www.imperialviolet.org/2016/11/28/cecpq1.html

[21] ——, "CECPQ2," Dec. 2018. [Online]. Available: https://www.imperialviolet.org/2018/12/12/cecpq2.html

[22] D. Marchsreiter and J. Sepúlveda, "Hybrid post-quantum enhanced tls 1.3 on embedded devices," in *2022 25th Euromicro Conference on Digital System Design (DSD)*, 2022, pp. 905–912.

[23] NIST, "Migration to Post-Quantum Cryptography." [Online]. Available: https://www.nccoe.nist.gov/crypto-agility-considerations-migrating-post-quantum-cryptographic-algorithms

[24] ——, "Module-Lattice-Based Digital Signature Standard." [Online]. Available: https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.204.ipd.pdf

[25] ——, "Module-Lattice-based Key-Encapsulation Mechanism Standard." [Online]. Available: https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.203.ipd.pdf

[26] ——, "Stateless Hash-Based Digital Signature Standard." [Online]. Available: https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.205.ipd.pdf

[27] ——, "NIST PQ project," Feb. 2022. [Online]. Available: https://csrc.nist.gov/projects/post-quantum-cryptography

[28] NIST NCCOE PQC Migration Project, "NIST SPECIAL PUBLICATION 1800-38C Draft." [Online]. Available: https://www.nccoe.nist.gov/sites/default/files/2023-12/pqc-migration-nist-sp-1800-38c-preliminary-draft.pdf

[29] C. Paquin, D. Stebila, and G. Tamvada, "Benchmarking post-quantum cryptography in tls," in *Post-Quantum Cryptography*, J. Ding and J.-P. Tillich, Eds. Cham: Springer International Publishing, 2020, pp. 72–91.

[30] S. Paul, Y. Kuzovkova, N. Lahr, and R. Niederhagen, "Mixed certificate chains for the transition to post-quantum authentication in tls 1.3," in *Proceedings of the 2022 ACM on Asia Conference on Computer and Communications Security*, ser. ASIA CCS '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 727–740. [Online]. Available: https://doi.org/10.1145/3488932.3497755

[31] C. Planet, "Initcwnd settings of major CDN providers," Feb. 2017, https://www.cdnplanet.com/blog/initcwnd-settings-major-cdn-providers/.

[32] E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.3," RFC 8446, Aug. 2018. [Online]. Available: https://rfc-editor.org/rfc/rfc8446

[33] D. Sikeridis, S. Huntley, D. Ott, and M. Devetsikiotis, "Intermediate certificate suppression in post-quantum tls: An approximate membership querying approach," in *Proceedings of the 18th International Conference on Emerging Networking EXperiments and Technologies*, ser. CoNEXT '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 35–42. [Online]. Available: https://doi.org/10.1145/3555050.3569127

[34] D. Sikeridis, P. Kampanakis, and M. Devetsikiotis, "Assessing the overhead of post-quantum cryptography in TLS 1.3 and SSH," in *Proceedings of the 16th International Conference on Emerging Networking EXperiments and Technologies*, ser. CoNEXT '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 149â€"156.

[35] ——, "Post-quantum Authentication in TLS 1.3: A performance study," in *27th Annual Network and Distributed System Security Symposium, NDSS 2020, San Diego, California, USA, February 23-26, 2020*. The Internet Society, 2020. [Online]. Available: https://www.ndss-symposium.org/ndss-paper/post-quantum-authentication-in-tls-1-3-a-performance-study/

[36] M. Sosnowski, F. Wiedner, E. Hauser, L. Steger, D. Schoinianakis, S. Gallenmüller, and G. Carle, "The performance of post-quantum tls 1.3," in *Companion of the 19th International Conference on Emerging Networking EXperiments and Technologies*, ser. CoNEXT 2023. New York, NY, USA: Association for Computing Machinery, 2023, p. 19–27. [Online]. Available: https://doi.org/10.1145/3624354.3630585

[37] M. Thomson and S. Turner, "Using TLS to Secure QUIC," RFC 9001, May 2021. [Online]. Available: https://www.rfc-editor.org/info/rfc9001

[38] X. Xie, X. Zhang, and S. Zhu, "Accelerating Mobile Web Loading Using Cellular Link Information," in *Proceedings of the 15th Annual International Conference on Mobile Systems, Applications, and Services*, ser. MobiSys '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 427–439. [Online]. Available: https://doi.org/10.1145/3081333.3081367

[39] D. Xu, A. Zhou, X. Zhang, G. Wang, X. Liu, C. An, Y. Shi, L. Liu, and H. Ma, "Understanding Operational 5G: A First Measurement Study on Its Coverage, Performance and Energy Consumption," in *Proceedings of the Annual Conference of the ACM Special Interest Group on Data Communication on the Applications, Technologies, Architectures, and Protocols for Computer Communication*, ser. SIGCOMM '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 479–494. [Online]. Available: https://doi.org/10.1145/3387514.3405882