Hybrid Classical-Quantum Cryptography in a Post-Quantum World

Tyler Jeng
Georgia Institute of Technology
Atlanta, United States

Shruti Kakubal
Georgia Institute of Technology
Atlanta, United States

Zhuolin Li
Georgia Institute of Technology
Atlanta, United States

Delaney Gomen
Georgia Institute of Technology
Atlanta, United States

Abstract

The current Internet infrastructure is built on asymmetric cryptography. Transport Layer Security (TLS), a crucial protocol which enables secure data communication across the internet, heavily depends on public key algorithms [23]. The security of public key cryptography relies on certain mathematical problems that present-age computers find difficult to solve. However, sufficiently powerful quantum computers could easily break the hardness of such problems and render protocols like TLS trivial. Post Quantum Computing (PQC) [11] threatens to endanger the very notions of data confidentiality, authentication, and integrity that many secure applications are dependent on.

The National Institute of Standards and Technology (NIST) has theorized a few algorithms which should resist the overwhelming computing power of quantum devices [2]. However, there remains a large uncertainty for protocols using classical cryptography: how do we transition from our existing systems to PQC-safe ones?

As we have seen over the years, the industry is slow to adopt changes. Thus, it is important to maintain backward compatibility while preparing for future attacks. Hybrid schemes which offer the ability to use both traditional and PQC-based algorithms would be the most appropriate way to introduce updates without breaking the current system [21]. This idea has been applied to TLS [20], but there are many issues with its implementation and subsequent adoption. If applications do upgrade themselves to support hybridized TLS, there is still the issue of latency in the initial handshake due to larger key sizes and longer computations [20].

Research has been done to tune and optimize classical TLS such as [37]. To the best of our knowledge, our work is the first to explore if and how such optimizations could be applied to improve PQC-based TLS. In addition, numerous isolated incidents have highlighted the shortcomings of hybrid TLS deployments, so we have consolidated and analyzed these events for a unified perspective. Finally, while there has been prior work that measures the performance of PQC-based TLS, a comprehensive evaluation of its current usage, impediments and possible improvements remains wanting.

Thus, in this paper, we provide a complete review of hybrid TLS, followed by benchmarking the cryptographic operations of the traditional TLS key exchange (RSA-2048 + ECDHE) and post-quantum TLS key exchange (Kyber768 + ECDHE) to showcase its limitations. This includes measuring the performance of the PQC certificate algorithm Falcon too. Finally, we explore some mechanisms to optimize the performance of post-quantum TLS and improve the rate of adoption, especially in low-bandwidth IoT applications.

1 Introduction

1.1 Public Key Infrastructure (PKI)

PKI is the set of policies and procedures needed to manage security through public key encryption. The PKI binds public keys to the identity of a user or entity in the form of digital certificates. It consists of a hierarchy of Certificate Authorities (CA) to issue and revoke certificates, forming a chain of trust. The root CA at the top of the hierarchy acts as the trust anchor. The goal of any PKI is to provide confidentiality through encryption, integrity through MACs and authentication through digital signatures [36].

1.2 TLS

TLS is an application-layer protocol that provides end-to-end security for data transfers across internet applications [23]. TLS is also built on a PKI - it is the backbone of secure internet transactions across the world, so it is extremely important that it stays secure. It ensures that the browser does not connect to an impostor when it tries to access a website, and that our data is securely shared only with the application server.

TLS uses asymmetric encryption to perform the handshake, i.e. set up the connection between the client (browser) and the server. It has two crucial steps, which also happen to be the main points of vulnerability [43]:

(1) `Key exchange` – both sides will pick two large numbers, i.e. key shares and share them with each other (using either RSA or ECDHE). These numbers are used to derive symmetric keys used for encrypting the rest of the session data. They are also used to generate the MAC keys that are used for testing the integrity of the connection.
(2) `Authentication` – The server shares its certificate and public key with the client to confirm its identity. This is also done using RSA/ECDSA.

1.3 Impact of PQC Algorithms on PKI/TLS

PQC algorithms have different storage and resource requirements than classical cryptographic algorithms, since they use larger keys and heavier computations. Network scientists are still working on minimizing the time taken to set up TLS connections and the amount of data transferred. Quantum cryptography will only challenge this further.

As mentioned earlier, TLS (or any PKI for that matter) uses public-private key pairs for both encryption and authentication. These keys are picked based on the hardness of factorization problem in case of RSA or the discrete logarithm problem in case of DH. Consider RSA as an example. A quantum computer could be used to factorize the public key and consequently figure out the private key associated with it – the computer would just test all numbers at once [54]. The attacker could then break both confidentiality and integrity of the scheme. Thus, PKI must be made quantum-resistant.

Considering its wide usage and the time it would take to update all the applications that depend on PKI, hybrid schemes which combine existing traditional algorithms with new quantum cryptographic methods are being introduced [9]. The amalgamation of algorithms aims to ensure that even if one is broken, the other will maintain its security. The traditional and post-quantum key exchange and authentication can be performed in parallel, with the results combined to protect against both classical and quantum-equipped attackers.

1.4 NIST Standard Quantum-Safe Algorithms:

NIST selected 4 post-quantum algorithms for standardization - CRYSTALS-Kyber, CRYSTALS-Dilithium, Sphincs+ and FALCON [2]. They have been renamed as follows:

(1) `Federal Information Processing Standard (FIPS) 203 [47]` - This has been chosen as the primary quantum-safe encryption standard. It uses a smaller encryption key than other PQC algorithms and is faster as well. It is based on CRYSTALS-Kyber (renamed Module Lattice-Based Key Encapsulation Mechanism or ML-KEM). This protocol has been adopted for PQC-based key exchange in hybrid TLS [15].
(2) `FIPS 204` - This is the primary digital signature standard, based on CRYSTALS-Dilithium [44]. (Renamed Module Lattice-Based Digital Signature Algorithm or MLDSA). This scheme is being used for PQC authentication in hybrid TLS schemes.
(3) `FIPS 205` - Digital signature standard chosen as a backup for ML-DSA. It is based on Sphincs+, also known as SLH-DSA (Stateless HASH) [45].
(4) `FIPS 206` - This standard is yet to be released. It will be based on FALCON, also known as FN-DSA (FFT over NTRU-Lattice Based Digital Signature Algorithm) [27].

2 Background and Prior Work

In this section, we review the best practices for testing performance in public key cryptography that we use in our study, give a summary of how quantum resistant applications have been tested and implemented over the last 20 years, and survey recent algorithms and their performance metrics.

2.1 Testing Performance in Cryptography

Our own metrics and tests follow from a growing foundation of literature that use best practices of testing efficiency and performance in cryptography and adapt it to testing post-quantum cryptographic standards. to test the efficiency and performance of post-quantum cryptographic standards. Not surprisingly, many tests for post-quantum TLS standards follow the same framework as for testing traditional TLS standards. An example of one of these traditional tests originated back in 2006, where Coarfa et al. simulated two different web networks to test TLS efficiency [17]. Their metric was throughput (connections per second) and they used CPU power as a dependent variable that they varied to further test performance.

This basic testing structure has survived as we begin to test protocols for the post-quantum era. Like Coarfa et al., in 2022 Bernstein et al. also controlled their computing resources to measure macro benchmarks in a smaller testing environment[10]. They created their own tool called `tls_timer` for measurement and tested against several different protocols, like we will do in our analysis. In 2024, Collins et al. used the open-source library `liboqs` to build quantum-resistant TLS protocols to test [1, 18]. They were primarily interested in speed and measured speed by seeing how many CPU cycles each protocol would take. While the concern about quantum-resistant methods are their run times and efficiency, these authors mentioned that lattice-based schemes could still be faster than traditional methods.

2.2 Creating Quantum-Resistant Applications

Since NIST started working towards a postquantum standard, various open-source libraries, researchers, and organizations have created their own implementations using various postquantum methods such as ML-KEM (Crystals-Kyber) and ML-DSA (Crystals-Dilithium), which are projected to become the standard for secure communications by 2030 [6]. While there are public efforts to encourage the development of quantum-resistant applications, like the open source `liboqs` library mentioned in the previous section, the commercial sector has also caught on. Amazon's AWS-LC has become the first FIPS 140-3 compliant library supporting ML-KEM [4].

Behind these efforts, academic research has played an essential role in validating (and creating) these protocols. Before NIST's Post-Quantum-Cryptography initiative had even commenced, foundational work demonstrated the feasibility of lattice-based cryptography by focusing on performance and resistance to side-channel attacks [3, 14]. Proposals have been made on how to integrate postquantum cryptography into TLS by eliminating handshakes [53] and through other experimental approaches [38]. Many hybrid solutions have been attempted by combining ML-KEM, ML-DSA, and traditional protocols [13, 28, 49]. Academics have tried to guarantee that post-quantum solutions are secure and scalable, and testing of these applications will be covered in the next section.

There is also emphasis on creating postquantum protocols that can be easily adopted or integrated into legacy systems [42]. Bernstein et al. pointed out in 2022 that none of the initial proposals received by NIST to replace traditional asymmetric cryptography in a post-quantum era used a key exchange protocol akin to Diffie-Hellman key exchange, raising concerns about future compatability [10]. Others have mentioned the need for policy alignment, interoperability, and industry-specific needs that must be met to deploy post-quantum solutions [18, 49].

2.3 Testing Quantum Resistant Algorithms

Quantum-resistant public key cryptography has come a long way since Pernler and Cooper published in 2009 a survery of current algorithms believed to be resistant to quantum computing attacks [51]. The literature has moved from algorithms being mostly theoretical with limited implementation or performance benchmarks to stress-testing them in diverse computing environments and by trying to simulate them within real-world constraints. Recent studies focus on evaluating postquantum schemes for efficiency, reliability, and interoperability, and much of our cited prior work was published within the last five years.

Berstein et al. proposed "McTiny," focusing on deploying post-quantum protocols in resource-constrained environments [12]. Bos et al. focused on tweaking assumptions made in lattice-based key creation to try and improve efficiency of these calculations [13]. Barton et al. tested various NIST finalists and compared performance across platforms [8]. Various ecosystems and schemes made available by the 'liboqs' library were tested by Garrach et al. [30] Mandev and Kavun expanded post-quantum signature algorithms like ML-DSA into an Android email application plug-in, creating one of the few tests of real-world post-quantum applications in the literature [41]. A growing theme in network security research is that usability and performance are just as critical to overall success of a given protocol than are theoretical and algorithmic viability.

Other tests of `liboqs` protocols in hybrid settings include An et al., focusing broadly on key exchange protocols that use lattice-based structures [5]. Fitzgibbon and Ottaviani targeted constrained devices specifically to test the NIST's post-quantum cryptography proposals [25]. For further hardware acceleration and efficiency, Dziechiarz et al. and Oplika et al. both in 2024 zeroed in on improving digital signature technology like ML-DSA, further demonstrating the rise in popularity of testing these algorithms based on their adaptability and real-world potential [24, 48].

3 Approach

Our approach is focused on the structured analysis of quantum-vulnerable aspects of the TLS1.3 protocol as well as an examination into the state of current PQC algorithms, supported by benchmarking metrics and visualizations. This paper will contain two main sections of analysis in addition to a final proposed hybrid model: we first address the issue of post-quantum by evaluating the vulnerable aspects of the TLS 1.3 protocol and proposing secure modifications, then we benchmark real-world quantum-hybrid implementations such as Amazon AWS Key Management System (KMS) and browser-based hybrid-scheme TLS deployments. We conclude with a thorough investigation into the feasibility of our changes, focusing on the tradeoffs for performance/speed, security, and ease of deployment/overhead.

3.1 Systematic evaluation of the TLS1.3 protocol

We begin by examining the general structure of TLS1.3 and identifying its quantum-vulnerable features. RSA (or ECDH) encryption is a key player in TLS's structure which allows for the safe communication of certificates as well as the establishment of the shared secret from which forward secrecy relies upon [62]. It relies on prime computations which are difficult for traditional computers to dismantle without knowing the proper key. However, these currently 'safe' algorithms would be trivially defeated using quantum computers, namely through Shors algorithm [16][33]. To ensure sensitive data and communications are secured in the face of this threat, we seek solutions which remain secure despite the developing state of quantum computing. Some of these proposed changes include minor adjustments to qualities like key length. For instance, though the TLS handshake is often followed by secure symmetric encryption through an algorithm such as AES, minor adjustments to this protocol such as lengthening the block length to 512 or 1024 would render it secure in the face of quantum computing [26]. For other, more exploitable aspects of the TLS handshake, we offer a combination of groundbreaking algorithms which both modify and reinforce the safety of the system. As leaders in safe and secure technological development, NIST's proposed algorithms are the main contenders for such cryptographic functions which are resistant to the computational prowess of quantum computers [19]. However, as with all security protocol adjustments, we must ensure that the transition to this proposed infrastructure does not inequitably incur technical debt through significant overhead nor hamper the user experience through sluggish operation times.

3.2 Benchmarking and comparative analysis

Through thorough benchmarking, we posit that our suggested modernistic framework maintains a delicate balance between speed, security, and usability.

We utilize the pyca 'cryptography' library to establish a control for our benchmarks; in addition to other widely used cryptographic algorithms, pyca implements openSSL's TLS1.3 structure, allowing us to capture data on the most widely used TLS implementations in modern times [61]. Since we are primarily examining the effects of different design choices on performance, safety, and simplicity, we elect Elliptic Curve Diffie Hellman (ECDH) as our asymmetric key encryption method [60]. Its shorter key sizes and faster overall speed are paramount attributes for properly assessing our new quantum algorithms' performance despite its greater complexity. As our baseline, ECDH will be measured for its handshake, key generation, verification, and signing speeds [52].

For evaluating the performance of our proposed PQC algorithms, we use the liboqs python library, which provides a comprehensive suite of cutting-edge quantum-safe algorithms [57]. Our benchmarking focuses on ML-DSA, ML-KEM, and Falcon which will seamlessly replace aspects of the TLS1.3 protocol's certificate authentication and key exchange mechanisms. For ML-DSA and Falcon, we measure the key generation, signing, and signature verification speeds, whereas ML-KEM will be measured through its encapsulation and decapsulation efficiency [57]. Each set of assessments will be compared to their quantum-vulnerable counterpart, visualized through graphs produced by the matplotlib library [32]. These diagrams illustrate the differences in performance for each of our tested algorithms in a digestible manner. To run our scripts, we recommend an operating system such as linux or macOS, C99-compliant compiler such as gcc, and an instance of python version 3.7 or higher; our github, located at https://github.com/zhuolin-tech,

further details the installation and setup of our project. Our thorough documentation ensures transparency and reproducible results which can be verified by third parties.

3.3 Limitations and Ethical Considerations

For our evaluation and benchmarking, we rely on open-source tools such as the liboqs and pyca libraries. This ensures unambiguous reproducibility and encourages scrutiny and aligns with open-source licensing policies. However, it is critical to note that some businesses may be developing proprietary solutions which further optimize the TLS infrastructure, and, due to intellectual property limitations or otherwise, we are unable to obtain these developing ideas. As a result, we posit that our tests are performed using the most optimal, up-to-date implementations available to the public rather than a full exploration into the development of PQC algorithms.

Additionally, we must approach all frameworks, ours included, with a level of skepticism, as even well-maintained and updated libraries can contain unseen vulnerabilities or assumptions which aren't fully visible to the average user. Quantum computing as a research field remains quite young; as quantum computing is further developed, we may discover new exploits which compromise the safety of both classical and supposed quantum-safe systems. This uncertainty reinforces the need for continuous testing and development in this field to validate and improve PQC frameworks over time.

Responsible deployment of our framework should also be handled with caution, and we propose that these updated standards be publicized by centralized agencies such as the US government's NIST and further enforced by government agencies. These organizations have adequate tools, infrastructure, and enforcement power which allows them to roll out cryptographic standards at scale while remaining equitable to different industries and communities. Without this kind of oversight, we risk misconfigurations and incompatible and unusable software which threaten even greater security breaches.

Finally, we note that there may be mild variations in statistical data from our benchmark results which are caused by hardware variations. Since cryptographic operations are heavily influenced by processor speed, memory bandwidth, and architecture, our results may not be exactly reproducible. Despite this, we run our environment from an accessible setup which can be emulated reliably, and we hold that our trends still provide a good estimate of performance and insight into the tradeoffs for each design choice. Our tests were run using a MacBook Pro's M1 Pro chip to measure cycles.

4 Benchmarking Results

Running our scripts, we obtain statistics for the RSA, ECC, ML-KEM, ML-DSA, and Falcon algorithms which highlight the tradeoffs for implementing PQC functions. We first examine the speeds of each method, as performance is our primary concern. Examining Figure 1, we see that RSA broadly has the highest cost, requiring significantly more time for key generation and signing, as indicated by the keygen and sign operations. ECC holds similar statistics, albeit with lower keygen speeds compared to RSA. Both algorithms, with

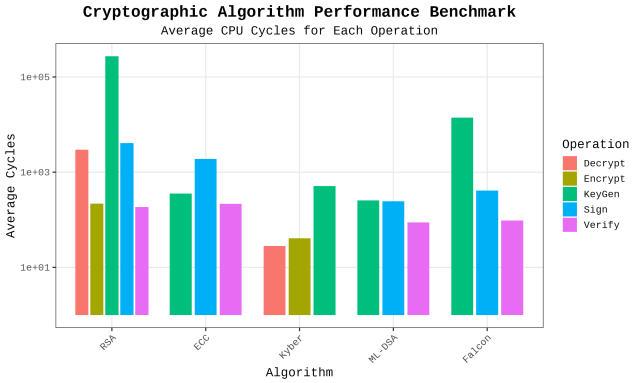

Figure 1: Graph highlighting the relevant operation speeds of the RSA, ECC, ML-KEM (Kyber), ML-DSA, and Falcon algorithms, measured in processor cycles.

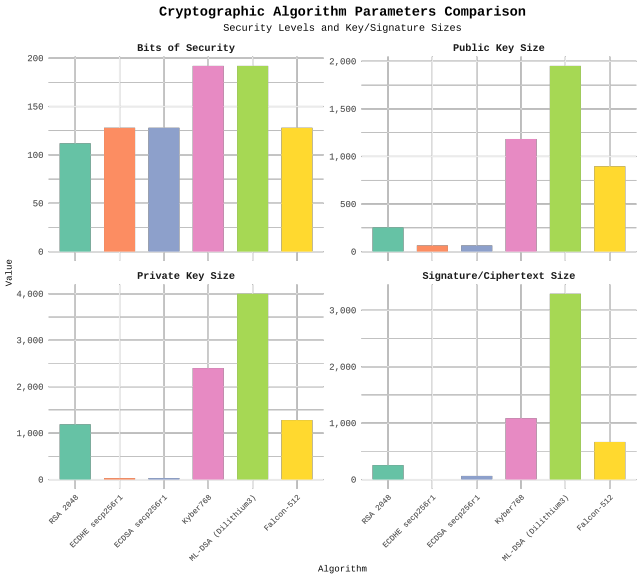

Figure 2: Graphs investigating the security bit, signature/ciphertext size, and public and private key size parameters of RSA, ECC, Kyber, ML-DSA, and Falcon

their average keygen times of 273478 and 355 cycles respectively, are largely inefficient even amongst the proposed PQC algorithms. ML-DSA boasts incredible performance, having the lowest execution time for keygen, sign, and verify operations at 256, 244, and 88 cycles, respectively. Though Falcon remains competitive with ML-DSA in terms of both signing and verification speeds at 411 and 96 cycles respectively, its large keygen speed at 13971 cycles is a detriment. On the other hand, ML-KEM (Kyber) proves to be a strong candidate for replacing RSA by performing encryption, decryption, and key generation operations multiple magnitudes faster than RSA. Its keygen, encrypt, and decrypt operations average to 509, 40, and 28 cycles whereas RSA lags at 273478, 219, and 2977 cycles for the same actions. However, by combining these performance details with context from the Figure 2, which describes the

parameters of each algorithm including the key and signature sizes, we understand that ML-DSA is not a winner in all categories since it necessitates a significant amount of overhead through its large key and certificate sizes. Additionally, we suggest that Falcon be reserved for situations in which lighter overhead is important since its key and signature sizes are significantly smaller than ML-DSA's. Ultimately, we still assert that ML-DSA and ML-KEM are the best candidates for our proposed quantum-safe hybrid model.

5 Hybrid TLS Key Exchange Optimizations

Some schemes, like [29], integrate quantum key distribution into hybrid TLS itself, but such improvements in security come at a very high cost in terms of speed and bandwidth. QKD slows down the already slower PQC hybrid TLS by 68%.

Although ML-KEM has been found to be faster than ECDHE [59] since it uses polynomial arithmetic, its key size is much larger than that of the latter, which is an issue in memory-constrained IoT devices. Some optimizations have been made for the PQC algorithm code at the assembly level, such as [59] and hardware accelerations such as [55], but this would obviously be difficult to implement as a software engineer or network operator.

In addition, hybrid TLS is still slower than classical TLS because of the signing operation of Dilithium/Falcon [59]. In fact, PQ-based authentication is the biggest bottleneck of hybrid TLS, since authentication happens on both the server and client side. In addition, since signatures require more memory, certificate chains become larger and consequently, TLS authentication messages become bulkier.

Prior research into classical TLS has given rise to optimization techniques like handshake latency reduction [40] and reusing sessions [7]. We believe that leveraging such methods in the context of PQC-hybrid TLS could yield similar performance improvements and mitigate the impact of slower connection setup and other issues introduced by PQC algorithms. While suggesting these optimizations, we kept two crucial issues in mind:

- The optimization should ensure that no other interdependent protocols are affected, since network protocols are layered.
- TLS requires fixed-length messages, so the client and server implementations would have to be adjusted for the extra data. Failing to do so could lead to parsing errors, message fragmentation or other vulnerabilities – issues that would only further increase the handshake time.

5.1 Optimization 1 – Preferential TLS

In TLS 1.3, the client picks a cipher suite for encryption and authentication that it thinks the server supports and sends the key material in the ClientHello message itself. It has been established that post-quantum based KEMs like ML-KEM have performance comparable to traditional key exchange algorithms, and PQC-based signing is usually the bottleneck [34]. The HTTP client's list of cipher suites should be explicitly ordered to ensure accurate and sufficiently secure connection setup. We recommend the following modes in descending order of priority:

(1) Hybrid Mode – This is the optimal scenario, wherein PQC-based key agreement (like ML-KEM) and PQC-based digital signatures (like Dilithium) are combined with their classical counterparts like ECDHE and ECDSA for the best possible defense.
(2) Partial Hybrid Mode – This should be used in low-bandwidth environments which are more prone to packet loss and/or fragmentation, like IoT-based applications. Here, the PQC-based key agreement like ML-KEM is combined with a traditional signature algorithm like ECDSA, eliminating the latency caused by PQC-based signing. Even if a quantum-enabled attacker could break the signature scheme, the actual keys used for the transaction are still protected. Ideally, this mode would not be used in highly sensitive applications.
(3) Fallback/Legacy Mode – this is the mode that applications would resort to when the client and/or server do not support PQC schemes. Classical cryptography would be used for both key exchange and authentication.

5.2 Optimization 2 – Finetune TLS Settings

(1) Connection Timeout
This value should be increased to allow for longer handshake time, but we have to ensure that it is not too high, causing idle connections to stay open forever.
(2) Max Idle Time
This value should be increased when the network has bursts of traffic. Connections can be reused more, decreasing request latency as well.
(3) Connection Pooling - Here, we maintain a predefined number of established connections in a pool. When an application requests a connection to the server, it uses one of the connections in the pool rather than creating a new one. Ideally, we could maintain more PQC-based connections (detected using the cipher suite) to increase the adoption. This will also limit resource usage, which is important since PQC-based algorithms require more computational resources.
(4) TLS Record Size
The TLS record protocol fragments application data into records that encapsulate the encrypted data, headers and MAC before it is transmitted over TCP. When the TLS record size is large, the associated TCP buffer also increases in size. Larger records mean more TCP packets are created, which necessitates more reassembly by the transport layer before data is passed to the TLS layer. This increases latency and the probability of packet loss – a critical issue with PQC algorithms that have large key and ciphertext sizes. Studies show that packet loss rates above 3-5% significantly impact the performance of post-quantum algorithms that span across many packets [50].
We can control the impact of packet loss by adjusting the congestion window (CWND), i.e. the number of packets that can be sent before an acknowledgement has to be received. The CWND increases dynamically in the absence of packet loss, so the initial value is set quite low. Ideally, TLS handshake records should remain within the initial CWND as exceeding this threshold would force the server

to wait for an acknowledgment, which adds an additional round-trip time (RTT) for the connection setup [46].
This effect becomes more evident in PQC-based TLS where post-quantum keys and certificate chains grow larger. For example, older browsers provided an initial CWND of only four TCP segments, definitely insufficient for a full certificate chain. While modern systems typically have an initial CWND of ten segments, this can still be exceeded when using large PQC keys and certificates [46].
Our suggested optimization is as follows: the server should adjust the congestion window during connection establishment if a quantum-safe cipher suite is chosen. This allows the server to account for larger post-quantum keys and certificates, avoiding unnecessary fragmentation and additional RTTs. This approach is also consistent with the official TLS hybrid design goals published by the IETF, which emphasize minimizing handshake latency in hybrid post-quantum TLS deployments [56].

5.3 Optimization 3 – TLS Session Resumption

This is a mechanism wherein the client and server can bypass key exchange [7]. They use a shared secret which was previously agreed upon or derive a new one. In the context of hybrid TLS, the prior and new connections both should use PQC, as it might be counterproductive if the previously used secret was not quantum safe in the first place. In addition, the master secret which new keys are derived from could be QKD based for higher levels of security, but this would strain the bandwidth of the connection.

5.4 Optimization 4 – TLS False Start

TLS False Start allows the client to start sending application data before the handshake completes [39]. Specifically, instead of waiting for the server to send its "Handshake Finished" message, the client can start data transmission once it sends its own "Cipher-Suite-to-be-Used" and "Finished" messages. The handshake is retroactively validated, i.e. the cryptographic hash of all prior handshake messages is verified later [7].

This optimization reduces handshake latency by one full RTT, but it introduces a small window in which the attacker may attempt to tamper with handshake messages or influence the negotiation of a weaker cipher suite, giving rise to potential downgrade attacks. However, the attack surface is restricted to very few actions and the attacker cannot derive any useful information without access to the full handshake state [39].

We propose restricting the use of False Start only to connections that use PQC-safe cipher suites. If the handshake fails to validate or the connection resets, the cipher suite is downgraded to one of the traditional options and a full handshake is performed.

Although TLS False Start was standardized by IETF [39], it is not widely used in browsers because of the complexity of coordinating early data transmission with late handshakes and the associated risks. However, the growing popularity and adoption of quantum-safe TLS could make False Start with stricter cipher suite requirements a potentially promising avenue for optimization.

5.5 Optimization 5 – OPTLS [37]

0 – RTT [31]

TLS 1.3 offers a couple of improvements over TLS 1.2, the most impactful being the reduction in connection setup latency from 3 RTT to 2 RTT. However, the time required to resume a connection is still the same as it was in TLS 1.2. Optimized TLS (OPTLS) [37] introduces a novel optimization known as 0-RTT, which speeds up resumed connections by 1 RTT. As of March 2017, Chrome and Firefox did not support 0-RTT [58]. Given the pressing need to adopt PQC based protocols, current support should be reevaluated.

Browsers require perfect forward secrecy (PFS) to enable optimizations that reduce latency [58], including 0-RTT. In OPTLS, the client and server choose some random values x and y and compute the shared key $g^x y$. The session key is derived from $g^x y$ and $g^x s$, where g^s is the static secret key of the server. $g^x s$ provides security as long as s is secure, and $g^x y$ provides security if s is compromised, ensuring PFS.

In the 0-RTT mode, the client can send data in the very first message to the server, after sharing its public key, g^x. This message is encrypted using a key derived from $g^x s$, thus dependent on the server's static key which was previously shared. A common use case would be a search engine query, where the browser and server would have to complete the TLS handshake before the query can be sent to the server. In the 0-RTT mode, data transmission can start immediately if the server's key shared in a previous session is already cached in the browser. This optimization would prove to be very useful for PQC keys and ciphertext, which tend to be quite large.

However, the data sent in the first message encrypted by $g^x s$ is vulnerable to replay attacks, since it is sent before authentication. The server should detect and reject 0-RTT messages it has already seen. This mechanism is used in QUIC [22]. Alternatively, service providers like Cloudflare and Akamai restrict the scope of data in the first message to GET requests with no query parameters, since these are unlikely to be used for sensitive actions [58]. In addition, web applications using 0-RTT are expected to be replay-safe.

It is important to distinguish 0-RTT from TLS False Start. The latter allows the client to send application data before the handshake completes but still requires the "Finished" message of the client to be sent. However, 0-RTT focuses on sending data from the first message itself, leveraging a key derived from a prior session. It is easy to see that 0-RTT is more secure than False Start.

Offline Signatures

In addition to key exchange, OPTLS offers another optimization with respect to authentication [37]. It is more flexible than TLS1.3 in terms of digital signatures. OPTLS has variants where server signatures can be done offline, or in some cases not at all. Moreover, public keys can be cached for use in future sessions. This is especially beneficial for systems using PQC-based signature algorithms like Dilithium and Falcon which have much larger digital signatures compared to that of classical signature algorithms. This optimization could potentially save a substantial amount of bandwidth, consequently enhancing the usability of PQC-based signature schemes in resource-constrained IoT environments.

5.6 Optimization 6 - Suppress Intermediate CA (ICA) certificates

PQ-based authentication results in significantly larger certificate chains and keys, potentially causing higher packet loss rates and more RTTs. A study of public servers by Amazon [35] found that most internet servers use two ICAs maximum, so applications could just cache these certificates. This would allow clients to request ICA suppression when they already have the certificates. This method is backward compatible and does not compromise security since the root CA's public key is already pre-installed in the system and is required to validate the whole chain anyway.

6 Discussion

Benchmarking results also show that the postquantum algorithms (e.g., ML-KEM and ML-DSA) sometimes have more computationally expensive computational overhead than what was used traditionally. High-load environments: Need to consider latency during keygen, encapsulation, and signature verification. Optimisation techniques such as hardware acceleration and parallel processing are the primary way to combat these delays, successfully and massively.

And the deployment challenges go way beyond tech. Bridging the gap to be able to deploy hybrid encryption on top of existing TLS infrastructures involves interoperability, regulations and standards. Guidance is required to facilitate the effective transition from classical PKI systems to industry and standards body/ government stakeholder coordinated collaboration.

Investigations of Algorithmic and Key management should be pursued to improve performance in the future. More system transparency and accountability: Combining advanced audit mechanisms (such as blockchain monitoring) in future investigations. Those are important steps to help break through the current barriers and prepare for this upcoming post-quantum era takeover of hybrid encryption.

7 Conclusion

In this study, we assessed quantum vulnerabilities of TLS 1.3 and presented a hybrid encryption method using classical cryptosystems together with post-quantum algorithms. TLS 1.3 was taken to the max and analyzed architecturally, with all relevant post-quantum algorithms: ML-KEM for key encapsulation and ML-DSA for digital signatures were also evaluated as benchmarks. In our empirical results, this hybrid approach is found to be an efficient counter to quantum-induced threats (the parallel execution of both classical and post-quantum processes). We intend that this dual-layered security architecture improves the secrecy and privacy of data, its integrity, as well as paves the path for end-to-end post-quantum computation. Experimental results demonstrate that in controlled lab conditions, the model not only performs competitively but also provides a significant security increase. The results offer important groundwork for future work on fine-tuning the hybrid model for practical use cases.

Nevertheless, there are still many hurdles that need to be crossed before the hybrid encryption model will break into full mainstream adoption. Increased computational overhead in post-quantum algorithms, especially during key generation and signature verification, will hamper system efficiency within high-demand environments. Moving from traditional public-key infrastructure to a hybrid is a bunch of complexities: interoperability issues, regional use case regulation relevance, and international standards. Future work should investigate algorithm performance tuning, hardware acceleration usage, and a wide-ranging security audit framework for auditing logs. To summarize, next steps: Industry and academia need to develop holistic guidelines in collaboration with regulators for the end-to-end transition to a quantum-safe world. Our results, therefore, illustrate the potential of this model for deployment in the future. In conclusion, the paper sets up a framework for the study of possible amplifications to do so. While we did not put them in our scope to implement, good experiments will improve their efficiency and be used in practical use cases in the real world.

Acknowledgments

We thank the developers of the open-source projects we utilized for metrics, and thanks to the professor and TAs of CS6262 for the gracious opportunity to work on this project.

References

[1] 2025. Open-Quantum-Safe/Liboqs. Open Quantum Safe.
[2] Gorjan Alagic, David Cooper, Quynh Dang, Thinh Dang, John M. Kelsey, Jacob Lichtinger, Yi-Kai Liu, Carl A. Miller, Dustin Moody, Rene Peralta, Ray Perlner, Angela Robinson, Daniel Smith-Tone, and Daniel Apon. 2022. Status Report on the Third Round of the NIST Post-Quantum Cryptography Standardization Process. doi:10.6028/NIST.IR.8413
[3] Erdem Alkim, Léo Ducas, Thomas Pöppelmann, and Peter Schwabe. 2016. Post-Quantum Key {Exchange—A} New Hope. In *25th USENIX Security Symposium (USENIX Security 16)*. 327–343.
[4] Amazon. 2023. AWS-LC Is Now FIPS 140-3 Certified | AWS Security Blog. https://aws.amazon.com/blogs/security/aws-lc-is-now-fips-140-3-certified/.
[5] Hyeongcheol An, Rakyong Choi, Jeeun Lee, and Kwangjo Kim. [n. d.]. Performance Evaluation of Liboqs in Open Quantum Safe Project (Part I). ([n. d.]).
[6] National Institute of Standards and Technology. 2024. *Module-Lattice-Based Key-Encapsulation Mechanism Standard.* Technical Report Federal Information Processing Standard (FIPS) 203. U.S. Department of Commerce. doi:10.6028/NIST.FIPS.203
[7] Nimrod Aviram, Kai Gellert, and Tibor Jager. 2021. Session resumption protocols and efficient forward security for TLS 1.3 0-RTT. *Journal of Cryptology* 34, 3 (2021), 20.
[8] Jon Barton, William J. Buchanan, Nikolaos Pitropakis, Sarwar Sayeed, and Will Abramson. 2022. Performance Analysis of TLS for Quantum Robust Cryptography on a Constrained Device. doi:10.48550/arXiv.1912.12257 arXiv:1912.12257 [cs]
[9] Fruzsina Bene and Attila Kiss. 2023. Public Key Infrastructure in the Post-Quantum Era. In *2023 IEEE 17th International Symposium on Applied Computational Intelligence and Informatics (SACI)*. IEEE, 000077–000082.
[10] Daniel J. Bernstein, Billy Bob Brumley, Ming-Shing Chen, and Nicola Tuveri. 2022. {OpenSSLNTRU}: Faster Post-Quantum {TLS} Key Exchange. In *31st USENIX Security Symposium (USENIX Security 22)*. 845–862.
[11] Daniel J. Bernstein and Tanja Lange. 2017. Post-quantum cryptography. *Nature* 549, 7671 (2017), 188–194. doi:10.1038/nature23461
[12] Daniel J. Bernstein and Tanja Lange. 2020. {McTiny}: Fast {High-Confidence} {Post-Quantum} Key Erasure for Tiny Network Servers. In *29th USENIX Security Symposium (USENIX Security 20)*. 1731–1748.
[13] Joppe Bos, Craig Costello, Léo Ducas, Ilya Mironov, Michael Naehrig, Valeria Nikolaenko, Ananth Raghunathan, and Douglas Stebila. 2016. Frodo: Take off the Ring! Practical, Quantum-Secure Key Exchange from LWE.
[14] Joppe W. Bos, Craig Costello, Michael Naehrig, and Douglas Stebila. 2014. Post-Quantum Key Exchange for the TLS Protocol from the Ring Learning with Errors Problem.
[15] Chromium Blog. 2023. Protecting Chrome Traffic with Hybrid Kyber KEM. https://blog.chromium.org/2023/08/protecting-chrome-traffic-with-hybrid.html. Accessed: April 20, 2025.
[16] Classiq. 2022. Quantum Cryptography – Shor's Algorithm Explained. https://www.classiq.io/quantum-cryptography-shors-algorithm-explained. Accessed: April 20, 2025.
[17] Cristian Coarfa, Peter Druschel, and Dan S. Wallach. 2006. Performance Analysis of TLS Web Servers. *ACM Trans. Comput. Syst.* 24, 1 (Feb. 2006), 39–69. doi:10.1145/1124153.1124155

[18] Daniel Collins, Loïs Huguenin-Dumittan, Ngoc Khanh Nguyen, Nicolas Rolin, and Serge Vaudenay. 2024. {K-Waay}: Fast and Deniable {Post-Quantum} {X3DH} without Ring Signatures. In *33rd USENIX Security Symposium (USENIX Security 24)*. 433–450.

[19] Computer Security Resource Center. 2017. Post-Quantum Cryptography - Selected Algorithms. https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms. Accessed: April 20, 2025.

[20] Eric Crockett, Christian Paquin, and Douglas Stebila. 2019. Prototyping post-quantum and hybrid key exchange and authentication in TLS and SSH. Cryptology ePrint Archive, Paper 2019/858. https://eprint.iacr.org/2019/858

[21] Flo D, Michael P, and Britta Hale. 2025. *Terminology for Post-Quantum Traditional Hybrid Schemes.* Internet-Draft draft-ietf-pquip-pqt-hybrid-terminology-06. Internet Engineering Task Force. https://datatracker.ietf.org/doc/draft-ietf-pquip-pqt-hybrid-terminology/06/ Work in Progress.

[22] Fynn Dallmeier, Jan P Drees, Kai Gellert, Tobias Handirk, Tibor Jager, Jonas Klauke, Simon Nachtigall, Timo Renzelmann, and Rudi Wolf. 2020. Forward-secure 0-RTT goes live: implementation and performance analysis in QUIC. In *Cryptology and Network Security: 19th International Conference, CANS 2020, Vienna, Austria, December 14–16, 2020, Proceedings 19*. Springer, 211–231.

[23] Tim Dierks and Christopher Allen. 1999. Rfc2246: The TLS protocol version 1.0.

[24] Dominik Dziechciarz and Marcin Niemiec. 2024. Efficiency Analysis of NIST-Standardized Post-Quantum Cryptographic Algorithms for Digital Signatures in Various Environments. *Electronics* 14, 1 (Dec. 2024), 70. doi:10.3390/electronics14010070

[25] Gregory Fitzgibbon and Carlo Ottaviani. 2024. Constrained Device Performance Benchmarking with the Implementation of Post-Quantum Cryptography. *Cryptography* 8, 2 (June 2024), 21. doi:10.3390/cryptography8020021

[26] FMTAD. 2024. Quantum Computing Encryption Threats: Why RSA and AES-256 Remain Secure. https://www.freemindtronic.com/quantum-computing-encryption-threats-why-rsa-and-aes-256-remain-secure. Accessed: April 20, 2025.

[27] Pierre-Alain Fouque, Jeffrey Hoffstein, Paul Kirchner, Vadim Lyubashevsky, Thomas Pornin, Thomas Prest, Thomas Ricosset, Gregor Seiler, William Whyte, Zhenfei Zhang, et al. 2018. Falcon: Fast-Fourier lattice-based compact signatures over NTRU. *Submission to the NIST's post-quantum cryptography standardization process* 36, 5 (2018), 1–75.

[28] Carlos Rubio Garcia, Abraham Cano Aguilera, Juan Jose Vegas Olmos, Idelfonso Tafur Monroy, and Simon Rommel. 2023. Quantum-Resistant TLS 1.3: A Hybrid Solution Combining Classical, Quantum and Post-Quantum Cryptography. In *2023 IEEE 28th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD)*. 246–251. doi:10.1109/CAMAD59638.2023.10478407

[29] Carlos Rubio Garcia, Abraham Cano Aguilera, Juan Jose Vegas Olmos, Idelfonso Tafur Monroy, and Simon Rommel. 2023. Quantum-resistant TLS 1.3: A hybrid solution combining classical, quantum and post-quantum cryptography. In *2023 IEEE 28th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD)*. IEEE, 246–251.

[30] Mohamed Amine Garrach, Chetan Waghela, Mahima Mary Mathews, and L S Sreekuttan. 2022. Benchmarking Speed of Post-Quantum Lattice Based PKE/KEM Schemes Using Liboqs. In *2022 International Conference on Trends in Quantum Computing and Emerging Business Technologies (TQCEBT)*. 1–5. doi:10.1109/TQCEBT54229.2022.10041663

[31] Felix Günther, Britta Hale, Tibor Jager, and Sebastian Lauer. 2017. 0-RTT key exchange with full forward secrecy. In *Advances in Cryptology–EUROCRYPT 2017: 36th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Paris, France, April 30–May 4, 2017, Proceedings, Part III 36*. Springer, 519–548.

[32] J. D. Hunter. 2007. Matplotlib: A 2D graphics environment. *Computing in Science & Engineering* 9, 3 (2007), 90–95. doi:10.1109/MCSE.2007.55

[33] James Johnson. 2023. The Vulnerabilities to the RSA Algorithm and Future Alternative Algorithms to Improve Security. (2023).

[34] Panos Kampanakis and Will Childs-Klein. 2024. The impact of data-heavy, post-quantum TLS 1.3 on the Time-To-Last-Byte of real-world connections. *Cryptology ePrint Archive* (2024).

[35] Panos Kampanakis and Michael Kallitsis. 2023. Speeding Up Post-Quantum TLS Handshakes by Suppressing Intermediate CA Certificates. *Amazon Science* (2023). https://assets.amazon.science/2c/3e/bb02f57b45a1a383fdc936a57c45/speeding-up-post-quantum-tls-handshakes-by-suppressing-intermediate-ca-certificates.pdf

[36] Keyfactor. 2024. What is PKI? A Public Key Infrastructure Definitive Guide. https://www.keyfactor.com/education-center/what-is-pki/. Accessed: April 20, 2025.

[37] Hugo Krawczyk and Hoeteck Wee. 2016. The OPTLS protocol and TLS 1.3. In *2016 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, 81–96.

[38] Eva Kupcova, Jozef Simko, Matus Pleva, and Milos Drutarovsky. 2024. Experimental Framework for Secure Post-Quantum TLS Client-Server Communication. In *2024 International Symposium ELMAR*. 213–216. doi:10.1109/ELMAR62909.2024.10694092

[39] Adam Langley, Nagendra Modadugu, and Bodo Moeller. 2016. Transport Layer Security (TLS) False Start. RFC 7918. doi:10.17487/RFC7918

[40] Hyunwoo Lee, Doowon Kim, and Yonghwi Kwon. 2021. TLS 1.3 in practice: How TLS 1.3 contributes to the internet. In *Proceedings of the Web Conference 2021*. 70–79.

[41] Radoslav Mandev and Elif Bilge Kavun. 2023. Performance Comparison of Post-Quantum Signature Algorithms Through An Android Email Application Plug-in. In *2023 IEEE International Conference on Omni-layer Intelligent Systems (COINS)*. 1–6. doi:10.1109/COINS57856.2023.10189254

[42] BIRUK KIROS MELES. 2025. Implementation and Performance Evaluation of Quantum-Safe Cryptography in C: A Comparative Study Using LibOQS and OpenSSL. (Feb. 2025).

[43] Microsoft Research. 2020. Post-Quantum TLS. https://www.microsoft.com/en-us/research/project/post-quantum-tls/. Accessed: April 20, 2025.

[44] National Institute of Standards and Technology. 2024. *Module-Lattice-Based Digital Signature Standard*. Federal Information Processing Standards Publication 204. U.S. Department of Commerce. doi:10.6028/NIST.FIPS.204 Accessed: April 20, 2025.

[45] National Institute of Standards and Technology. 2024. *Stateless Hash-Based Digital Signature Standard*. Federal Information Processing Standards Publication 205. U.S. Department of Commerce. doi:10.6028/NIST.FIPS.205 Accessed: April 20, 2025.

[46] Thierry Notermans. 2021. *How to Optimize TLS Performance?* https://kadiska.com/how-to-optimize-tls-performance/

[47] National Institute of Standards, Technology (NIST), Gorjan Alagic, Quynh Dang, Dustin Moody, Angela Robinson, Hamilton Silberg, and Daniel Smith-Tone. 2024. Module-Lattice-Based Key-Encapsulation Mechanism Standard. doi:10.6028/NIST.FIPS.203

[48] Filip Opiłka, Marcin Niemiec, Maria Gagliardi, and Michail Alexandros Kourtis. 2024. Performance Analysis of Post-Quantum Cryptography Algorithms for Digital Signature. *Applied Sciences* 14, 12 (Jan. 2024), 4994. doi:10.3390/app14124994

[49] José Ignacio Escribano Pablos, Misael Enrique Marriaga, and Ángel L. Pérez del Pozo. 2022. Design and Implementation of a Post-Quantum Group Authenticated Key Exchange Protocol With the LibOQS Library: A Comparative Performance Analysis From Classic McEliece, Kyber, NTRU, and Saber. *IEEE Access* 10 (2022), 120951–120983. doi:10.1109/ACCESS.2022.3222389

[50] Christian Paquin, Douglas Stebila, and Goutam Tamvada. 2020. Benchmarking Post-quantum Cryptography in TLS. In *Post-Quantum Cryptography*, Jintai Ding and Jean-Pierre Tillich (Eds.). Springer International Publishing, Cham, 72–91.

[51] Ray A. Perlner and David A. Cooper. 2009. Quantum Resistant Public Key Cryptography: A Survey. In *Proceedings of the 8th Symposium on Identity and Trust on the Internet (IDtrust '09)*. Association for Computing Machinery, New York, NY, USA, 85–93. doi:10.1145/1527017.1527028

[52] pyca/cryptography. n.d.. Welcome to pyca/cryptography. https://cryptography.io/en/latest/. Accessed: April 20, 2025.

[53] Peter Schwabe, Douglas Stebila, and Thom Wiggers. 2020. Post-Quantum TLS Without Handshake Signatures. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security (CCS '20)*. Association for Computing Machinery, New York, NY, USA, 1461–1480. doi:10.1145/3372297.3423350

[54] Moolchand Sharma, Vikas Choudhary, RS Bhatia, Sahil Malik, Anshuman Raina, and Harshit Khandelwal. 2021. Leveraging the power of quantum computing for breaking RSA encryption. *Cyber-Physical Systems* 7, 2 (2021), 73–92.

[55] Dimitrios Sikeridis, Panos Kampanakis, and Michael Devetsikiotis. 2020. Post-quantum authentication in TLS 1.3: A performance study. *Cryptology ePrint Archive* (2020).

[56] Douglas Stebila, Scott Fluhrer, and Shay Gueron. 2025. *Hybrid key exchange in TLS 1.3.* Internet-Draft draft-ietf-tls-hybrid-design-12. Internet Engineering Task Force. https://datatracker.ietf.org/doc/draft-ietf-tls-hybrid-design/12/ Work in Progress.

[57] Douglas Stebila and Michele Mosca. 2016. Post-quantum key exchange for the internet and the open quantum safe project. In *International Conference on Selected Areas in Cryptography*. Springer, 14–37.

[58] Nick Sullivan. 2017. *Introducing 0-RTT.* https://blog.cloudflare.com/introducing-0-rtt/ Accessed: April 20, 2025.

[59] George Tasopoulos, Jinhui Li, Apostolos P Fournaris, Raymond K Zhao, Amin Sakzad, and Ron Steinfeld. 2022. Performance evaluation of post-quantum TLS 1.3 on resource-constrained embedded systems. In *International Conference on Information Security Practice and Experience*. Springer, 432–451.

[60] Jay Thakkar. 2020. ECDSA vs RSA: Everything You Need to Know. https://www.sectigo.com/resource-library/ecdsa-vs-rsa-everything-you-need-to-know. Accessed: April 20, 2025.

[61] Transactional. 2022. *A Survey of Database TLS Libraries.* https://transactional.blog/blog/2022-database-tls-survey Accessed: April 20, 2025.

[62] Shanika Wickramasinghe. 2024. RSA Algorithm in Cryptography: Rivest Shamir Adleman Explained. https://www.splunk.com/en_us/blog/learn/rsa-algorithm-in-cryptography-rivest-shamir-adleman-explained.html. Accessed: April 20, 2025.

Hybrid Classical-Quantum Cryptography in a Post-Quantum World

Received 25 April 2025