Resource Utilization Impacts of Post-Quantum Digital Signature Standards FIPS 204 and FIPS 205 on Web Server Environments

Bachelor's Thesis in Software Engineering

Johannes Krabbe
E-mail: johannes.krabbe@code.berlin
Address: Fließstraße 5, 12439 Berlin

Date of submission: 23.04.2025

Student ID: 20.01.011
Semester: Spring 2025

First assessor: Peter Ruppel
Second assessor: Adam Roe

Declaration

I hereby confirm that I have written the thesis titled "***Resource Utilization Impacts of Post-Quantum Digital Signature Standards FIPS 204 and FIPS 205 on Web Server Environments***" by myself, without contributions from any sources other than those cited in the text and bibliography. This also applies to all graphics, drawings, and images included in the thesis.

Furthermore, I confirm that neither this work nor parts of it have been previously or concurrently used as an assessment submission in other courses or in other examination proceedings.

Berlin, 23.04.2025

Johannes Krabbe

Contents

List of Figures

List of Tables

1. Introduction

Shor's algorithm [1] in combination with the possibility of advanced Quantum Computers (QCs) threatens to make widely used cryptography algorithms, like RSA or DSA, insecure [2]. Although existing QCs would need a further increase in the number of qubits, the possibility alone already poses the threat of "store now, decrypt later" attacks. To mitigate these attack NIST released the First 3 Finalized Post-Quantum Encryption Standards in 2024, namely the Federal Information Processing Standards (FIPS) 203, 204 and 205. FIPS 203 is intended as the primary standard for general encryption and is already implemented by Cloudflare to protect TLS 1.3 handshakes against "store now decrypt later" attacks. FIPS 204 and FIPS 205 are both standards for quantum secure digital signature algorithms.

This Bachelor thesis investigates which impact the adoption of the Digital Signature Algorithms described in the FIPS 204 and 205 will have on typical Web Server Environments. It will investigate the impact by:

1. Giving an overview of typical Web Server Environments
2. Examining the current role of Digital Signature Algorithms
3. Benchmarking different Pre-Quantum and Post-Quantum digital signature algorithms
4. Discussing the impact Post-Quantum digital signature algorithms will have on current web server environments

2. Background

2.1. Typical Web Server Environments

Web server environments have evolved radically since the early days of the Internet. From simple static file servers in the 1990s to today's complex distributed systems, web infrastructure has continuously adapted to meet growing demands for performance, security, and scalability. This evolution has been marked by shifts from monolithic architectures to micro services, and from physical hardware to virtualized and containerized deployments.

Modern web servers typically operate on multi-core processors with significant RAM allocations to handle concurrent requests efficiently. Enterprise-grade servers often feature dedicated hardware for cryptographic acceleration, while even commodity servers benefit from CPU-level optimizations for common cryptographic operations like those used in RSA and ECDSA algorithms.

The shift toward virtualization and cloud-based deployments has fundamentally changed web server environments. Most production websites now run on virtual machines or containers orchestrated by systems like Kubernetes, allowing for efficient resource utilization and rapid scaling. These environments introduce additional layers that may impact cryptographic performance.

Most web environments on a large scale use sophisticated load balancing techniques to distribute traffic across multiple server instances. This horizontal scaling approach helps manage high traffic volumes but requires consistent cryptographic implementations across all instances, particularly for session persistence and certificate handling.

TLS implementations in web servers have been highly optimized for current cryptographic standards. Servers typically cache session information and employ various acceleration techniques to minimize the performance impact of cryptographic operations. These optimizations are specifically designed around the computational characteristics and signature sizes of algorithms like RSA, ECDSA, and EdDSA.

2.2. The role of Digital Signature Standards in the web

Digital signatures form the backbone of web security, providing authentication and integrity across various protocols and applications. They play a crucial role in estab-

lishing trust in an environment where communicating parties may never physically meet.

2.2.1. TLS Handshakes and Digital Signatures

Transport Layer Security (TLS) relies heavily on digital signatures during the handshake process to establish secure connections. The TLS handshake involves several cryptographic operations where digital signatures are essential.

Server Authentication occurs when a client connects to a server, and the server presents its certificate containing its public key. This certificate is signed by a Certificate Authority (CA) using digital signature algorithms like RSA, ECDSA, or EdDSA.

Certificate Verification happens client side as the client verifies the digital signature in the server's certificate using the CA's public key. This verification confirms the server's identity.

During Key Exchange in the handshake, the server uses its private key to sign certain handshake messages. When using the TLS 1.3 Protocol [3], after receiving the `ClientHello` it responds with the `ServerHello`, the Certificate and a digitally signed transcript hash of all previous handshake messages. In practise this means that whenever a TLS 1.3 handshake happens, there is a signing operation happening on the server.

Key Generation on the server-side involves web servers generating their key pairs during initial setup. The private key is carefully protected on the server, while the public key is incorporated into a Certificate Signing Request (CSR). Certificate Authorities generate their own key pairs used to sign certificates. This means that key Generation operations do not need to happen on every request and are way less frequent.

Verification Operations are happening on the client most of the time. Browsers and applications verify server certificate signatures using CA public keys stored in their trust stores.

2.2.2. Secure Messaging Protocols: The Signal Protocol

The Signal Protocol, developed by Open Whisper Systems [4], is the one of the most popular secure messaging protocols, used by applications like WhatsApp, Signal, and Facebook Messenger. Digital signatures are fundamental to its security model.

The Signal Protocol uses Identity Keys where each user generates a long-term identity key pair. The public key is shared with contacts, while the private key never leaves the device.

Prekeys and Signed Prekeys are generated by users. Users generate multiple prekeys and at least one signed prekey. The signed prekey is signed with the user's identity private key, allowing others to verify its authenticity.

Users publish their public keys used for signatures that establish trust on an per chat basis to the messaging server. Larger key sizes could potentially have an impact on server storage requirements for applications that implement the Signal Protocol.

2.3. Current quantum security measures

The Quantum Threat Timeline Report 2024 by the Global Risk Institute [5] reports on experts' estimates of the likelihood of a quantum computer being able to break RSA-2048 in 24 Hours. Over half of the asked experts are 70 percent sure that there will be a quantum computer in the next 20 years that will be able to do that. Nearly the same amount of experts also said that this happening within the next 5 yeas is less likely than 1 percent.

Although there are currently no quantum computers with enough quantum bits (qbits), quantum security is already a important consideration, because of "store now decrypt later" attacks. Malicious actors can store Pre-Quantum TLS handshake data and messages, that are encrypted with the symmetric keys established through these Pre-Quantum secure TLS handshakes and decrypt them once they have access to a quantum computer with enough qbits.

Cloudflare, one of the largest providers of internet infrastructure, already integrates Post-Quantum encryption into TLS handshakes. The adoption rate of Post-Quantum encryption in April 2025 is already at 38.6%, according to Cloudflare's own data reporting service Cloudflare Radar [6].

This is already an important step into the right direction, since quantum secure encryption protocols, like the Module-Lattice-Based Key-Encapsulation, described in FIPS 203 [7], mitigate "store now decrypt later" attacks.

Digital signature algorithms are used in TLS handshakes to mitigate man in the middle attacks. These man in the middle attacks can only happen while establishing a TLS connection. Therefore these attacks can only become possible once Quantum

computers have enough qbits to break already existing digital signature standards, like ECDSA or EdDSA.

In practice this means that current internet infrastructure still has time to adapt the Quantum secure digital signature standards until quantum computers with enough qbits become viable.

2.4. Security Strength Classification

Different cryptography algorithms, like digital signature algorithms, can provide different strength of security, not only depending on the algorithm used, but also depending on their input parameters. In order to make it easier to compare different algorithms NIST defines a security strength of a certain algorithm with its according parameters as a number associated with the amount of work that is required to break this algorithm [8].

NIST approved a list of five different categories of security strengths: 80, 112, 128, 192 and 256 bits. The security category 1, with the security strength of 80 bits, is no longer considered sufficiently secure [8].

2.5. Pre-Quantum Secure Digital Signature Algorithms

The most prevalent use case for Digital Signature Algorithms are TLS and SSL handshakes. These protocols most commonly use RSA, ECDSA and EdDSA [3]. RSA is most commonly used in older SSL protocols, more recent protocols like TLS 1.3 are relying more on modern Algorisms like ECDSA and EdDSA, instead of RSA. These Algorithms have not only highly optimized Implementations, but also are already considered by CPU manufactures when creating chips and have therefore a very fast execution time.

2.5.1. ECDSA

The Elliptic Curve Digital Signature Algorithm (ECDSA), described in FIPS 186-5 [9], is a discrete logarithm based cryptography algorithm and therefore vulnerable to attacks from quantum computers.

ECDSA can be instantiated with different prime field curves, the approved curves by NIST are P-256, P-384 and P-521. The concrete security categories, according to NIST SP 800-57 Part 1 [8], are 3, 4 and 5 respectively.

The key and signature sizes can be found in Table 1. Note that the public key size can be compressed to half the size.

Table 1. ECDSA key and signature sizes in bytes

	Private Key	Public Key	Signature Size
ECDSA P-256	32	64	64
ECDSA P-384	48	96	96
ECDSA P-521	~66	~131	~131

Source: [9]

2.5.2. EdDSA

The Edwards-curve Digital Signature Algorithm (EdDSA), also described in FIPS 186-5, uses a variant of a Schnorr signature based on twisted Edwards curves [9].

IETF RFC 8032 [10] describes the specific parameters used for the edwards25519 and edwards448 curves. This thesis will only examine the EdDSA algorithm instantiated with the edwards25519 curve.

According to FIPS 186-5 [9], the Edwards-curve digital signature algorithm instantiated with the edwards25519 curve (Ed25519) is intended to provide approximately 128 bits of security and has therefore the security category of 3.

Ed25519 digital signatures have a Public Key and Private Key size of 32 bytes and a signature size of 64 bytes.

2.5.3. Security threads to Pre-Quantum Secure Digital Signature Algorithms

These widely deployed signature algorithms face a significant threat from quantum computing advancements. Shor's algorithm, developed by mathematician Peter Shor in 1994 [1], poses a particularly devastating threat to RSA, ECDSA, and EdDSA [2]. This quantum algorithm can efficiently factor large integers and compute discrete logarithms in polynomial time, effectively breaking the mathematical foundations these cryptographic systems rely on.

The security of RSA depends on the difficulty of factoring large composite numbers into their prime factors, while ECDSA and EdDSA rely on the discrete logarithm problem over elliptic curves. On a sufficiently powerful quantum computer, Shor's algorithm could solve these problems exponentially faster than the best known classical algorithms. For example, a quantum computer with several thousand logical qubits could potentially break a 2048-bit RSA key in hours or days, compared to billions of years using classical computing methods.

This vulnerability creates a significant risk for the entire public key infrastructure that secures current internet communication. If quantum computers reach the necessary scale and stability, attackers could forge digital signatures, impersonate trusted entities and compromise the authenticity guarantees that these signature schemes currently provide. This could undermine the trust model of the internet.

2.6. Quantum Secure Digital Signature Algorithms

In 2016 the National institute for Standards and Technologies (NIST) in the USA called for researchers to submit candidates to start the Post-Quantum cryptography standardization process [11]. NIST then, after putting the suggested algorithms under scrutiny, by several leading cryptography researchers, released their final candidates that should be used for Encryption going forward to ensure protection from advanced quantum computers. NIST released three standards in this context: FIPS 204, 205 and 206. FIPS 204 includes the algorithm responsible for encryption, FIPS 205 and 206 are both standards for Digital signatures.

2.6.1. ML-DSA

FIPS 204 describes the Module-Lattice-Based Digital Signature Algorithm (ML-DSA) which is derived from the CRYSTALS-Dilithium submission to NIST's Post-Quantum Cryptography Standardization Project.

There are three different sets of parameters included for ML-DSA in FIPS 204, namely ML-DSA-44, ML-DSA-65 and ML-DSA-87. These parameter sets are designed to meet certain security categories, explained in SP 800-57, Part 1 [8].

The categories do not describe the security strength as a certain number of bits of security. Instead it claims that the computational resources needed to break ML-DSA with the according parameter set are greater than or equal to the computational resources needed to break a generic block cipher with a prescribed key size or a generic hash function with a prescribed output length. This leads to more or less accurate estimates of security strength, depending on the underlying model of computation used [12].

Concretely the claimed security strength categorys of ML-DSA-44, ML-DSA-65 and ML-DSA-87 are 2, 3 and 5 respectively [12]. The concrete byte sizes of the keys and signatures can be found in Table 2.

Table 2. ML-DSA key and signature sizes in bytes

	Private Key	Public Key	Signature Size
ML-DSA-44	2560	1312	2420
ML-DSA-65	4032	1952	3309
ML-DSA-87	4896	2592	4627

Source: [12]

FIPS 204 is considered as the primary standard for Post-Quantum digital signatures due to its smaller footprint and faster execution times when compared to FIPS 205.

2.6.2. SLH-DSA

FIPS 205 describes the Stateless Hash-Based Digital Signature Algorithm (SLH-DSA) which is derived from the SPHINCS+ submission. This standard is designed as a backup method in case ML-DSA proves vulnerable.

The standards are based on entirely different mathematical problems, providing cryptographic diversity. As NIST explains: "we want to have a backup standard that is based on a different math approach than ML-KEM. As we advance our understanding of future quantum computers and adapt to emerging cryptanalysis techniques, it's essential to have a fallback in case ML-KEM proves to be vulnerable." [13].

FIPS 205 specifies 12 parameter sets that are approved for use, the parameter sets differentiate themselves by the hash function family used to instantiate the hash function (SHAKE or SHA-2), the length in bits of the security parameter and if the parameter set is primary intended for relatively small signature sizes ('s') or relatively fast execution times ('f') [14].

Table 3 lists the concrete key and signature sizes in bytes for the different parameter sets of SLH-DSA. Since there is no difference in size between the hash function family used, the table does not list them individually.

Table 3. SLH-DSA key and signature sizes in bytes

	Private Key	Public Key	Signature Size
SLH-DSA-128s	64	32	7856
SLH-DSA-128f	64	32	17088
SLH-DSA-192s	96	48	16224
SLH-DSA-192f	96	48	35664
SLH-DSA-256s	128	64	29792
SLH-DSA-256f	128	64	49846

Source: [14]

3. Benchmarks

In order to examine the impact of quantum secure digital signature protocols on current webserver environments, I have mesured execution times of currently used algorithms as well as the algorithms described in FIPS 204 and FIPS 205. In order to make these algorithms comparable the benchmarks used the different parameter sets, provided by NIST, so one can judge them based on their security categories, as described in SP 800-57, Part 1 [8].

3.1. Setup

The the Pre-Quantum digital signature algorithms benchmarked are EdDSA and ECDSA. The parameter sets used for the ECDSA Benchmarks are P-256, P-384 and P-521 as described in [9]. The benchmarks for EdDSA are only conducted with the edwards25519 curve, also described in [9].

The Post-Quantum benchmarks are conducted on ML-DSA and SLH-DSA. The Parameter sets used for MLDSA are MLDSA-44, MLDSA-65 and MLDSA-87 as described in FIPS 204 [12]. The SLHDSA benchmarks are only conducted with the SHAKE hash function and use all parameter sets described in FIPS 205 [14], namely SLHDSA-128f, SLHDSA-128s, SLHDSA-192f, SLHDSA-192s, SLHDSA-256f and SLHDSA-256s.

All benchmarks use the RustCrypto/signatures [15] implementation of the above specified algorithms. The exact implementation of the benchmarks can be found in the `main`, `ml-dsa` and `slh-dsa` branch in this repository [16].

All execution times refer to the mean execution time of the raw data, since the maximum derivation measured between the mean and the extreme values are always below 1.6%. The entire raw data can be found here [17]. Each algorithm got benchmarked on the execution time of the key generation, the creation of a signature, the verification of a signature and a round trip time. The size of the signature payload, used for the benchmarks, is 128 bytes, to represent typical TLS certificate sizes.

All benchmarks were conducted on a Ubuntu 24.04.2 root server, powered by an `Intel Core i5-12500` processor. To ensure comparability between the benchmarks, the benchmarks were executed one by one and the server was not tasked with any other processes.

3.2. Results

The results of the signature execution time benchmarks, found in Table 4, show that the SLHDSA algorithm is by far the slowest of every algorithm tested.

When looking at the SLHDSA results closer, a particularly interesting result of the benchmark is that the SLHDSA-192s parameter set has a roughly 55% slower key generation and roughly 16% slower verification time then the SLHDSA-256s parameter set, desptie the latter offering higher security.

To verify that the benchmarking code itself is not flawed, I ran the benchmarks that can be found in the library repository on GitHub [15] and got similar results. There are several reasons why this could happen, for example the server hardware used for this test could be particularly optimized to run 256-bit operations, or there is still optimization in this particularly implementation of the code. This thesis will not analyze the cause of this unexpected result.

Table 4. Signature algorithm execution times

Algorithm	keygen	sign	verify	round_trip
ECDSA P-256	0.095 ms	0.111 ms	0.187 ms	0.395 ms
ECDSA P-384	0.392 ms	0.432 ms	0.786 ms	1.611 ms
ECDSA P-521	0.576 ms	0.618 ms	1.024 ms	2.213 ms
Ed25519	0.011 ms	0.012 ms	0.023 ms	0.048 ms
MLDSA-44	0.115 ms	0.191 ms	0.031 ms	0.338 ms
MLDSA-65	0.189 ms	0.267 ms	0.043 ms	0.503 ms
MLDSA-87	0.29 ms	0.147 ms	0.062 ms	0.509 ms
SLHDSA-128f	2.361 ms	54.765 ms	3.383 ms	60.397 ms
SLHDSA-128s	151.696 ms	1160.359 ms	1.123 ms	1304.848 ms
SLHDSA-192f	3.454 ms	88.929 ms	4.808 ms	97.457 ms
SLHDSA-192s	222.028 ms	1988.104 ms	1.662 ms	2213.755 ms
SLHDSA-256f	8.998 ms	180.93 ms	4.842 ms	194.031 ms
SLHDSA-256s	143.464 ms	1713.669 ms	2.337 ms	1857.408 ms

3.2.1. Signing speed and signature size

Signing speed and signature size are the most important values that need to be considered when looking at the resource utilization impact, digital signature algorithms have on web server environments. The most common digital signature operation on

servers is the signing operation, since this operation must occur in virtually every TLS handshake on the server.

3.2.1.1. Security Category 3

The algorithms Ed25519 and ECDSA P-256, which are most commonly used in TLS 1.3 handshakes, fall into category 3, therefore this is the most important category the digital signature algorithms should be compared in. The Post-Quantum digital signature algorithms wich are also classified in category 3 are MLDSA-44, SLHDSA-192f and SLHDSA-192s.

When looking at the visualization of the execution time of the signing speed in relation to the signature size, shown in Figure 1, one can see that both signing time and signature size of the Pre-Quantum digital signature algorithms are virtually zero when compared to the SLHDSA algorithms.

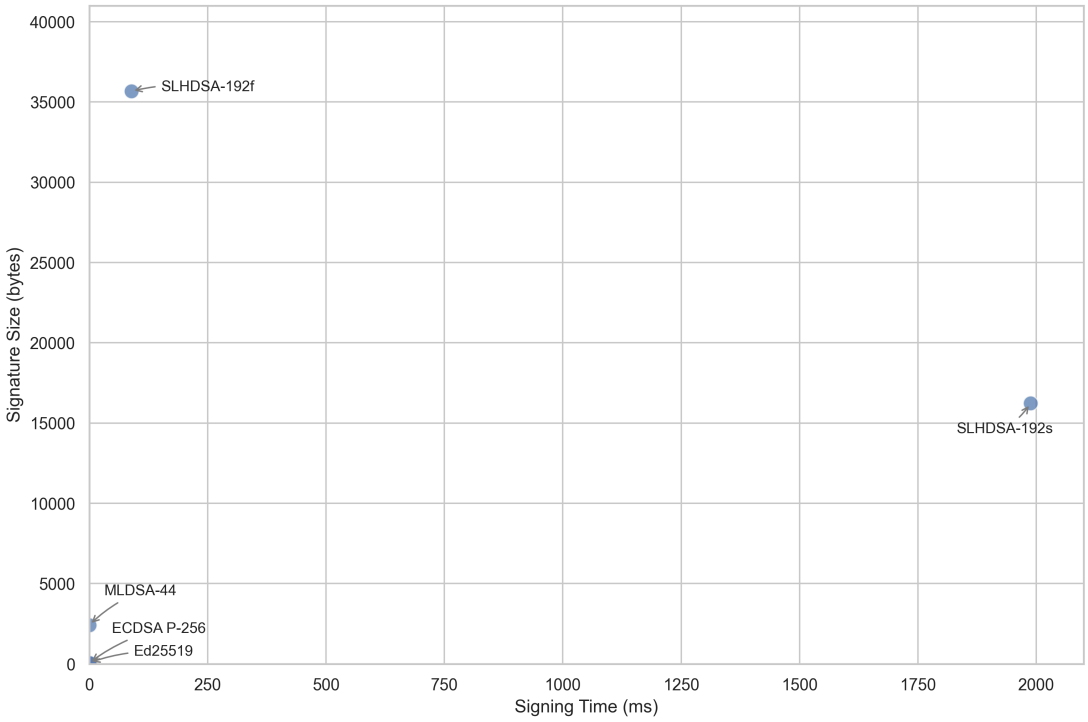

Figure 1. Signing Speed in relation to signature size of category 3 signature algorithms

Compared to the fastest Pre-Quantum algorithm, Ed25519, all Post-Quantum algorithms are significantly slower. MLDSA-44 is more than 15 times slower as Ed25519, SLHDSA-192f is more than 7400 times slower and SLHDSA-192s is more than 165000 times slower.

When compared to the, also widely used, ECDSA P-256 algorithms one can still see that MLDSA-44 is roughly 1.7 times slower, SLHDSA-192f is more than 800 times slower and SLHDSA-192s is more than 17000 times slower.

The signature sizes of the Post-Quantum digital signature algorithms are also significantly larger than the 64 bytes that can be found in both Pre-Quantum algorithms. Concretely the signatures of MLDSA-44 are more than 37 times larger, the signatures of SLHDSA-192s are more than 250 times larger and the signatures of SLHDSA-192f are more than 550 times the size of the signatures produced by the Pre-Quantum algorithms.

These results clearly show why FIPS 205, describing the SLHDSA signature algorithm, is intended as a backup for FIPS 204. Even optimized for fast execution times (SLHDSA-192f) or optimized for small signature sizes (SLHDSA-192s), the algorithm comes no where near the speed or size provided by MLDSA.

3.2.1.2. Security Category 5

With both regular computers and quantum computers, getting more computing power, algorithms that are more secure will be needed in the future. Security Category 5, with 256 bits of security is currently the highest category defined in NIST SP 800-57 Part 1 [8]. From the algorithms tested, ECDSA P-521, MLDSA-87, SLHDSA-256s and SLHDSA-256f are the only algorithms believed to have at least 256 bits of security and therefore fall into security category 5.

An interesting obervation that can be made when comparing execution times of ECDSA P-521 and MLDSA-87 is that the Pre-Quantum algorithm has a significantly slower execution time than the Post-Quantum algorithm. Signature sizes of MLDSA-87 are still more than 34 times larger than the ECDSA P-521, but in environments where key and signature sizes are less important then speed, MLDSA-87 is a better choice than, even when disregarding quantum security.

Both SLHDSA variants (SLHDSA-256s and SLHDSA-256f), still have no advantage over any other algorithm in this category and therefore should only be used if MLDSA proves vulnerable.

3.2.2. Key Generation Performance

Key generation is typically a one-time operation for servers but remains important for understanding the full lifecycle of cryptographic implementations. The benchmarks reveal significant differences in key generation times across algorithms.

The MLDSA variants have a key generation time range from 0.115 ms for MLDSA-44 to 0.29 ms for MLDSA-87, remaining competitive with ECDSA implementations.

The SLHDSA variants are dramatically slower, with key generation times in the "s" variants being particularly resource-intensive. SLHDSA-192s requires 222.028 ms, which is over 20000 times slower than Ed25519.

The key generation performance has implications for scenarios requiring frequent key rotation or ephemeral keys. While most web servers generate keys infrequently, environments with high security requirements that mandate regular key rotation would face significant overhead when using SLHDSA variants.

3.2.3. Verification Performance

Verification performance is particularly important for the client side of TLS handshakes.

The "f" variants of the SLHDSA algorithms show verification times between 3.383 ms and 4.842 ms, while the "s" variants perform better at verification with times between 1.123 ms and 2.337 ms. These times are still drastically slower than any other algorithm tested.

Interestingly, MLDSA shows excellent verification performance, with MLDSA-44 requiring only 0.031 ms, making it competitive with Ed25519 and faster than all ECDSA variants.

A notable observation is that MLDSA algorithms demonstrate an asymmetric performance profile, with verification being significantly faster than signing. This characteristic could make MLDSA well suited for TLS applications where a server needs to sign once but the clients frequently need to verify the signature.

4. Discussion

This thesis examines the impact of quantum secure digital signature algorithms on typical web server environments. The diverse range of web applications with varying workloads means there is no single answer regarding how MLDSA or SLHDSA implementation will affect webserver environments when integrated into TLS handshakes.

4.1. Impact on Different Web Service Types

4.1.1. High-Bandwidth Services

For companies like Netflix that primarily focus on video streaming, most computing resources are allocated to video encoding and delivery. TLS handshakes represent a minor portion of their overall computing demands. Therefore, implementing quantum secure digital signatures would likely have a relatively small impact on their server infrastructure, as the application code is already resource-intensive.

4.1.2. E-commerce Platforms

E-commerce platforms like Amazon serve numerous users who individually consume less computing power compared to streaming services. While TLS handshake data can be cached for returning users, these platforms must handle millions of unique sessions daily. The benchmarks indicate that replacing current signature algorithms with MLDSA would increase signing time by a factor of 1.7-15x (depending on whether comparing to ECDSA P-256 or Ed25519). This could significantly impact server capacity requirements during peak shopping periods.

4.1.3. API Services

For API-centric services that establish numerous short-lived connections, the impact could be substantial. Each API call typically requires a new TLS handshake. The increased signing time of post-quantum algorithms would directly affect request latency and server throughput. MLDSA-44 would introduce moderate overhead, while SLHDSA variants would likely be very expensive for high-volume API services without significant architectural changes.

4.1.4. Network Bandwidth Considerations

Beyond computational costs, the increased signature sizes have implications for network bandwidth. The MLDSA-44 signatures are approximately 37 times larger than current Ed25519 signatures, while SLHDSA variants are 250-550 times larger. This increased payload size of TLS handshake data will be noticable by users in

constrained bandwidth environments, for example mobile users in areas with slower connection speeds.

Content delivery networks (CDN) are often billed based on data transferred. Depending of the relative size of the content, compared to signature sizes, there might be a noticeable impact on costs.

4.1.5. IoT Devices

Internet of Things (IoT) devices present a particularly challenging case for post-quantum cryptography implementation. These devices are often characterized by very limited computational resources. They often operate on low-power microcontrollers with minmal processing capabilities and constrained memory in both RAM and flash storage.

For these devices, the impact of post-quantum digital signatures would most likely be profound. The benchmark results show that even the most efficient Post-Quantum algorithm tested (MLDSA-44) requires significantly more computational resources than current algorithms. The SLHDSA variants would likely be completely impractical for most IoT implementations due to their extreme computational demands.

The memory constraints of IoT devices also make the larger key and signature sizes problematic. With MLDSA-44 signatures being 37 times larger than Ed25519 signatures, and SLHDSA variants being 250-550 times larger, these increased sizes could exceed the available memory on many devices. This is particularly concerning since IoT devices often cannot cache TLS connection data and must establish new connections frequently, amplifying the impact of the increased computational and memory requirements.

Additionally, the energy consumption implications are significant. The increased computational demands of Post-Quantum algorithms would lead to higher power consumption, potentially reducing battery life by a substantial margin. For devices expected to operate for years on a single battery, this could render current designs non-viable.

IoT device manufacturers will likely need to upgrade hardware specifications, by adding generally stronger computing chips and bigger RAM and flash storage. Implementing specialized hardware accelerators for cryptography algorithms could also be a big opportunity to mitigate some of the beforementioned problems.

4.2. Mitigation Strategies

To address these challenges, web services might need to optimize the use of digital signature algorithms by maximizing TLS session reuse, in order to reduce the cost of handshakes over multiple requests.

Future chips could also include hardware optimizations for Post-Quantum algorithms. Depending on research in this field, execution times might be able to be reduced on newer hardware. But replacing old hardware with newer, more optimized, hardware is a long process, so even if there will be improved chips it will take some time for big server clusters to completely switch to more efficient hardware.

4.3. Security vs. Performance Tradeoffs

The benchmarks clearly demonstrate the tradeoff between security and performance. While MLDSA offers a reasonable compromise with manageable performance impact, SLHDSA presents significant challenges for widespread adoption in its current form. Web service providers will need to carefully evaluate their specific security requirements against the performance implications.

5. Conclusion

The transition to Post-Quantum digital signature algorithms represents a significant challenge for web server environments. The benchmarks conducted in this thesis demonstrate that while MLDSA offers a reasonable path forward with manageable performance impact, SLHDSA in its current form presents substantial challenges for widespread adoption due to its computational demands and signature sizes.

Web service providers will need to carefully plan their transition strategies. The specific impact will vary considerably based on the nature of the web service, with services relying on a lot of different users establishing new connections frequently likely facing greater challenges than content delivery platforms, like video streaming services.

As quantum computing advances continue, further optimization of Post-Quantum algorithms is essential. The significant performance gap between classical and Post-Quantum algorithms highlighted in this research underscores the importance of continued research and development in this area to ensure the security of web communication in the Post-Quantum era without unacceptable performance loss.

Future work should focus on:

1. Hardware acceleration techniques for post-quantum algorithms
2. Protocol optimizations to reduce the impact of larger signature sizes
3. Real-world deployment case studies across various web service types

The web infrastructure that underlies modern digital life will require significant adaptation to maintain both security and performance in a Post-Quantum world, but with proper planning and continued algorithmic improvements, a smooth transition appears achievable.

References

[1] P. Shor, "Algorithms for quantum computation: discrete logarithms and factoring," in *Proceedings 35th Annual Symposium on Foundations of Computer Science*, Nov. 1994, pp. 124–134. doi: 10.1109/SFCS.1994.365700.

[2] S. Singh and E. Sakk, *Implementation and Analysis of Shor's Algorithm to Break RSA Cryptosystem Security*. 2024. doi: 10.36227/techrxiv.170259160.05374043/v2.

[3] E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.3," Aug. 2018. doi: 10.17487/RFC8446.

[4] Open Whisper Systems, "The Signal Protocol." Accessed: Apr. 22, 2025. [Online]. Available: https://signal.org/docs/

[5] Global Risk Institute, "Quantum Threat Timeline Report 2024," Dec. 2024. Accessed: Apr. 20, 2025. [Online]. Available: https://globalriskinstitute.org/mp-files/quantum-threat-timeline-report-2024.pdf/

[6] "Adoption & Usage Worldwide | Cloudflare Radar." Accessed: Apr. 20, 2025. [Online]. Available: https://radar.cloudflare.com/adoption-and-usage

[7] National Institute of Standards and Technology (US), "Module-lattice-based key-encapsulation mechanism standard," Washington, D.C., Aug. 2024. doi: 10.6028/NIST.FIPS.203.

[8] E. Barker, "Recommendation for Key Management: Part 1 – General," May 2020. doi: 10.6028/NIST.SP.800-57pt1r5.

[9] National Institute of Standards and Technology (US), "Digital Signature Standard (DSS)," Washington, D.C., Feb. 2023. doi: 10.6028/NIST.FIPS.186-5.

[10] S. Josefsson and I. Liusvaara, "Edwards-Curve Digital Signature Algorithm (EdDSA)," Jan. 2017. doi: 10.17487/RFC8032.

[11] "Submission Requirements and Evaluation Criteria for the Post-Quantum Cryptography Standardization Process." Dec. 2016.

[12] National Institute of Standards and Technology (US), "Module-lattice-based digital signature standard," Washington, D.C., Aug. 2024. doi: 10.6028/NIST.FIPS.204.

[13] "NIST Releases First 3 Finalized Post-Quantum Encryption Standards," *NIST*, Aug. 2024, Accessed: Apr. 18, 2025. [Online]. Available: https://www.nist.gov/news-events/news/2024/08/nist-releases-first-3-finalized-post-quantum-encryption-standards

[14] National Institute of Standards and Technology (US), “Stateless hash-based digital signature standard,” Washington, D.C., Aug. 2024. doi: 10.6028/NIST.FIPS.205.

[15] “RustCrypto/signatures: Cryptographic signature algorithms: DSA, ECDSA, Ed25519.” Accessed: Apr. 18, 2025. [Online]. Available: https://github.com/RustCrypto/signatures

[16] J. Krabbe, “Johannes-Krabbe/dsa-bench.” Accessed: Apr. 18, 2025. [Online]. Available: https://github.com/Johannes-Krabbe/dsa-bench

[17] J. Krabbe, “Digital Signature Algorithm Benchmarks.” Accessed: Apr. 22, 2025. [Online]. Available: https://zenodo.org/records/15263914

Appendix

A Tools & Fonts Used

This thesis was typeset using **Typst** (https://github.com/typst/typst).

The fonts used are **Common Serif**, Copyright © 2022 Common Serif Project Authors, and **Fira Code**, Copyright © 2024 The Fira Code Authors. Both are licensed under the SIL Open Font License 1.1, available at https://openfontlicense.org.