Post-Quantum Authentication in TLS 1.3: A Performance Study

Dimitrios Sikeridis*, Panos Kampanakis†, Michael Devetsikiotis*
* ECE Dept., Univ. of New Mexico, USA, dsike@unm.edu / mdevets@unm.edu
†Security & Trust Organization, Cisco Systems, USA, panosk@cisco.com

***Abstract*—The potential development of large-scale quantum computers is raising concerns among IT and security research professionals due to their ability to solve (elliptic curve) discrete logarithm and integer factorization problems in polynomial time. All currently used public key algorithms would be deemed insecure in a post-quantum (PQ) setting. In response, the National Institute of Standards and Technology (NIST) has initiated a process to standardize quantum-resistant crypto algorithms, focusing primarily on their security guarantees. Since PQ algorithms present significant differences over classical ones, their overall evaluation should not be performed out-of-context. This work presents a detailed performance evaluation of the NIST signature algorithm candidates and investigates the imposed latency on TLS 1.3 connection establishment under realistic network conditions. In addition, we investigate their impact on TLS session throughput and analyze the trade-off between lengthy PQ signatures and computationally heavy PQ cryptographic operations. Our results demonstrate that the adoption of at least two PQ signature algorithms would be viable with little additional overhead over current signature algorithms. Also, we argue that many NIST PQ candidates can effectively be used for less time-sensitive applications, and provide an in-depth discussion on the integration of PQ authentication in encrypted tunneling protocols, along with the related challenges, improvements, and alternatives. Finally, we propose and evaluate the combination of different PQ signature algorithms across the same certificate chain in TLS. Results show a reduction of the TLS handshake time and a significant increase of a server's TLS tunnel connection rate over using a single PQ signature scheme.**

I. Introduction

Digital communications have completely penetrated everyday life and the physical world as enablers of numerous critical services including telemedicine, online banking, massive e-commerce, machine-to-machine automation, mobile and cloud computing. In this reality, public-key cryptography is ubiquitous in almost all cryptographic protocols, such as TLS which builds encrypted tunnels between digital entities. Studies suggest that over 60% of Internet connections are implemented over the TLS-based secure HTTPS protocol [20], [62], while TLS adoption is expected to keep increasing as users and client vendors strive for ubiquitous encryption and privacy [50].

Apart from connection integrity and confidentiality, TLS also provides authentication usually with the use of X.509 certificates [45]. Such certificates are issued by trusted third-parties called Certificate Authorities (CAs). Endpoints verify the communicating peer's identity and public key (PK) contained inside his certificate by leveraging a chain of certificates that is rooted to a pre-trusted root CA. The two most popular digital signature algorithms used in certificates today are the Elliptic Curve Digital Signature (ECDSA) and Rivest–Shamir–Adleman (RSA). Their security guaranties rely on the hardness of the elliptic curve discrete logarithm (ECDL) and integer factorization (IF) problems respectively.

While the security of the aforementioned schemes cannot be practically challenged by conventional computer systems, this would not be the case in a post-quantum world where a large scale quantum computer has become a reality [60]. Shor's quantum algorithm [74], [85], assuming a practical quantum computer (QC) is available, would solve ECDL and IF problems in polynomial time which would render ECDSA and RSA insecure. In this scenario a QC-equipped attacker would be able to impersonate signers that use these algorithms. Thus, encrypted tunnel (e.g. TLS, IKEv2) authentication, PK infrastructure (PKI), CAs, and software signing would be broken.

To address this, the cryptographic community has been researching quantum-resistant public key algorithms for some time, while the US NIST started a public project to standardize quantum-resistant public key encapsulation and signature algorithms. Similarly, ETSI has formed a Quantum-Safe Working Group [32] that aims to make assessments and recommendations on the various proposals from industry and academia regarding real-world deployments of quantum-safe cryptography. Moreover, the IETF has seen multiple proposals that attempt to introduce and investigate PQ algorithms in protocols like TLS and IKE [33], [40], [66], [89], [91].

At the moment of this writing, NIST's evaluation process has moved from Round 1 to Round 2 where 26 PQ algorithms were chosen with security guarantees being the primary criterion, while performance was treated as a future goal [2]. Evidently, the actual integration of these algorithms into existing protocols (e.g., TLS, IKEv2, SSH) and their coexistence with today's Internet infrastructure present challenges that pertain to (a) additional latency due to their heavy operations, (b) communication overhead impact from the increased public key and signature sizes, and (c) optimal use of existing hardware towards faster implementations. This gap is actively being studied by research teams in the industry with NXP Semiconductors, Microsoft and Queensland University of Technology [15], Amazon [18], Cloudflare and Google focusing on the impact of key exchange mechanisms (KEMs)

eprint manuscript received on 1/23/2020. Early draft. Final paper to appear in NDSS 2020, 23-26 Feb. 2020, San Diego, CA, USA
ISBN 1-891562-61-4, https://dx.doi.org/10.14722/ndss.2020.24203
www.ndss-symposium.org

on TLS [51], [53]–[55].

These efforts mostly focus on TLS PQ key exchange as confidentiality is considered more urgent. Since a QC-equipped attacker would be able to decrypt stored communications retroactively, ensuring quantum-resistant transmissions for critical data is a priority. This is not the case with authentication as digital entity impersonation cannot happen retroactively. However, there are numerous incentives that drive the early study of PQ authentication in today's protocols. First, PKI refresh cycles are traditionally long and migration to new primitives can take years. A case in point is ECDSA. Even though it was standardized in 2005 [4] offering clear performance advantages over RSA [39], its adoption was still not broad a decade later. Finally, the computational performance of PQ algorithms, along with the fact that the sizes of the resulting PQ certificates are significantly larger, will definitely impact the TLS handshake by worsening user experience and connection performance. Thus, it is important to investigate and identify promising PQ signature candidates specifically for utilization in TLS.

In this paper, we study the overhead introduced by PQ certificates in the establishment of TLS 1.3 tunnels. Our goal is to identify PQ signature candidates that could be employed in TLS 1.3 without major protocol updates, measure their performance in real-world deployments and contribute to the overall discussion about their use in encrypted tunnels. The **key contributions of our work** are summarized as follows:

(i) We analyze the candidate signature algorithms from NIST's Post-Quantum Cryptography Project and compare them in terms of performance, security level claims and key/signature sizes. In addition, we integrate software implementations of these schemes in X.509 certificates for TLS 1.3 authentication in the OQS OpenSSL library [72].

(ii) We conduct large-scale measurements to investigate the PQ authentication algorithm impact on TLS 1.3 handshake establishment in realistic network conditions. Moreover, we investigate the impact that larger PQ certificate chains or slower sign/verify operations have on the throughput of a server performing PQ authentication in HTTPS.

(iii) We demonstrate the viability of adopting two PQ candidate signature algorithms, Dilithium, and Falcon, for time-sensitive applications over TLS and we argue that Falcon is more suitable for the web if floating point hardware is available at the server. We argue that less time-sensitive applications can use a larger set of PQ candidate algorithms.

(iv) We propose and study the use of different PQ signature scheme combinations in the same certificate chain to improve the performance of PQ authentication in TLS which, to the best of our knowledge, has not been examined before.

(v) We provide an in-depth discussion of PQ authentication challenges for encrypted tunnelling protocols, alternatives, and present insights towards optimizing future deployments.

Note that we do not evaluate the PQ security claims or security proofs of the PQ algorithms. These will be assessed in the NIST standardization process. We also do not perform an exhaustive benchmark of all the available parameter sets of the signature algorithms under consideration by NIST. Relying on preliminary findings and individual algorithm claims, we have explicitly chosen to focus on a subset of algorithms and parameter sets that would seamlessly fit in TLS. To the best of our knowledge, this is the first work that assesses the performance of PQ certificates in TLS 1.3 by considering realistic network conditions.

The rest of the paper is organized as follows: Section II presents background on X.509 PKI and TLS 1.3. In Section III, the different PQ candidate signature families and algorithms are presented, while Section IV details the integration of PQ authentication into TLS 1.3. Section V presents the experimental procedure, results and their analysis. Section VI summarizes related work. Finally, Section VII discusses the general implications and potential solutions of integrating new PQ signatures in encrypted tunnels, while Section VIII concludes this paper.

II. Background

This section presents an overview of the TLS 1.3 handshake protocol, along with a summary of the X.509 PKI currently used in TLS.

A. X.509 Certificates and PKI

A digital entity's (e.g., a server) identity is bound to its public key via a digital certificate. Since X.509 is the most common PKI standard adopted by IETF protocols, X.509 certificates play an important role in digital authentication for various protocols (e.g., TLS, SSH, IKEv2). X.509 certificates are defined by RFCs 5280 [22] and 6818 [92], and their sizes usually vary between 0.5 and 1.5KB.

A PKI infrastructure consists of various parts. A Certificate Authority (CA) issues an entity's certificate that assures the entity's identity and public key tie to that identity. The identity is included in the `Subject` field of the certificate, while the entity's public key is stored in the `Subject Public Key Information` along with the algorithm used by the issuer to create the signature. A certificate contains a specific validity period and extensions included by CAs to enable additional functionality. The certificate is signed by the CA's private key using the specified signature algorithm and the signature is added to the certificate's `Signature` field.

Certificates are exchanged between entities during a session setup in order for each party to verify the peer's identity. At the top of the X.509 PKI there are trusted CAs that self-sign their own certificates known as root CA certificates. Normally a root CA issues certificates for intermediate CAs (ICAs). Following that, the root CA is kept offline for security purposes. An ICA can further issue certificates for other ICAs that in turn sign leaf certificates in the PKI. This process results to the creation of certificate chains of trust that usually consist of two to four certificates but can be arbitrarily long. A leaf certificate is validated by an endpoint if (a) the endpoint trusts the chain's root CA, and (b) all the signatures from the leaf to the root certificate of the chain are verified by using the public key of the issuer.

B. TLS 1.3 Encrypted Tunnels

The TLS protocol design provides endpoint authentication and establishes encrypted communication tunnels between

them. The aim is to ensure data integrity, confidentiality, and authenticity. In August 2018, the latest version of the protocol —TLS 1.3— was published as RFC 8446 [76]. TLS 1.3 offers significant improvements over its predecessor TLS 1.2 that include elimination of insecure or obsolete features, complete encryption of the handshake procedure, and reduced handshake latency by eliminating a round-trip.

Below we present the full TLS 1.3 handshake, and a summary of the messages exchanged between a client that initiates a connection with a remote server: Initially, the client calculates an ephemeral public/private key pair for the key exchange and sends to the server a `ClientHello` message that contains a random nonce, the protocol versions and cipher suites that the client supports, and possible extensions which include pre-shared keys, a list of supported signature algorithms (`signature_algorithms` extension), or a list of supported signature algorithms specifically for certificates (`signature_algorithms_cert` extension).

In turn, the server calculates its own ephemeral key pairs, determines the desired cryptographic parameters and responds with a `ServerHello` message that contains the server's nonce, a public key for the key exchange, the preferred cipher suite and optionally `key_share` and `pre_shared_key` extensions. At this point, by combining the `ClientHello` and `ServerHello` messages the two entities generate a shared key and the connection becomes encrypted. The server continues by sending a `Server Change Cipher Spec` message for TLS 1.2 compatibility purposes ("middlebox compatibility mode"). This is followed by the server sending `ServerEncryptedExtensions` extensions, and optionally a `ServerCertificateRequest` message if client certificate authentication is required. The encrypted extensions include the server's certificate and certificate chain for authentication (`ServerCertificate`), and a `ServerCertificateVerify` message that contains a signature over the handshake. Finally, the server ends his part of the handshake with a `ServerFinish` message that contains a Message Authentication Code (MAC), namely verification data generated by a hash of all the messages exchanged so far.

Following that, and again for compatibility purposes, the client responds with a `Client Change Cipher Spec` message to the client. If client authentication is requested the client then sends its public key certificate (a `ClientCertificate` message) and optionally a `ClientCertificateVerify`. The TLS 1.3 handshake finishes with the client `ClientFinish` message. The complete TLS 1.3 handshake and its variations are detailed in RFC 8446 [76], while an illustrated byte-per-byte description can be found in [29].

III. Post-Quantum Candidate Signature Schemes

In this section we present the different families of PQ signature algorithms, and document details regarding the specific schemes and parameter sets used in this study.

A. Quantum-Resistant Families of Problems

Since currently used public-key cryptography schemes would be threatened by large-scale quantum computers, research for alternatives —namely post-quantum schemes— able to resist QC attacks has been surging for the last decade. The goal has been to identify suitable problems or subproblems of NP-hardness that are not solvable in polynomial time by quantum algorithms [16].

Hash-based Cryptography This family of PQ signature algorithms relies on Merkle trees and few or one-time-signature (FTS/OTS) used with secure cryptographic hash functions. Important security requirements for these functions include collision and preimage resistance. The first scheme in the family, Merkle signature scheme (MSS), was presented in the late 1970s [16]. The use of hash functions with Merkle trees and FTS/OTS for generating signatures is considered mature, well-understood, and significantly reliable. Hash-Based Signature (HBS) schemes generate keypairs for the FTS/OTS. The FTS/OTS signs a message and the corresponding public key becomes a leaf in the Merkle tree. The resulting Merkle tree root is the public key. Currently, the most mature schemes of this family are the stateful LMS [58] and XMSS [43], and stateless SPHINCS$^+$, one of the NIST signature candidates [5]. A stateful HBS relies on an OTS and a signer needs to ensure that the OTS private key is never reused. This state management requirement is considered an important disadvantage. While stateless SPHINCS$^+$ alleviates this issue, it also leads to an increase in signature size to $\sim$40-60 KB and slower performance.

Lattice-based Cryptography Another family of hard problems rely on lattices [63], [68]. A lattice is the set of all integer linear combinations of linearly independent vectors in real n-space $\mathbb{R}^n$. There are many lattice-related NP-hard problems used for cryptographic purposes including the shortest vector problem (SVP) (i.e., find a shortest vector in the Euclidean norm), the closest vector problem (CVP) (i.e., find a lattice vector that minimizes the distance from another target lattice), and the lattice basis reduction problem. The lattice problems NIST PQ signature candidates depend on are learning with errors (LWE) [75], ring learning with errors (RLWE) [57], module learning with rounding or errors (MLWR or MLWE) and NTRU [41]. Some of these may be reducible to these NP-hard lattice problems. The NTRU and LWE families have been studied more extensively than others. Lattice-based algorithms are promising quantum-resistant solutions with relatively efficient implementations and strong security properties. Among the NIST PQ signature candidates the list of lattice-based schemes includes Dilithium [30], qTesla [3], and Falcon [34].

Multivariate Cryptography Another family of problems that are used by some NIST's signature algorithm candidates is related to solving multivariate quadratic equations over finite fields which is an NP-hard problem. The system's hardness to solve depends mainly on the degree, the number of variables, and the underlying finite field's size. Multivariate PQ schemes often lead to excessive key or signature sizes. Thus, recent research has focused on reducing keys sizes for these schemes [16]. The NIST multivariate-based signature candidates are Rainbow [27], MQDSS [21], LUOV [12], and GeMSS [19].

Finally, Picnic [38] is a NIST candidate signature scheme that does not rely on structure hardness (lattice, quadratic equations, Merkle trees). Picnic depends on zero-knowledge proofs, and symmetric key primitives like hash functions and block ciphers.

Signature Algorithm	Specification	Hard Problem	Public Key Size (Bytes)	Private Key Size (Bytes)	Signature Size (Bytes)	Claimed Classical Security Level	Claimed PQ Security Level
RSA 3072	[59]	Integer Factorization	387	384	384	128 bits	∼0 bits
ECDSA 384	[4]	EC Discrete Logarithm	48	48	48	192 bits	∼0 bits
Dilithium II	[30], [31]	Module Learning with Errors	1184	2800	2044	100 bits	91 bits
Falcon 512	[34]	NTRU	897	1281	690	114 bits	103 bits
MQDSS 48	[21]	Multivariate	46	13	20854	160 bits	99 bits
Picnic $L1FS$	[38]	Zero-Knowledge Proofs	33	49	34036	128 bits	64 bits
SPHINCS$^+$ SHA256-128f-simple	[5]	Hash-Based	32	64	16976	128 bits	64 bits
Rainbow Ia - $Cyclic$	[27]	Multivariate	58144	92960	64	143 bits	106 bits
Dilithium IV	[30], [31]	Module Learning with Errors	1760	3856	3366	174 bits	158 bits
Falcon 1024	[34]	NTRU	1793	2305	1330	263 bits	230 bits

TABLE I: Conventional and Post-Quantum Signature Algorithms and Parameter Sets used in this study.

B. PQ Signature Algorithms and Parameter Sets Studied

In their PQ Project, NIST defined five security levels based on the bits of security offered by the algorithm parameter set. In our study, we focus on Level 1, 3 and 5. Level 1 corresponds to 128-bit security, while Levels 3 and 5 offer 192 and 256 bits of security respectively. The choice of specific parameter sets for each algorithm in our study was done specifically for use in TLS, and by taking into account their performance and signature/key size outputs.

SPHINCS$^+$ [5]: The SPHINCS$^+$ signature algorithm specification defines 36 different parameter sets, that utilize different hash functions including Haraka, SHA256, and SHAKE256. For our experiments, we chose to integrate and evaluate the `SHA256-128f-simple` parameter set that corresponds to NIST's Level 1 because it was the most efficient one integrated in PQClean [73].

Dilithium [30], [31]: In our study, we integrated and evaluated Dilithium II which was in PQClean [73] and corresponds to NIST's Level 1. We also examined Dilithium IV, Dilithium's highest security level (Level 3). We also tested Dilithium III (Level 2) as a present alternative since Dilithium II offers less than 128 bits of classical security.

Falcon [34]: Regarding the Falcon signature scheme, its Round 2 submission describes two parameter sets, Falcon 512 that provides NIST's Level 1 security, and Falcon 1024 that corresponds to Level 5. In our experiments we integrated and evaluated the version provided by the Falcon team in `liboqs` [71], [88]. That version did not include the floating point hardware optimizations that improve Falcon's signing performance by ∼20 times [70].

qTesla [3]: Initially, the heuristic version of the qTesla I parameter set (NIST Level 1) tested as a very promising TLS candidate. However, at the time of this writing, an update on qTesla's Round 2 submission eliminated all Round 2 heuristic parameter sets. In their place the qTesla team kept the provably-secure parameter sets that result in significantly bigger signature (2.5KB) and public key (∼15KB) which would double the handshake time in TLS. Therefore, we did not evaluate qTesla further.

Picnic [38]: In this study, we examine the Round 1 Picnic L1FS parameter set which existed in the OQS OpenSSL [72] and corresponds to NIST's Level 1. The Round 2 submission preserves all Round 1 parameter sets, and adds another three that are referred to as Picnic2. This new parameter set reduces the size of the produced signature by making the whole scheme significantly slower while the signature is still over 10KB [37, § 11.3, § 11.4]. Thus, although our evaluation of Picnic is slightly optimistic, it still does not perform well in TLS.

MQDSS [21]: The MQDSS specification provides multiple parameter sets, and specifically recommends two of them. MQDSSS-31-48 corresponds to NIST Levels 1-2. In our study, we integrated and evaluated the MQDSSS-31-48 implementation from `liboqs` [71], [88]. Note that, in a recent NIST mailing list thread, G. Zaverucha & D. Kales offered a new attack which the MQDSS team acknowledged and will address in the future with a new parameter set that will probably worsen our measured performance.

Rainbow [27]: The specification of Rainbow suggests three different variants, and parameter sets. By examining the cryptographic operations' performance, along with the public keys of each alternative, we chose to test the Rainbow Ia - Cyclic parameter set integrated in the PQClean project [73] because it was the most promising one for TLS. Rainbow Ia offers Level 1 security.

Table I summarizes the conventional, and PQ signature algorithms examined in this study and presents the sizes of public keys, private keys, and signatures.

Two NIST Round 2 candidate signature schemes were not included in our study, namely GeMSS and LUOV. The **GeMSS** specification [19] presents 9 parameter sets with large public key sizes that range between 350 and 3090 KB. Those values are too large to be considered for practical use in TLS and would even present functionality limitations (see discussion about Rainbow in Section V-A). The **LUOV** specification [12] presents 6 parameter sets for Round 2 with big public key sizes, accompanied with small signatures and relatively good cryptographic operation performance. This scheme was not considered for our testing for two reasons: (a) it resembles, like GeMSS, Rainbow (small signatures, big public key) which we examined, and (b) a new attack against LUOV and lifted structure schemes was presented at the Second PQC Standardization Conference [28].

IV. Post-Quantum Authentication in TLS 1.3

This section presents details regarding the implementation and integration of PQ authentication into TLS 1.3 with its implications. To integrate PQ signatures into X.509 and TLS we utilized the `OQS OpenSSL` [72] library, which is a fork

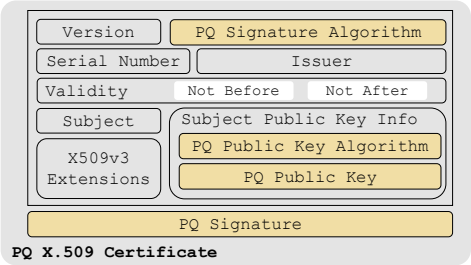


Fig. 1: Post-Quantum X.509 Certificate

of OpenSSL that introduces post-quantum algorithms from the `liboqs` library [71], [88]. The `OQS OpenSSL` version we used was based on `OpenSSL` version `1.1.1c` with ASM optimizations for SHA256, SHA512, RSA and ECDSA256. No ASM optimizations were included for ECDSA384 which we used as reference/comparison in our experiments (see Section V).

Migrating to PQ authentication will require changes in X.509 and SSL libraries; in our case `OpenSSL`. Fig. 1 shows the format of a PQ X.509 certificate and the fields where PQ algorithm support will need to be added. The new certificate will carry the subject's PQ public key and the specific PQ signature algorithm used to create the signature. Then the certificate will be signed by the issuer. The PQ signature will be placed in the `Signature` field. The addition of the PQ public key and PQ signature to the X.509 certificate will increase the size of the certificate and the size of the related certificate chains (Table I. In TLS 1.3, the maximum default size of a X.509 certificate or certificate chain is 2^{24}-1 bytes and the signature size limit is 2^{16}-1 bytes. To include new algorithm support, the `OQS OpenSSL` library defines new X.509 algorithm identifiers [25], [72].

Next, we focus on the TLS 1.3 handshake pieces affected by PQ algorithms. Fig. 2 shows an overview of the messages exchanged by a client who attempts to set up a TLS 1.3 session with a server utilizing quantum-resistant authentication. We assume a classic web scenario where the client is not authenticated. The `ClientHello` message will negotiate the desired PQ signature algorithm using the `signature_algorithms` or `signature_algorithms_cert` extensions. These are essentially lists of algorithm identifiers. The next adjustment for PQ authentication is the PQ X.509 certificate/chain transmission by the server with the `ServerCertificate` message that will now include PQ certificates. In addition, the server will sign the transcripts of the handshake and transmit a PQ `CertificateVerify` message that contains a PQ signature. Currently, the maximum default size of this message is 102.4 KB. Additionally, when a certificate chain exceeds 16 KB, TLS utilizes Record Fragmentation [76]. Finally, the client will will use the PQ signature algorithm to perform verification operations to the received signatures before sending its own `Finished` message to end the PQ TLS 1.3 handshake.

V. Performance Evaluation

In this section we present the performance of each PQ signature scheme after its integration into OQS OpenSSL. In our experiments, we assume a classic web scenario where only the server is authenticated using X.509 certificates. All TLS

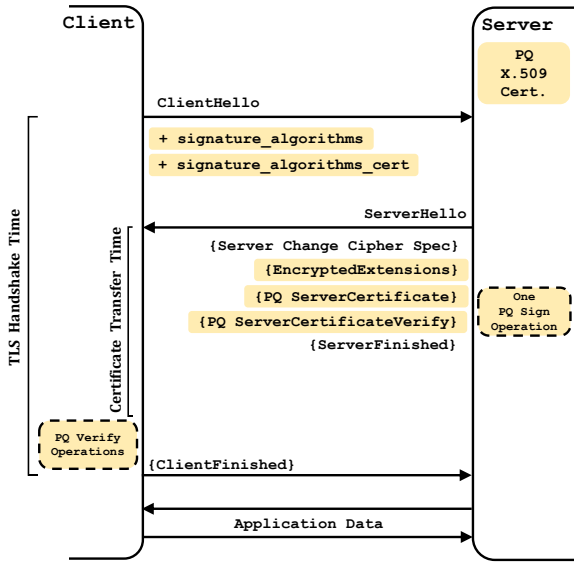


Fig. 2: Post-Quantum TLS 1.3 Handshake

1.3 handshakes are full 1-RTT mode without PSK resumption. In order to compare PQ authentication in TLS against current algorithms, we utilize the RSA and ECDSA as baselines. For RSA authentication, we use the 3072-bit version that provides 128 bits of security, (Table I). Regarding ECDSA, we utilize the `secp384r1` curve which offers 192 bits of security. We note that ECDSA with `secp256r1` or Ed25519 would offer 128-bit security level equivalent to RSA3072 with better performance, but we chose higher security level for ECC signatures because it is believed that these primitives will be broken by a QC before their equivalent security level big number (RSA) signatures. We could have tested against RSA2048 or ECDSA256 instead, but our experiments showed that slightly faster and smaller signatures will negligibly speed up handshakes that take 113ms [61]. All RSA and ECDSA/EdDSA would offer 0 bits of security in a post-quantum setting. Our objective is to document strictly the behaviour of PQ authentication in TLS. Thus, we utilize a classic `X25519` elliptic curve Diffie-Hellman (ECDH) key exchange with Curve25519. The experiments involved one local machine and several cloud-based instances. The local host was equipped with an Intel i5-8350U processor and 16 GBs of RAM, while the experiments were implemented on a virtual machine that utilized four cores running at 1.7 GHz each and 8GB of RAM. Our cloud machines were Google Cloud Platform (GCP) `n1-standard-2` [86] instances running on an Intel Xeon processor with two cores at 2 GHz each and 8 GBs of RAM. All the participating machines were running Ubuntu 18.04 in an `x86_64` architecture. TCP Segmentation Offload (TSO) and Large Receive Offload (LRO) were enabled on the hosts' virtual network cards.

Readers should note that in our evaluation we do not consider the impact of PQ signatures employed in CRL and OCSP revocation checks which would take place over a different connection. In addition, potential use of OCSP staples would introduce another signature generation and verification. Staples can be cached which would not affect the handshake until the staple expired. That falls outside the scope of our work. Similarly, we do not consider SCT checking which requires

Signature Algorithm	Local Machine (ms)				Cloud Instance (ms)			
	Sign		Verify		Sign		Verify	
	Mean	St. Dev.	Mean	St. Dev.	Mean	St. Dev.	Mean	St. Dev.
RSA 3072	3.19	0.023	0.06	0.001	2.39	0.010	0.04	0.002
ECDSA 384	1.32	0.012	1.05	0.020	1.28	0.015	0.93	0.004
Dilithium II	0.82	0.021	0.16	0.005	0.41	0.018	0.12	0.008
Falcon 512	5.22	0.054	0.05	0.004	6.50	0.091	0.07	0.003
MQDSS 48	10.30	0.147	7.25	0.100	10.25	0.181	7.40	0.110
Picnic $L1FS$	4.09	0.050	3.25	0.049	3.17	0.051	2.39	0.044
SPHINCS$^+$ SHA256-128f-simple	93.37	0.654	3.92	0.043	62.7	0.548	2.50	0.037
Rainbow Ia	0.34	0.015	0.83	0.036	0.25	0.020	0.48	0.044
Dilithium IV	1.25	0.021	0.30	0.012	0.46	0.019	0.23	0.010
Falcon 1024	11.37	0.102	0.11	0.005	14.20	0.156	0.14	0.005

TABLE II: Duration of Sign and Verify Operations: Mean and Standard Deviation

more signature verifications and TLS data. PQ revocation and SCTs could slow down the handshake, as they do already, but there are techniques to alleviate their impact which we discuss in Section VII-B.

Finally,we use the following metrics to evaluate the TLS performance from the server and client's perspective:

- *TLS Handshake Time*: The time between the `ClientHello` message and the end of the handshake. This metric models the latency a client experiences before establishing a secure encryption tunnel excluding the TCP handshake.
- *Certificate Transfer Time*: The time between the `ServerHello` message and the `Client Change Cipher Spec` message that includes the transfer of certificate chains from the server to the client.
- *TLS Transactions per Second*: This metric expresses the rate of TLS session establishment from the server's perspective.

A. Speed of Cryptographic Operations

First, we evaluate the performance of cryptographic operations for the signature algorithms used in this study. Sign and verify operations are significant differentiation points for the PQ algorithms since they are performed by the server and the client respectively in the TLS handshake. The measurements for the classic algorithms (RSA, ECDSA) were taken using the `OpenSSL speed` command, while for the PQ schemes we used the testing scripts from `liboqs` [71], [88]. Table II shows average sign and verification times of the various schemes run on the machines of this study. Since the duration of these operation directly affects the experiment's TL handshake time metric, we measure the algorithms' standalone performance in milliseconds for each type of machine used instead of CPU clock cycles which is the common practice. We observe that Dilithium, and Rainbow offer competitive performance for 128 and 192-bit security comparable to current algorithms (RSA3072 and ECDSA384). Falcon offers great verification times, and slower signing. We can see that only Falcon is slower in the cloud machine. We attribute this behavior to Falcon's floating-point operations which do not exist in other PQ signature schemes. MQDSS and Picnic are slower and SPHINCS$^+$ is significantly slower, especially for sign operations. Our signing and verification measured performance is generally confirmed by the SUPERCOP benchmarks [9] and [81].

Signature Algorithm	Cert Chain Size (KB)		`CertificateVerify` Size (KB)
	One ICA	Two ICAs	
RSA 3072	1.63	2.44	0.38
ECDSA 384	1.34	2.15	0.05
Dilithium II	6.90	10.42	2.04
Falcon 512	3.54	5.37	0.69
MQDSS 48	42.24	63.42	20.85
Picnic $L1FS$	66.20	99.57	30.03
SPHINCS$^+$ SHA256-128f-si	34.46	51.74	16.98
Rainbow Ia	116.86	175.35	0.06
Dilithium IV	10.70	16.11	3.37
Falcon 1024	6.56	9.89	1.33

TABLE III: Sizes of certificate chains (excluding the root), and `CertificateVerify` extension

NIST Round 2 submissions included optimized versions of the algorithms that take advantage of parallelization and hardware acceleration (e.g., AVX2). These optimizations speed up performance of the algorithms significantly. The implementations we used in our experiments (Table II) were not optimized. Using the optimized versions would make signing and verification greatly faster. For example, AVX2 optimized Falcon signs $\sim$20 times faster and AVX2 optimized Dilithium IV signs faster than Dilithium II without AVX2. We expect that future hardware acceleration of the algorithms themselves would improve their performance further and eventually the TLS cert chain and `CertificateVerify` size will be the bottleneck of PQ authentication.

B. PQ TLS Overhead Analysis - NIST Security Level 1

At the first phase of our experiments we studied the performance of PQ schemes in NIST's Level 1. The experimental setup consisted of our local machine operating as a client in N. Carolina and a remote machine from GCP's N. Virginia region acting as the server. Two chain lengths were considered in our experiments as shown in Fig. 5. The first involves a single ICA and the second involves two ICAs since these account for 77%

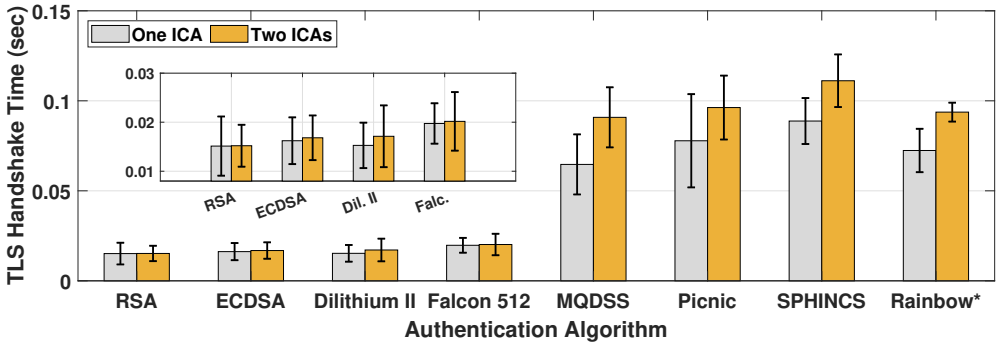


Fig. 3: TLS Handshake Time for NIST's PQ Security Level 1 Signature Algorithms: Average and Standard Deviation

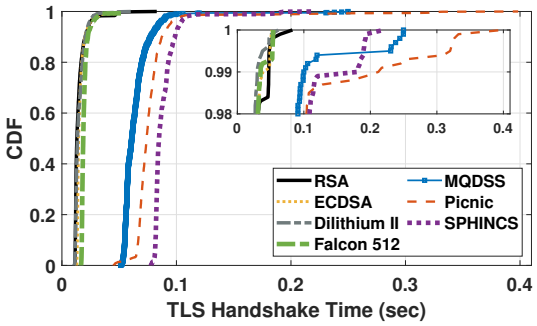


Fig. 4: TLS Handshake Time for NIST's PQ Security Level 1 Signature Algorithms: Empirical CDF - One ICA

of the SSL cert chains on the Internet today [84]. The same PQ signature scheme was used for all certificates across each chain. Table III summarizes the exact Distinguished Encoding Rules (DER) encoded certificate chain sizes excluding the root CA cert which is not sent to the client, and the TLS `CertificateVerify` extension size.

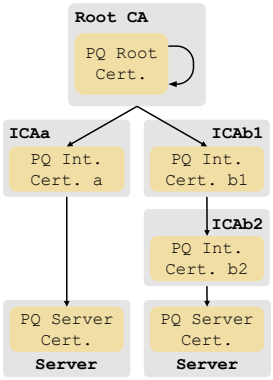


Fig. 5: X.509 certificate chains in our experiments

Current WebPKI certificates include extensions (SCT, OCSP, CRL, SAN) that we did not use in our experiments. These fields sometimes lead to conventional certificates that exceed 1.5KB. Our tests included less extensions resulting to slightly smaller sizes than common web certificates. Based on the data we collected, we do not expect that one or two hundreds of bytes of certificate extensions would effectively change our results unless they push the certificate chain over the TCP congestion window (`cwnd`). PQ SCT and OCSP signatures would add significant data to the handshake. We further discuss this point in Section VII-B.

In our performance experiments, we first measure the average TLS handshake time over 1000 handshake attempts. Fig. 3 shows these results for the PQ algorithms in NIST Level 1 for the two certificate chain lengths considered. During our experiments, all TLS records were received successfully and no errors were observed due to the excessive size of

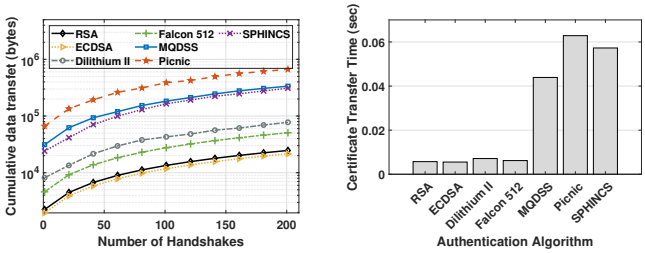


Fig. 6: Impact of PQ Certificate Transfer NIST's PQ Security Level 1 Signature Algorithms (one ICA): (a) Cumulative Transfer Overhead, (b) Average Transfer Time

the certificates for six out of the seven algorithms examined. More specifically, the SSL client application was not able to process the size of a Rainbow certificate yielding an `excessive message size` error. After examining the trace, we observed that the client was issuing an `SSL Alert` because of the certificate public key size, however, the server transmitted the chain before closing the connection. Since this happened consistently for all handshake attempts, Fig. 3 shows the duration of this exchange (the * denotes partial handshake without a `ClientFinish` message).

Fig. 4 shows the empirical cumulative density function of the single ICA measurements. The long tails observed are attributed to the network transmission path. It is evident from the results that long certificates and slow operations significantly reduce the authentication performance in TLS for algorithms like MQDSS, Picnic, and SPHINCS$^+$. Fig. 6(a) shows the cumulative data transfer between the `ServerHello` and the `Client Change Cipher Spec` message for 200 consecutive handshakes providing a insight on the network traffic generated when implementing PQ certs in TLS. Fig. 6(b) shows the average time elapsed during this period. As expected, algorithms with bigger public keys and signatures lead to longer time to transfer the certificates and TLS `CertificateVerify`. For small signatures (i.e., Dilithium II, Falcon 512) that equates to less than 5ms extra whereas for bigger ones (i.e., MQDSS, Picnic, SPHINCS$^+$) it becomes ∼35-55ms extra.

The client and server were in close proximity in this experiment. The transfer times of lengthy signatures and chains could exceed 0.5s for longer client-server distances. That is because algorithms with excessively large certificate sizes and signature messages are penalized by the TCP congestion window. Certificate chains and `CertificateVerify` payloads of high size (i.e. MQDSS, Picnic, and SPHINCS$^+$ from Table

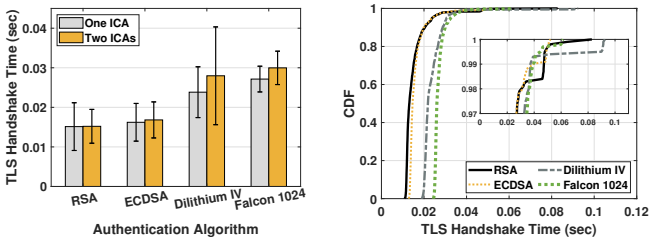


Fig. 7: TLS Handshake Time for Higher PQ Security Level Signature Algorithms: (a) Average and Standard Deviation, (b) Empirical CDF - One ICA

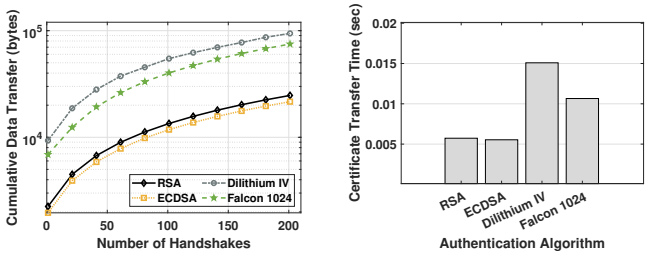


Fig. 8: Impact of PQ Certificate Transfer for Higher PQ Security Level Signature Algorithms (one ICA): (a) Cumulative Transfer Overhead, (b) Average Transfer Time

III) incur round-trip(s) which significantly slows down the handshake due to the TCP congestion window (`cwnd`). Falcon 512 and Dilithium II, on the other hand, do not incur extra round-trips and the additional cert transfer time would remain ∼5ms even over longer distances.

In Fig. 3, algorithms that result to smaller certificate chains and `CertificateVerify` payloads (i.e., Dilithium, Falcon from Table III) show competitive performance against conventional signature algorithms. Falcon 512 has smaller signatures and public key, but we see it performing worse than Dilithium II because its signing operation at the server takes longer (∼6.5ms without AVX2 optimizations). Signing duration of 6.5ms and chain/TLS signature transfer of ∼3ms are more noticeable in a localized scenario (e.g., a server in N. Virginia) with a baseline RSA3072 handshake time of ∼15ms. Falcon 512 would look better for the average Internet web connection of ∼110ms [61] as we show in Section V-E.

C. PQ TLS Overhead Analysis - NIST Security Levels 3, 5

Next, we extend our experiments to measure the performance of algorithms that belong to higher security levels in NIST's level scale. For the majority of the examined PQ algorithms, the parameter sets that provide higher quantum-resistant authentication result to significantly larger public keys and signatures [3], [5], [21], [27], [31], [34], [38] and worse performance. Due to these sizes, the certificate chain and `CertificateVerify` message were shown in Section V-B to exceed the TCP `cwnd` which causes significant slowdowns because of the extra round-trip. Moreover, in a recent NIST mailing list thread, G. Zaverucha & D. Kales offered a new attack which the MQDSS team acknowledged and will address with a new parameter set which is likely to have even worse performance.

Thus, for Levels 3 and 5, we limit our attention to the two PQ algorithms that presented the most promising results at Level 1, namely Dilithium II and Falcon 512. We examine Dilithium IV at NIST security Level 3, and Falcon 1024 at Level 5. Currently, Dilithium does not offer a parameter set at Level 5, while the Falcon's Level 3 parameter set (Falcon 768) was removed in Round 2 [34].

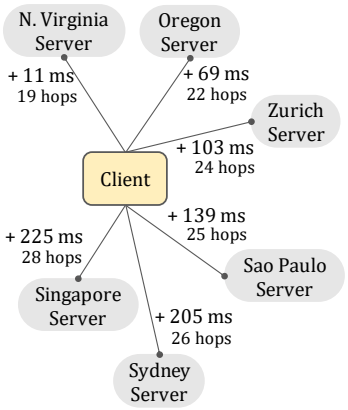
Again, we measure the average TLS handshake time over 1000 handshakes for our client in N. Carolina and a server in N. Virginia. Fig. 7(a) shows these results for the examined algorithms for the two certificate chain lengths, while Fig. 7(b) shows the empirical cumulative density function of the single ICA measurements. We observe that by increasing the security level of the PQ schemes the absolute added latency to the TLS handshake is on average 7.6-12.1ms over RSA3072. Falcon 1024 would look almost as good as Falcon 512 for the average Internet web connection of ∼110ms [61] as we show in Section V-E. Fig. 8(a) shows the cumulative data transfer between the `ServerHello` and the `Client Change Cipher Spec` message for 200 consecutive handshakes. Fig. 8(b) shows the average time elapsed during this period which is an additional ∼5ms for Falcon 1024. The Dilithium IV cert transfer is slower because the data size triggers an extra round-trip (∼11ms). Falcon 1024 does not add an extra round-trip which means that the additional cert transfer time would remain ∼5ms even over longer distances, but it adds ∼14ms of signing. An additional 14ms is more noticeable in a localized scenario (N. Carolina to N. Virginia) with a baseline RSA3072 handshake time of ∼15ms.

D. Combining PQ Signature Schemes

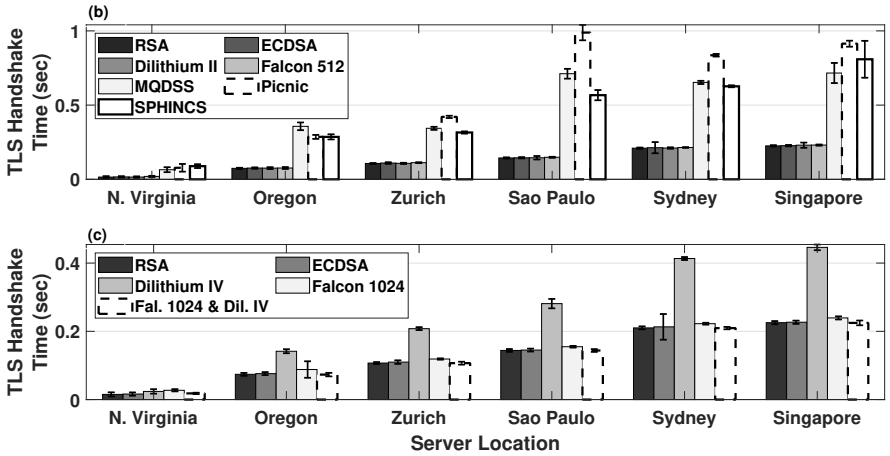
The diverse nature of the cryptographic primitives used for each PQ signature algorithm leads to solutions with different characteristics and behaviour, namely different signature/key sizes and computational complexity (Tables I, II). Therefore, by using different PQ signature schemes in the same certificate chain, we can leverage each algorithm's specific strengths and effectively reduce the overall TLS handshake time. To the best of our knowledge, this is the first time where using different algorithms along the certificate chain has been proposed to improve total handshake performance. To illustrate this intuition, we examine a proof-of-concept combination of the Falcon 1024, and Dilithium IV algorithms across a chain with one ICA. The ICA cert includes a Falcon 1024 public key and signature from the root CA, while the server cert utilizes a Dilithium IV public key and a Falcon 1024 signature from the ICA. By doing so, we reduce the duration of the server's signing operation by approximately 30 times. In addition, the overall certificate chain size is equal to 6.52 KB, which is smaller than both the pure Falcon 1024 chain (6.56 KB), and the pure Dilithium IV chain (10.70 KB) (Table III). Table IV shows a comparison yielded following the experimental procedures of Section V-C

Signature Scheme	TLS Handshake (ms)	
	Mean	St. Dev.
RSA 3072	15.13	6.03
Dilithium *IV*	24.20	2.62
Falcon 1024	27.14	3.30
Fal. 1024 & Dil. *IV*	18.11	1.58

TABLE IV: Average Handshake Time and σ of the Dilithium and Falcon combination (single ICA).



(a) Round-trip time and hops between client and servers



(b) NIST Level 1, (c) NIST Levels 3, 5

Fig. 9: TLS Handshake Time at a global scale

(client-server round-trip ∼11ms). The combined case reduces (on average) the overall handshake by 25.16% compared to the single Dilithium IV, and by 33.27% when put against the single Falcon 1024 case. The combination can work with other signature schemes too as discussed in Section VII-B, while it is especially suitable for the cases where (a) the client and server are in close proximity, (b) floating point hardware is not available at the signer, or (c) when leaf certificates are renewed relatively often and lower security levels are acceptable. The drawback is that the verifier is required to support both signature algorithms. The performance improvement of this method will be marginal for long client-server distances. The client and server Section V-E and V-F further examine this proposed alternative of combining schemes.

E. Global Scale Performance Analysis

Next, we extend our evaluation by adding more remote servers across different continents. Fig. 9(a) shows the experimental setup, the average number of hops, and the measured average round-trip latency between our local client and each remote server. Again, for each PQ signature algorithm and parameter set, 1000 handshake attempts were performed and their duration was averaged. A certificate chain with a single ICA was utilized. Fig. 9(b) shows the mean and standard deviation of the TLS handshake time when algorithms with NIST's security Level 1 were tested, while Fig. 9(c) does the same for Level 3 and 5. The performance was similar across all regions, and no handshake failures were reported due to middlebox misbehaviour. Consistently, Dilithium and Falcon show the most promise for replacing RSA3072 with minimum additional delay.

Using the same setup, in a second experiment, our client uniformly performed 3000 handshakes with the remote servers in the course of a whole day for each examined algorithm. The goal was to account for Internet's unpredictability by measuring TLS handshake latency in diverse topologies across longer time-frames. Table V summarizes the additional latency introduced to the TLS handshake by the PQ signature algorithms when compared against RSA3072 at the 50^{th} and 95^{th} percentile. The observed latency overhead of Level 1 Dilithium II was notably small making it the most pertinent for integration into TLS. Level 1 Falcon 512 also showed small overhead over RSA3072. Regarding higher security levels, Dilithium IV nearly doubled the handshake duration because the data size triggered an extra round-trip, while Falcon performed better. Although the % increase seems high for Dilithium IV, the absolute time increase is ∼145ms at the 50^{th} percentile and ∼229ms at the 95^{th} percentile. The Dilithium+Falcon combination (Section V-D) reduced the handshake latency by ∼8-50% in comparison to single PQ algorithm certificate chains.

Signature Algorithm	Handshake (ms)		Latency (%)	
	50^{th}	95^{th}	50^{th}	95^{th}
RSA3072	131.54	227.26	0	0
Dilithium II	140.20	232.51	6.58	2.31
Falcon 512	142.22	235.46	8.12	3.49
MQDSS 48	598.61	726.20	355.05	219.53
Picnic $L1FS$	634.90	985.88	382.63	333.79
SPHINCS$^+$ SHA256-128f-simple	553.15	904.98	320.49	298.19
Dilithium IV	276.55	449.88	110.22	97.95
Falcon 1024	152.96	240.74	16.28	5.93
Fal. 1024 & Dil. IV	140.74	228.42	6.98	0.50

TABLE V: TLS Handshake Times and Additional Latency over RSA at the 50^{th} and 95^{th} Percentile.

We have measured the time between the `ClientHello` message and the end of the handshake in our experiments so far. Extending the period under examination to include the TCP handshake (start with the `TCP SYN` message) increases the mean TLS handshake with `X25519` key exchange and RSA3072 certificates duration to 269.5ms. The respective duration measured in Firefox client-server connections is ∼113ms considering 3.48 billion data points from their nightly 70 version [61]. By extrapolation we estimate that the average PQ TLS handshake measured from the `TCP SYN` to `Ready for HTTP` will amount on average to ∼122-135ms for the best algorithm cases of each security level, namely Dilithium II (level 1), and Falcon 1024 (level 5).

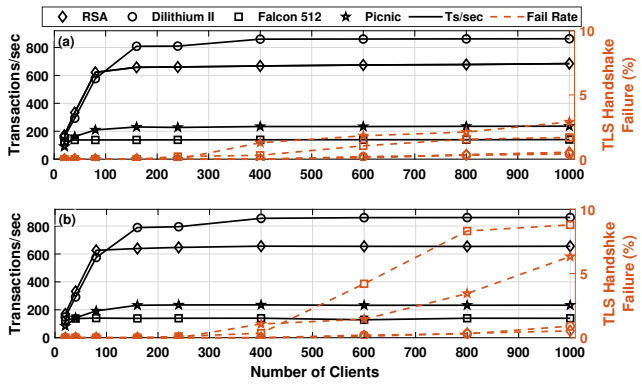


Fig. 11: Server Performance for NIST's PQ Security Level 1 Signature Algorithms: (a) One ICA, (b) Two ICAs

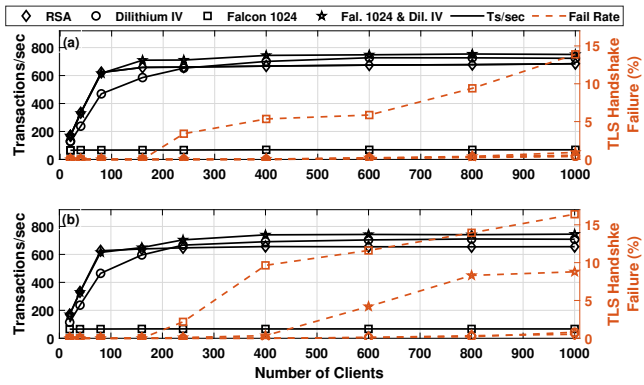


Fig. 12: Server Performance for NIST's Higher PQ Security Level Signature Algorithms: (a) One ICA, (b) Two ICAs

The PQ signing and verification times were relatively small (< 15ms) in our analysis so far for all tested algorithms except SPHINCS$^+$. Such times have relatively little impact on an average handshake time of 113ms [61] or 269.5ms in our tests. Arguably using optimized or hardware accelerated versions of these algorithms would speed up the handshake, but not by a lot because the certificate and signature data will still incur a significant handshake slowdown.

F. Server-Side Performance of TLS PQ Authentication

Up to this point, we focused on measuring the impact of PQ authentication in TLS from the client's perspective. Next, we measure the performance of a server that utilizes PQ certificates in TLS 1.3 to service simultaneous secure connections with multiple clients. To do so, an Nginx web server [64] was set up on one of the aforementioned cloud instances, and was configured to utilize the OQS OpenSSL library. Thus, our server was able to sign handshakes and send PQ certificates using the signature schemes under study. We used the Siege tool [35], configured with the OQS OpenSSL library, to simulate multiple clients and set up PQ authenticated TLS connections. Siege clients were running in four remote cloud instances. For this experiment, all client instances were placed in relatively close proximity to each other. The exact topology details are shown in Fig. 10.

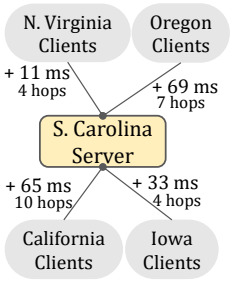


Fig. 10: Round-trip latency and number of hops between clients and server

The clients that were uniformly distributed among the four locations, attempted simultaneous TLS connections with the server for 60 seconds. During that time we measured the request rate that the server was able to handle, i.e. the number of successful transactions per second. The requested web page was only 0.6KB. In addition, we captured the server's overall availability by measuring the number of TLS handshake failures during the load testing period. Fig. 11 shows these results for promising signature algorithms of NIST's Level 1 as the total number of clients increases from 20 to 1000. Both cases of one and two ICAs are examined. Evidently, while still at low load RSA3072 outperforms Dilithium II and Falcon 512 as expected from Fig. 3. However, as the server's saturation point is reached we observe that Dilithium II allows the server to handle $\sim$25% more connections per second compared to RSA3072 because Dilithium signing is over five times faster compared to RSA3072 at the server (Table II). Regarding Falcon 512, the server's saturation point is reached early at a much lower transaction rate than RSA and Dilithium due to slow signing at the server (Table II).

Fig. 3 showed that the Dilithium bigger cert chain and `CertificateVerify` size do not affect the server performance other than slightly slowing down the handshake. To further reinforce that point, the server's transaction rate is similar for both cert chain lengths (Fig. 11(a) vs. Fig. 11(b)) for the same algorithms which shows that just a few additional KB of certificate payload do not affect the server performance.

To further demonstrate the higher impact of signing over cert chain size on a busy server we tested Picnic L1FS which presents much bigger certificate chain and `CertificateVerify` message (Table III) while maintaining a better than Falcon 512 sign operation (Table II). Fig. 11 shows that the server signing with Picnic L1FS outperforms signing with Falcon 512 by 1.7-fold higher transaction rate. That can be attributed to the sign operation performance. Moreover, and in accordance with our earlier experiments, Picnic still leads to a much lower maximum transaction rate compared to RSA3072 and Dilithium II due to its excessively big cert chains that slow down the handshake significantly (Fig. 3). Another interesting observation is that before reaching the server saturation point, Picnic's transaction rate is half than RSA3072. The reason is that up to this point the server has not gotten overloaded and the transaction slowdown solely comes from the TLS performance slowdown seen in Fig. 3.

The same behaviour is observed for parameter sets that yield higher NIST security levels as seen in Fig. 12. The server's saturation point is reached significantly early with Falcon 1024 that also presents the highest handshake failure rate among alternatives because of its signing performance. Also, under high load, the use of Dilithium IV results to higher transaction rates in comparison to RSA3072 and Falcon 1024 because of its better signing performance (Table II). Moreover, we examined the proposed PQ combined certificate chain scenario (Dilithium and Falcon in Section V-D). Results

demonstrate that by using PQ combined certificate chains the server is able to handle more TLS connection requests than all alternatives including the conventional RSA3072. More specifically, the Dilithium+Falcon combination slightly increases the average post-saturation transaction rate by ∼10% and ∼4% compared to RSA3072 and pure Dilithium IV respectively. The improvement over RSA3072 is because Dilithium IV operations are still faster. However, the difference is smaller than in Fig. 11 because the Dilithium IV chain slows down the handshake more (Fig. 7). The server's performance increase with Dilithium+Falcon over Falcon 1024 is due to the computationally lighter signing of Dilithium IV (Table II).

In summary, we saw that signing at the server is an important factor for the server's performance. Cert chain and signature size affect the server transaction rate by increasing the TLS handshake time but the effect is smaller than the impact of a heavy signing operation. We expect that the server performance penalty introduced by expensive sign operations will be alleviated by optimized implementations and hardware acceleration in the future.

Finally, it is worth noting that in our tests we observed that the client closest to the server saw a higher transaction rate due to the lower propagation delay. Thus, servers closely located to clients will see better performance than the average we are showing.

VI. Related Work

A large body of research on QC and PQ cryptography [11], [44], [49], [93] led to the NIST initiative to come up with novel PQ algorithms for use in a post-quantum world. Recently, more works are exploring NIST's Round 2 PQ candidate schemes focusing mainly on their computational performance, while some even evaluate their energy consumption [80]. The authors in [6], [87] conduct a hardware implementation and comparison of NIST Round 2 signature and KEM candidate algorithms. Their focus is on comparing specific algorithm operations and their impact on hardware design parameters, without taking into account performance as a part of a system or a protocol. In [47], Kannwischer et al. benchmark Round 2 algorithms on ARM Cortex-M4 and evaluate the more suitable ones for embedded devices without studying additional impact on specific use-cases. Similarly, the authors in [83] enhance the DTLS protocol with the PQ key exchange algorithm NTRU which showcases its feasibility for securing current systems.

In addition, research teams from the industry are also investigating the performance of PQ candidate algorithms in current Internet protocols. They have mostly been focusing on studying key exchange. In [15], Bos et al. propose the use RLWE for key exchange in TLS 1.2 and study its impact that slows down performance by a factor of ×1.4 for the client and ×1.2 for the server. A year later, in 2016, Google tested a hybrid KEM (New Hope combined with elliptic-curve schemes) and ECDSA certs with TLS 1.2 in Google Chrome [53]. Following the conclusion of NIST's Round 1 evaluation, Google and Cloudflare resumed the experiments on KEM integration into TLS. The new study [55] defined three KEM families based on the primitive and the key size and tested them with dummy extensions of proportional sizes in the TLS `ClientHello` message. The goal was to emulate the overhead that these key sizes introduce and explore issues or failures related to such larger handshakes. Recently, Google and Cloudflare have began work on a new experiments (CECPQ2 [54]) that integrate a hybrid KEM based on the HRSS NIST PQ candidate and X25519 into TLS 1.3 for further performance measurements on Chrome. Additionally, they were testing the SIKE NIST PQ candidate with X25519 in TLS 1.3 as part of their CECPQ2b experiment [51]. The results of this work are included in [52]. Moreover, Campagna recently discussed hybrid key exchange and double tunnelling for TLS and presented the significant slowdown of TLS 1.2 introduced by using hybrid key exchange with ECDHE and SIKE or BIKE NIST PQ candidates [18]. Campagna et al. showed that SIKE's impact on the handshake byte count is small, whereas BIKE's impact is significant.

More closely related to our work that focuses on signature performance, Cronin et al. [26] modeled and benchmarked the performance of forward-secure signatures against conventional RSA, DSA and ECDSA. Focusing more on PQ signatures, Kampanakis et al. discuss the impact of PQ signature schemes on protocols that utilize X.509 certificates in [46]. They consider the case of hybrid certificates that include hash-based signatures, and conduct performance experiments on TLS 1.2 and IKEv2. Also, their work discussed the use of PQ certificates by emulating their increased size through enlarging certificate chains with additional certificates. On the same topic, earlier work by Bindel et al. emulated large hybrid PQ certificates and studied their impact on TLS libraries and browsers [13]. The authors in [17] prototyped stateful XMSS signatures in TLS and S/MIME and pointed out the challenges with state management in live TLS connections. In [25], the authors discuss the challenges of implementing NIST's PQ key exchange and authentication algorithms in TLS, with their focus being mainly on hybrid (combination of PQ and conventional) schemes. In addition, they perform proof-of-concept experiments on PQ TLS that involve single certificate exchanges, without accounting for real-world network conditions. Finally, Paquin et al. [67] perform experiments with some hybrid PQ KEM and Signature schemes in TLS and show that lossy conditions have more negative effects on data-heavy TLS handshakes. The loss rates they use are not very common on the Internet today, but they could be experienced in constrained or congested environments, or remote cellular networks.

VII. Discussion about encrypted tunnels

A. Implications of our Findings

PQ signatures will have an impact on authenticated tunnels like (D)TLS and IKEv2/IPSec. These protocols provide fragmentation mechanisms to allow for lengthy signatures, but as we showed above, larger chains and potentially slower algorithms will impact the tunnel establishment. Applications with lower connection rates and tunnels that stay up longer will be less impacted than applications that establish short and fast connections. Thus, per connection overhead is important to be considered for the migration to PQ authentication algorithms. In addition, the more signatures used in a handshake, the more impact PQ algorithms would have on the protocol. For example, SCTs in leaf certificates will add two or three

signatures to the handshake and OCSP stapling would add one more.

IPSec VPN tunnels usually stay on for long periods of time. Experimental RFC4478 [65] defines how a tunnel can be re-authenticated, but in practice, `IKE_AUTH` messages are not exchanged unless the tunnel is torn down and re-established. Thus, for the majority of IKev2/IPSec VPN applications, a connection establishment that would take up to a few seconds would not have material impact on the tunnel. Similarly, WebVPN applications establish a TLS control connection and subsequently data DTLS connections over a duration of a few seconds. Onward, these tunnels stay up for long sessions that usually last hours or more. A connection establishment slowdown of up to a few seconds will not impact these connections. The caveat with such use-cases is the impact on the head-end that terminates these connections. It is not unusual for a VPN concentrator to terminate thousands of connections. A possible attempt of VPN clients to establish a connection to the hub simultaneously could lead to overload if the signing or verification operations are heavy. Most PQ signature candidates we studied (except SPHINCS$^+$) are not likely to be a significant performance concern for VPN tunnels that take advantage of algorithm optimizations.

Web connections, on the other hand, are usually short-lived. At the time of this writing, clients perform 70 requests per page to fetch 1-2KB resources per request on average [42]. If some of these requests are performed over new TLS connections, a heavier PQ authenticated TLS handshake would have a significant impact on HTTP performance overall. A few KB of extra authentication data per TLS connection has low amortization over 1-2KB of actual web content. On the other hand, the ever-increasing adoption of HTTP/2 [7] (55% at the time of this writing [42]) will improve amortization as HTTP/2 multiplexes data over a single HTTP connection. It is hard to say with certainty how much delay is excessive on the web. Reports like [1] lead us to believe that hundreds of extra milliseconds per handshake are not acceptable for the web. More importantly, extra round-trips mean almost doubling total handshake times which we consider excessive for time-sensitive web applications.

In summary of our testing, we found that Dilithium II, Falcon 512 and 1024 could be deployed in X.509 certs without detrimentally impacting time-sensitive TLS applications. Dilithium is preferable over Falcon, at least when floating point hardware is not available, because its superior signing performance allows for higher connection rates. Floating point hardware improves Falcon's signing performance by $\sim$20 times [70] which would make Falcon a better candidate than Dilithium in terms of performance in live protocols. Falcon would also have more impact on energy constrained devices [81]. Dilithium IV nearly doubled the handshake duration over RSA3072 because the data size triggers an extra round-trip. Dilithium IV could still be used if applications were amenable to approximately double the TLS handshake time or if some of the mechanisms in Section VII-B (e.g., [90], [77, § 5.1.3]) were widely deployed. We also showed that other NIST PQ signature candidates would not be good candidates for such applications.

Dilithium II and Falcon 512's claimed classical security levels are $\sim$100 and $\sim$114 bits respectively. Although these asymptotic bounds are probably higher in practice because of memory costs, these parameter sets could be considered unacceptable to use. The variants that offer more than 128 bits of classical security are Dilithium III, IV and Falcon 1024. We already discussed that Falcon 1024 had acceptable handshake time with the shortcomings of higher energy cost and slow signing when floating point hardware is not present and that Dilithium IV was introducing extra round-trips. A rerun of our experiments in Section V-E showed that the use of **Dilithium III** certificates added 2.7KB extra to the handshake (compared to Dilithium II) with fast signing/verification times. That led Dililthium III to perform $\sim$7% slower than RSA3072 which is very similar to Dilithium II. Based on the above, if Dilithium II and Falcon 512 security levels are considered low, Dilithium III and Falcon 1024 are the best options. Falcon 1024 would be preferable when floating point hardware is available at the signer.

None of our tests included OCSP revocation checks and SCTs returned from the server. OCSP and SCTs could account for three or more PQ signatures which could push the server TLS data over the TCP `initcwd` and introduce round-trips for all Dilithium variants. Falcon data could fit in the `initcwd` at the cost of one slower OCSP singature assuming floating point hardware is not available. Falcon would be preferable specifically for SCTs because of its relatively fast verification and small signature. Readers should note that on the web, OCSP responders or staples are usually not used (unless the must-staple extension is present in the server cert) by most browsers and they are not supported by most servers. Thus, we consider OCSP signatures as a minor challenge for the PQ WebPKI case.

B. Minor Adjustments to Enable PQ Signatures

There are straightforward changes needed for TLS and IKEv2 to support PQ signatures. The TLS extensions `signature_algorithms_cert` and `signature_algorithms` will need to use new identifiers to convey to the TLS peer the PQ signature algorithms supported [25, § 4.1.1]. Also, X.509 will need to use new Public Key and Signature identifiers to convey the new algorithms [25, § 4.1.1], [46, § 1]. In the context of IKEv2, signature negotiation was not included in the protocol from the beginning. Recently RFC7427 [48] added support for signature algorithm negotiation. RFC7427 is not widely supported by VPN vendors, so in a PQ authenticated IKEv2 case, vendors will have to support all possible algorithms and not negotiate with the peer, or support RFC7427 and negotiate the PQ signature algorithms with new identifiers.

Crockett et. al. describe in [25, § 4.3.1] the challenges introduced by some algorithms with signatures and cert chains that exceed 2^{14} B and 102.4 KB respectively. Out of the tests we performed, only Rainbow fell in these categories and led to failed handshakes. We do not expect schemes with such big signatures to be used in TLS, thus TLS and protocol implementations will likely not require further updates to accommodate them.

On the other hand, the work in [90] proposes a new TLS extension to inform the server that the client does not need the ICA certificates in the certificate chain. The authors in

[77, § 5.1.3] proposes a similar method of omitting certificates from a handshake by using a pre-established certificate dictionary. Such mechanisms could save significant amounts of data from handshakes that approach or exceed the TCP `initcwnd` to prevent extra round-tripssimilar to Dilithium IV. They would also alleviate the impact of lossy networks which was shown to be high for data-heavy TLS handshakes. The way to implement the mechanism would be either for the client to keep track of ICAs, or maintain an ICA cache and a server leaf certificate Common Name (CN) cache. Both caches would get updated from a new TLS handshake when there is a cache miss. The server certificate itself would not be cached. It would still be returned from the server with every handshake. The CN cache could save additional data overhead from OCSP and SCT PQ signatures as well. Valid OCSP staples and SCTs could be bound to a CN cache entry. As long as the client had an entry in the cache for the server leaf certificate, he would not need to receive the ICAs, OCSP or SCT information. One challenge would then be that, at the time of this writing, SCTs are most usually included in the leaf certificate. To omit the SCT data, the certificate could no longer include the SCT information which would now be carried in a `signed_certificate_timestamp` extension [56]. That is a drastic paradigm change but it would allow the client to limit the TLS handshake data if the information is in its cache.

Section V-D showed how combining an efficient algorithm could improve the performance of the handshake. Using the same rationale at the root CA, the algorithm combination with relatively small signature size could shrink the ICA certificate size and speed up the handshake. Such schemes could be multivariate candidates like LUOV, GeMSS, and Rainbow or, qTesla, or Stateful HBS [43], [58] with small tree heights. Certificate chains that include different signature algorithms impose an extra requirement for a client to support multiple PQ signature algorithms.

C. Drastic changes to Enable PQ Authentication

CPU-intensive algorithms like Falcon signing could add significant load to busy servers. Such cases could benefit from batch signing options proposed in [8]. Batch signing would allow the server to sign batches of handshakes with one signature, but it would require broad client upgrades and could introduce delays while the server is buffering handshakes to batch-sign them.

Some lattice signature schemes like Falcon could offer message recovery as part of the signature [69]. The signature size doubles, but if the message is longer than a plain Falcon signature then message recovery shortens the $message$+sig size. This mechanism could be integrated in X.509 to make the PQ certificate smaller. An X.509 certificate could then only include the algorithm identifier and the signature. The message (`tbsCertificate`) would be recovered from the signature itself. Such methods would require significant changes in X.509 standards and their implementations in tunnelling protocols.

If Dilithium and Falcon are not standardized, tunnelling protocols will be significantly affected by the rest of the PQ signature algorithms. Given that the industry is constantly striving for faster handshakes and better performance [36], [90], [77] it is unlikely that the impact of the PQ signature algorithms (excluding Dilithium and Falcon) will be acceptable. In that case, more drastic protocol changes may be necessary.

One thought proposed in public fora is to follow the paradigm by Certificate Transparency [56] and offer a public repository or service that could be used out-of-band to retrieve certificates of the entities that a client would communicate with. To offer the same functionality, the work in [78] uses special DNS Resource Records to serve public keys used for SNI encryption. Such proposals would eliminate the need for certificates to be transferred in the handshake, but it comes with a requirement of retrieving and verifying the peer certificate out-of-band. Unless caching is enforced, this certificate retrieval step would need to be repeated before any TLS connection establishment which of course is unlikely to improve the overall performance. If caching the peer certificate is possible, it comes with concerns [82, § 7], [46, § 2.1]. We consider such a mechanism a drastic paradigm shift from the way the Internet works today.

On the other hand, the authors in [14], [79], [10, § 2.3], propose an alternative authentication method that uses PQ Key Encapsulation which could be more efficient than PQ signatures. The mechanism would require for the server leaf certificate to include a PQ KEM public key which can be relatively short. But it would also require new protocol message extensions and an extra round-trip in TLS because the signature ciphertext can only be generated after the client has retrieved the server's public key. It would work similarly for IKEv2. Changing the TLS or IKEv2 state machine in order to prevent the extra round-trip would not be trivial as it would affect current security analyses of the protocols [23], [24]. Moreover, the public key would still need to tied to the peer identity using PKI so the PQ cert chain containing PQ signatures would still need to be transferred which minimizes the improvement this mechanism offers.

VIII. Conclusion and Future Work

In this work, we integrated PQ signature algorithms into TLS 1.3 and evaluated the TLS handshake latency observed by a client along with the throughput of a PQ authenticated server by considering realistic network conditions. We proved that signature and certificate chain size impacts the total handshake time, especially with relatively fast signing and verification primitives. Our results show that the PQ algorithms with the best performance for time-sensitive protocols like HTTPS are Dilithium and Falcon. When floating point hardware is available Falcon seems more suitable for the web. Other protocols (e.g. VPN, SSH) that do not require frequent connection establishments could use one of the other PQ signature algorithms as well, assuming they do not overload a server terminating multiple client connections. We showed that although slightly slower signing does not significantly affect the overall duration of a single handshake, it could significantly impact the total throughput of a server. However, as optimizations and hardware accelerations improve signing performance, we expect signature and key size to have the most impact on the handshake and total server throughput. We also proposed combining different PQ signature algorithms in a certificate chain and we confirmed that it can improve the overall handshake speed. Finally, we offered improvements to

avoid round-trips by leveraging ICA suppression and discussed other challenges and alternatives in real-world protocols.

As future work, we plan to evaluate the performance of PQ authenticated VPNs and UDP-based tunnels like QUIC and DTLS. We also want to test hybrid PQ KEMs as tested by others [51], [52] along with PQ signatures to evaluate the "total" latency introduced to TLS by PQ algorithms. By combining results in [52] with our work, we could extrapolate that the total slowdown of a PQ handshake would be around 10-25%. Also, given that PQ key exchange could add 1-2KB each direction, this could push the server data to the `initcwnd` limit and introduce round-trips. Further experiments would be necessary to quantify the total PQ algorithm impact on TLS under realistic conditions that include lossy networks. It would also be of interest to study the impact of PQ SCT and OCSP signatures while using the caching and ICA suppression mechanisms already discussed. Finally, it is worth investigating hybrid certificates' [66] performance and the improvement that Stateful HBS, qTesla, GeMSS, LUOV or Rainbow would achieve when used only at the root CA. As a longer-term goal, in an attempt to reduce the certificate size, we would also like to study the message recovery capabilities offered by schemes like Falcon.

Acknowledgments

Many thanks to Richard Barnes for his valuable feedback. We would also like to acknowledge Luke Valenta from Cloudflare for his useful feedback and experimental results regarding longer TLS records in TLS handshakes. Hanno Böck also was kind enough to challenge us very insightfully on the concerns PQ signatures bring to PKI infrastructures. Finally, thank you to D. Stebila, Christian Paquin and the whole OQS OpenSSL [72] team for providing a library that made our testing possible.

References

[1] Akamai, "Akamai online retail performance report: Milliseconds are critical," https://www.akamai.com/uk/en/about/news/press/2017-press/akamai-releases-spring-2017-state-of-online-retail-performance-report.jsp, 2017.

[2] G. Alagic, G. Alagic, J. Alperin-Sheriff, D. Apon, D. Cooper, Q. Dang, Y.-K. Liu, C. Miller *et al.*, *Status report on the first round of the NIST post-quantum cryptography standardization process*. US Department of Commerce, National Institute of Standards and Technology, 2019.

[3] E. Alkim, P. S. L. M. Barreto, N. Bindel, P. Longa, and J. E. Ricardini, "the lattice-based digital signature scheme qtesla."

[4] ANSI, "ANSI X9.62, Public Key Cryptography For The Financial Services Industry: The Elliptic Curve Digital Signature Algorithm (ECDSA)," September 2005, american National Standards Institute, X9-Financial Services.

[5] J.-P. Aumasson, D. J. Bernstein, C. Dobraunig, M. Eichlseder, S. Fluhrer, S.-L. Gazdag, A. Hülsing, P. Kampanakis, S. Kölbl, T. Lange *et al.*, "SPHINCS+ - Submission to the 2nd round of the NIST post-quantum project," https://sphincs.org/data/sphincs+-round2-specification.pdf, 2019, Specification document (part of the submission package).

[6] K. Basu, D. Soni, M. Nabeel, and R. Karri, "NIST post-quantum cryptography - a hardware evaluation study," Cryptology ePrint Archive, Report 2019/047, 2019, https://eprint.iacr.org/2019/047.

[7] M. Belshe, R. Peon, and M. Thomson, "Hypertext Transfer Protocol Version 2 (HTTP/2)," RFC 7540, May 2015. [Online]. Available: https://rfc-editor.org/rfc/rfc7540.txt

[8] D. Benjamin, "Batch Signing for TLS," Internet Engineering Task Force, Internet-Draft draft-davidben-tls-batch-signing-02, Nov. 2019, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/html/draft-davidben-tls-batch-signing-02

[9] D. J. Bernstein, "eBACS: ECRYPT Benchmarking of Cryptographic Systems," https://bench.cr.yp.to/primitives-sign.html, 2019, Web page. Accessed 2019-02-08.

[10] ——, "Visualizing size-security tradeoffs for lattice-based encryption," Cryptology ePrint Archive, Report 2019/655, 2019, https://eprint.iacr.org/2019/655.

[11] D. J. Bernstein, D. Hopwood, A. Hülsing, T. Lange, R. Niederhagen, L. Papachristodoulou, M. Schneider, P. Schwabe, and Z. Wilcox-O'Hearn, "SPHINCS: Practical Stateless Hash-Based Signatures," in *Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques*, 2015, pp. 368–397.

[12] W. Beullens, A. Szepieniec, F. Vercauteren, and B. Preneel, "Luov: Signature scheme proposal for NIST PQC project (Round 2 version)," https://github.com/WardBeullens/LUOV/blob/master/Supporting_Documentation/luov.pdf, 2018.

[13] N. Bindel, U. Herath, M. McKague, and D. Stebila, "Transitioning to a quantum-resistant public key infrastructure," in *Proc. 8th International Conference on Post-Quantum Cryptography (PQCrypto) 2017*, ser. LNCS, T. Lange and T. Takagi, Eds. Springer, June 2017, to appear.

[14] J. Bos, L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, J. M. Schanck, P. Schwabe, G. Seiler, and D. Stehlé, "Crystals – kyber: a cca-secure module-lattice-based kem," Cryptology ePrint Archive, Report 2017/634, 2017, https://eprint.iacr.org/2017/634.

[15] J. W. Bos, C. Costello, M. Naehrig, and D. Stebila, "Post-quantum key exchange for the TLS protocol from the ring learning with errors problem," in *2015 IEEE Symposium on Security and Privacy*. IEEE, 2015, pp. 553–570.

[16] J. A. Buchmann, D. Butin, F. Göpfert, and A. Petzoldt, "Post-quantum cryptography: state of the art," in *The New Codebreakers*. Springer, 2016, pp. 88–108.

[17] D. Butin, J. Walde, and J. A. Buchmann, "Post-quantum authentication in OpenSSL with hash-based signatures," in *Tenth International Conference on Mobile Computing and Ubiquitous Network, ICMU 2017, Toyama, Japan, October 3-5, 2017*. IEEE, 2017, pp. 1–6. [Online]. Available: https://doi.org/10.23919/ICMU.2017.8330093

[18] M. Campagna, "Hybrid-key Exchanges as an Interim-to-Permanent solution to cryptographic agility," Jun. 2019. [Online]. Available: https://docbox.etsi.org/Workshop/2019/201906_ETSISECURITYWEEK/202106_DynamicNatureOfTechno/SESSION03_CHANGINGCRYPTOGRAPHY/AWS_CAMPAGNA.pdf

[19] A. Casanova, J.-C. Faug'ere, G. Macario-Rat, J. Patarin, L. Perret, and J. Ryckeghem, "GeMSS: A Great Multivariate Short Signature," Ph.D. dissertation, PhD thesis, UPMC-Paris 6 Sorbonne Universités, 2017.

[20] C.-l. Chan, R. Fontugne, K. Cho, and S. Goto, "Monitoring TLS adoption using backbone and edge traffic," in *IEEE INFOCOM 2018-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. IEEE, 2018, pp. 208–213.

[21] M.-S. Chen, A. Hülsing, J. Rijneveld, S. Samardjiska, and P. Schwabe, "MQDSS specifications," http://mqdss.org/specification.html.

[22] D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley, and W. Polk, "RFC 5280: Internet X. 509 public key infrastructure certificate and certificate revocation list (CRL) profile," *IETF, May*, 2008.

[23] C. Cremers, M. Horvat, J. Hoyland, S. Scott, and T. van der Merwe, "A Comprehensive Symbolic Analysis of TLS 1.3," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '17. New York, NY, USA: ACM, 2017, pp. 1773–1788. [Online]. Available: http://doi.acm.org/10.1145/3133956.3134063

[24] C. J. F. Cremers, M. Horvat, S. Scott, and T. van der Merwe, "Automated Analysis and Verification of TLS 1.3: 0-RTT, Resumption and Delayed Authentication," *2016 IEEE Symposium on Security and Privacy (SP)*, pp. 470–485, 2016.

[25] E. Crockett, C. Paquin, and D. Stebila, "Prototyping post-quantum and hybrid key exchange and authentication in tls and ssh," Cryptology ePrint Archive, Report 2019/858, 2019, https://eprint.iacr.org/2019/858.

[26] E. Cronin, S. Jamin, T. Malkin, and P. McDaniel, "On the performance,

feasibility, and use of forward-secure signatures," in *Proceedings of the 10th ACM Conference on Computer and Communications Security*, ser. CCS '03. New York, NY, USA: ACM, 2003, pp. 131–144. [Online]. Available: http://doi.acm.org/10.1145/948109.948130

[27] J. Ding, M.-S. Chen, A. Petzoldt, D. Schmidt, and Y. Bo-Yin, "Rainbow - Algorithm Specification and Documentation," https://csrc.nist.gov/projects/post-quantum-cryptography/round-2-submissions, 2019, The 2nd Round Proposal.

[28] J. Ding, Z. Zhang, J. Deaton, K. Schmidt, and F. Vishakha, "New Attacks on Lifted Unbalanced oil vinega," https://csrc.nist.gov/CSRC/media/Events/Second-PQC-Standardization-Conference/documents/accepted-papers/ding-new-attacks-luov.pdf, 2019.

[29] M. Driscoll, "The Illustrated TLS 1.3 Connection: Every byte explained," https://tls13.ulfheim.net, 2018, Web page. Accessed 2019-21-08.

[30] L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, P. Schwabe, G. Seiler, and D. Stehlé, "Crystals-dilithium: A lattice-based digital signature scheme," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pp. 238–268, 2018.

[31] ——, "CRYSTALS-Dilithium Algorithm Specifications and Supporting Documentation," https://pq-crystals.org/dilithium/resources.shtml, 2018, Submission to round 2 of the NIST post-quantum project.

[32] ETSI, "ETSI TC Cyber Working Group for Quantum-Safe Cryptography," https://portal.etsi.org/TBSiteMap/CYBER/CYBERQSCToR.aspx, 2017, Web page. Accessed 2019-07-25.

[33] S. Fluhrer, D. McGrew, P. Kampanakis, and V. Smyslov, "Postquantum Preshared Keys for IKEv2," Internet Engineering Task Force, Internet-Draft draft-ietf-ipsecme-qr-ikev2-08, Mar. 2019, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/html/draft-ietf-ipsecme-qr-ikev2-08

[34] P.-A. Fouque, J. Hoffstein, P. Kirchner, V. Lyubashevsky, T. Pornin, T. Prest, T. Ricosset, G. Seiler, W. Whyte, and Z. Zhang, "Falcon: Fast-Fourier lattice-based compact signatures over NTRU," https://csrc.nist.gov/projects/post-quantum-cryptography/round-2-submissions, 2018, Specification v1.1.

[35] J. Fulmer, "Siege HTTP regression testing and benchmarking utility," https://www.joedog.org/siege-home/, 2019, Web page. Accessed 2019-02-09.

[36] A. Ghedini and V. Vasiliev, "TLS Certificate Compression," Internet Engineering Task Force, Internet-Draft draft-ietf-tls-certificate-compression-05, Apr. 2019, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/html/draft-ietf-tls-certificate-compression-05

[37] Z. Greg *et al.*, "The Picnic Signature Algorithm Design Document," https://github.com/microsoft/Picnic/blob/master/spec/design-v2.1.pdf, 2019.

[38] ——, "The Picnic Signature Algorithm Specification," https://github.com/microsoft/Picnic/blob/master/spec/spec-v2.1.pdf, 2019.

[39] V. Gupta, D. Stebila, S. Fung, S. C. Shantz, N. Gura, and H. Eberle, "Speeding up Secure Web Transactions Using Elliptic Curve Cryptography," in *NDSS*, 2004.

[40] P. E. Hoffman, "The Transition from Classical to Post-Quantum Cryptography," Internet Engineering Task Force, Internet-Draft draft-hoffman-c2pq-05, May 2019, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/html/draft-hoffman-c2pq-05

[41] J. Hoffstein, J. Pipher, and J. H. Silverman, "Ntru: A ring-based public key cryptosystem," in *International Algorithmic Number Theory Symposium*. Springer, 1998, pp. 267–288.

[42] http archive, "Trends," http://httparchive.org/trends.php.

[43] A. Huelsing, D. Butin, S.-L. Gazdag, J. Rijneveld, and A. Mohaisen, "XMSS: eXtended Merkle Signature Scheme," RFC 8391, May 2018. [Online]. Available: https://rfc-editor.org/rfc/rfc8391.txt

[44] A. Hülsing, J. Rijneveld, S. Samardjiska, and P. Schwabe, "From 5-pass MQ-based identification to MQ-based signatures." *IACR Cryptology ePrint Archive*, vol. 2016, p. 708, 2016.

[45] International Telecommunications Union, "X.509: Information technology - open systems interconnection - the directory: Public-key and attribute certificate frameworks," https://www.itu.int/rec/T-REC-X.509/en.

[46] P. Kampanakis, P. Panburana, E. Daw, and D. Van Geest, "The Viability of Post-quantum X.509 Certificates." *IACR Cryptology ePrint Archive*, vol. 2018, p. 63, 2018.

[47] M. J. Kannwischer, J. Rijneveld, P. Schwabe, and K. Stoffelen, "pqm4: Testing and Benchmarking NIST PQC on ARM Cortex-M4," Cryptology ePrint Archive, Report 2019/844, 2019, https://eprint.iacr.org/2019/844.

[48] T. Kivinen and J. Snyder, "Signature Authentication in the Internet Key Exchange Version 2 (IKEv2)," RFC 7427, Jan. 2015. [Online]. Available: https://rfc-editor.org/rfc/rfc7427.txt

[49] S. Kölbl, M. M. Lauridsen, F. Mendel, and C. Rechberger, "Haraka v2-efficient short-input hashing for post-quantum applications," *IACR Transactions on Symmetric Cryptology*, pp. 1–29, 2016.

[50] P. Kotzias, A. Razaghpanah, J. Amann, K. G. Paterson, N. Vallina-Rodriguez, and J. Caballero, "Coming of age: A longitudinal study of tls deployment," in *Proceedings of the Internet Measurement Conference 2018*. ACM, 2018, pp. 415–428.

[51] K. Kwiatkowski, "Towards Post-Quantum Cryptography in TLS," Jun. 2019. [Online]. Available: https://blog.cloudflare.com/towards-post-quantum-cryptography-in-tls/

[52] K. Kwiatkowski and L. Valenta, "The TLS Post-Quantum Experiment," https://blog.cloudflare.com/the-tls-post-quantum-experiment/.

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

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

[55] ——, "Post-quantum confidentiality for TLS," Apr. 2018. [Online]. Available: https://www.imperialviolet.org/2018/04/11/pqconftls.html

[56] B. Laurie, A. Langley, and E. Kasper, "Certificate Transparency," RFC 6962, Jun. 2013. [Online]. Available: https://rfc-editor.org/rfc/rfc6962.txt

[57] V. Lyubashevsky, C. Peikert, and O. Regev, "On Ideal Lattices and Learning with Errors Over Rings," Cryptology ePrint Archive, Report 2012/230, 2012, https://eprint.iacr.org/2012/230.

[58] D. McGrew, M. Curcio, and S. Fluhrer, "Leighton-Micali Hash-Based Signatures," RFC 8554, Apr. 2019. [Online]. Available: https://rfc-editor.org/rfc/rfc8554.txt

[59] K. Moriarty, B. Kaliski, J. Jonsson, and A. Rusch, "PKCS# 1: RSA cryptography specifications version 2.2," *Internet Engineering Task Force, Request for Comments*, vol. 8017, 2016.

[60] M. Mosca, "Cybersecurity in an era with quantum computers: will we be ready?" *IEEE Security & Privacy*, vol. 16, no. 5, pp. 38–41, 2018.

[61] Mozilla, "Mozilla Telemetry Portal - Measurement Dashboard - HTPT_PAGE_TLS_HANDSHAKE distribution for Firefox Desktop," https://telemetry.mozilla.org/new-pipeline/dist.html, 2018, Beta 68/69, any OS, any architecture, any process. Web page. Accessed 2019-21-08.

[62] D. Naylor, A. Finamore, I. Leontiadis, Y. Grunenberger, M. Mellia, M. Munafò, K. Papagiannaki, and P. Steenkiste, "The cost of the S in HTTPS," in *Proceedings of the 10th ACM International on Conference on emerging Networking Experiments and Technologies*. ACM, 2014, pp. 133–140.

[63] H. Nejatollahi, N. Dutt, S. Ray, F. Regazzoni, I. Banerjee, and R. Cammarota, "Post-quantum lattice-based cryptography implementations: A survey," *ACM Computing Surveys (CSUR)*, vol. 51, no. 6, p. 129, 2019.

[64] Nginx, "NGINX: High Performance Load Balancer Web Server and Reverse Proxy," https://www.nginx.com, 2019, Web page. Accessed 2019-02-09.

[65] Y. Nir, "Repeated Authentication in Internet Key Exchange (IKEv2) Protocol," Internet Requests for Comments, RFC Editor, RFC 4478, Apr. 2006. [Online]. Available: http://www.rfc-editor.org/rfc/rfc4478.txt

[66] M. Ounsworth and M. Pala, "Composite Keys and Signatures For Use In Internet PKI," Internet Engineering Task Force, Internet-Draft draft-ounsworth-pq-composite-sigs-01, Jul. 2019, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/html/draft-ounsworth-pq-composite-sigs-01

[67] C. Paquin, D. Stebila, and G. Tamvada, "Benchmarking post-quantum cryptography in tls," Cryptology ePrint Archive, Report 2019/1447, 2019, https://eprint.iacr.org/2019/1447.

[68] C. Peikert, "A decade of lattice cryptography," *Found. Trends Theor. Comput. Sci.*, vol. 10, no. 4, pp. 283–424, Mar. 2016. [Online]. Available: http://dx.doi.org/10.1561/0400000074

[69] R. Pino, V. Lyubashevsky, and D. Pointcheval, "The Whole is Less Than the Sum of Its Parts: Constructing More Efficient Lattice-Based AKEs," in *Proceedings of the 10th International Conference on Security and Cryptography for Networks - Volume 9841*. Berlin, Heidelberg: Springer-Verlag, 2016, pp. 273–291. [Online]. Available: https://doi.org/10.1007/978-3-319-44618-9_15

[70] T. Pornin, "New efficient, constant-time implementations of falcon," Cryptology ePrint Archive, Report 2019/893, 2019, https://eprint.iacr.org/2019/893.

[71] O. Project, "liboqs," https://github.com/open-quantum-safe/liboqs, 2019, Web page. Accessed 2019-02-08.

[72] ——, "OQS OpenSSL," https://github.com/open-quantum-safe/openssl, 2019, Web page. Accessed 2019-02-08.

[73] P. Project, "PQClean," https://github.com/PQClean/PQClean, 2019, Web page. Accessed 2019-02-09.

[74] J. Proos and C. Zalka, "Shor's discrete logarithm quantum algorithm for elliptic curves," *Quantum Info. Comput.*, vol. 3, no. 4, pp. 317–344, Jul. 2003. [Online]. Available: http://dl.acm.org/citation.cfm?id=2011528.2011531

[75] O. Regev, "On lattices, learning with errors, random linear codes, and cryptography," *Journal of the ACM (JACM)*, vol. 56, no. 6, p. 34, 2009.

[76] E. Rescorla, "The transport layer security (TLS) protocol version 1.3," 2018.

[77] E. Rescorla, R. Barnes, and H. Tschofenig, "Compact TLS 1.3," Internet Engineering Task Force, Internet-Draft draft-rescorla-tls-ctls-03, Nov. 2019, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/html/draft-rescorla-tls-ctls-03

[78] E. Rescorla, K. Oku, N. Sullivan, and C. A. Wood, "Encrypted Server Name Indication for TLS 1.3," Internet Engineering Task Force, Internet-Draft draft-ietf-tls-esni-05, Nov. 2019, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/html/draft-ietf-tls-esni-05

[79] E. Rescorla, N. Sullivan, and C. A. Wood, "Semi-Static Diffie-Hellman Key Establishment for TLS 1.3," Internet Engineering Task Force, Internet-Draft draft-rescorla-tls-semistatic-dh-02, Nov. 2019, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/html/draft-rescorla-tls-semistatic-dh-02

[80] C. Roma, C.-E. A. Tai, and M. A. Hasan, "Energy Consumption of Round 2 submissions for NIST PQC Standards," *Second PQC Standardization Conference*, Aug 2019.

[81] M.-J. O. Saarinen, "Mobile energy requirements of the upcoming nist post-quantum cryptography standards," 2019.

[82] S. Santesson and H. Tschofenig, "Transport Layer Security (TLS) Cached Information Extension," RFC 7924, Jul. 2016. [Online]. Available: https://rfc-editor.org/rfc/rfc7924.txt

[83] J. Sepúlveda, S. Liu, and J. M. B. Mera, "Post-quantum enabled cyber physical systems," *IEEE Embedded Systems Letters*, 2019.

[84] SHODAN, "HTTPS (443) Overview," Jul. 2019, https://www.shodan.io/report/nWlAWhKG.

[85] P. W. Shor, "Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer," *SIAM J. on Computing*, vol. 26, no. 5, pp. 1484–1509, 1997.

[86] D. Sikeridis, I. Papapanagiotou, B. P. Rimal, and M. Devetsikiotis, "A Comparative taxonomy and survey of public cloud infrastructure vendors," *arXiv preprint arXiv:1710.01476*, 2017.

[87] D. Soni, K. Basu, M. Nabeel, and R. Karri1, "A Hardware Evaluation Study of NIST Post-Quantum Cryptographic Signature schemes," *Second PQC Standardization Conference*, Aug 2019.

[88] D. Stebila and M. Mosca, "Post-Quantum Key Exchange for the Internet and the Open Quantum Safe Project," Cryptology ePrint Archive, Report 2016/1017, 2016, https://eprint.iacr.org/2016/1017.

[89] D. Steblia, S. Fluhrer, and S. Gueron, "Design issues for hybrid key exchange in TLS 1.3," Internet Engineering Task Force, Internet-Draft draft-stebila-tls-hybrid-design-01, Jul. 2019, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/html/draft-stebila-tls-hybrid-design-01

[90] M. Thomson, "Suppressing Intermediate Certificates in TLS," Internet Engineering Task Force, Internet-Draft draft-thomson-tls-sic-00, Mar. 2019, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/html/draft-thomson-tls-sic-00

[91] C. Tjhai, M. Tomlinson, grbartle@cisco.com, S. Fluhrer, D. V. Geest, O. Garcia-Morchon, and V. Smyslov, "Framework to Integrate Post-quantum Key Exchanges into Internet Key Exchange Protocol Version 2 (IKEv2)," Internet Engineering Task Force, Internet-Draft draft-tjhai-ipsecme-hybrid-qske-ikev2-04, Jul. 2019, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/html/draft-tjhai-ipsecme-hybrid-qske-ikev2-04

[92] P. Yee, "Updates to the Internet X.509 public key infrastructure certificate and Certificate Revocation List (CRL) profile," 2013.

[93] Y. Yoo, R. Azarderakhsh, A. Jalali, D. Jao, and V. Soukharev, "A Post-Quantum Digital Signature Scheme Based on Supersingular Isogenies," Cryptology ePrint Archive, Report 2017/186, 2017, http://eprint.iacr.org/2017/186.