Mixed Certificate Chains for the Transition to Post-Quantum Authentication in TLS 1.3[*]

Sebastian Paul
Robert Bosch GmbH
Renningen, Germany
sebastian.paul2@de.bosch.com

Yulia Kuzovkova
Robert Bosch GmbH
Renningen, Germany

Norman Lahr
Fraunhofer SIT
Darmstadt, Germany
norman@lahr.email

Ruben Niederhagen
University of Southern Denmark
Odense, Denmark
ruben@polycephaly.org

ABSTRACT

Large-scale quantum computers will be able to efficiently solve the underlying mathematical problems of widely deployed public key cryptosystems in the near future. This threat has sparked increased interest in the field of Post-Quantum Cryptography (PQC) and standardization bodies like NIST, IETF, and ETSI are in the process of standardizing PQC schemes as a new generation of cryptography. This raises the question of how to ensure a fast, reliable, and secure transition to upcoming PQC standards in today's highly interconnected world.

In this work, we propose and investigate a migration strategy towards post-quantum (PQ) authentication for the network protocol Transport Layer Security (TLS). Our strategy is based on the concept of "mixed certificate chains" which use different signature algorithms within the same certificate chain. In order to demonstrate the feasibility of our migration strategy we combine the well-studied and trusted hash-based signature schemes SPHINCS$^+$ and XMSS with elliptic curve cryptography first and subsequently with lattice-based PQC signature schemes (CRYSTALS-Dilithium and Falcon). Furthermore, we combine authentication based on mixed certificate chains with the lattice-based key encapsulation mechanism (KEM) CRYSTALS-Kyber as representative for PQC KEMs to evaluate a fully post-quantum and mutually authenticated TLS 1.3 handshake.

Our results show that mixed certificate chains containing hash-based signature schemes only at the root certificate authority level lead to feasible connection establishment times despite the increase in communication size. By analyzing code size and peak memory usage of our client and server programs we further demonstrate the suitability of our migration strategy even for embedded devices.

KEYWORDS

Transport Layer Security, Post-Quantum Cryptography, Authentication, Public Key Infrastructure, Embedded Systems.

1 INTRODUCTION

With recent landmark achievements in the realm of quantum-computing research, the advent of powerful, large-scale quantum computers seems inevitable [4, 53]. Because of the existence of Shor's algorithm — a polynomial time quantum algorithm solving integer factorization and discrete logarithms — most currently used public key cryptography (denoted as "conventional cryptography" in the following) will be broken once sufficiently large universal quantum computers are built. This looming threat has sparked an increased interest in the field of Post-Quantum Cryptography (PQC), leading to numerous standardization efforts [15, 25, 41].

Compared to current public key cryptosystems, PQC primitives generally incur a higher cost in some metric: computational cost, storage requirements, or network bandwidth. Consequently, the impact of these novel schemes needs to be carefully evaluated when integrated into protocols and applications. Since key-exchange messages of conventionally established sessions can be recorded, an adversary may later be able to break into such sessions once a powerful quantum computer is available. Hence, it is considered critical to protect the confidentiality of application data against such attack scenarios. As a result, most existing migration strategies focus on confidentiality by fostering hybrid key exchange (KEX) in protocols [13, 16, 33, 52]. In these hybrid schemes, conventional key agreement, e. g., ECDHE, is combined with a post-quantum (PQ) key encapsulation mechanism (KEM) to eventually protect against "harvest now, decrypt later" attacks.

As adversaries cannot retroactively impersonate entities in communication sessions, i. e., break authentication properties, the migration towards post-quantum authentication has attracted less attention so far. Nevertheless, the migration to post-quantum authentication still needs to be completed before large-scale quantum computers exist. Since authentication is typically based on long-term public keys in form of certificates involving trusted third parties, i. e., certificate authorities (CAs), a transition towards post-quantum authentication is generally considered complex and time-consuming. In this transition the network protocol Transport Layer Security (TLS) will play a very important role, since it is widely regarded as the "gold standard" for building secure, networked applications. Thus, fast adoption of upcoming PQC standards requires (a) schemes that are fully trusted in the security community (e. g., long track record of no attacks), (b) a clear understanding of the performance impact of PQC, and (c) viable migration plans for applications and protocols that cover confidentiality as well as authentication.

[*]Accepted for publication at the 17th ACM Asia Conference on Computer and Communications Security (ACM ASIA CCS '22), May 30–June 3, 2022, Nagasaki, Japan.

Contributions. In this work, we propose and investigate a migration strategy towards post-quantum authentication for TLS 1.3 — the most recent version of the TLS standard [46]. Our approach is based on the concept of "mixed certificate chains", where we combine well-studied and trusted hash-based signature schemes (SPHINCS$^+$ or XMSS) at the root CA level with either conventional elliptic curve cryptography (ECC) or lattice-based PQC signature schemes (CRYSTALS-Dilithium or Falcon) at the subsequent certificate levels, i. e., intermediate certificate authorities (ICAs) and end entities (EEs). To demonstrate the feasibility of our migration strategy, we perform several experiments under realistic network conditions. The main contributions of our work are summarized as follows:

- Our two-step migration strategy based on mixed certificate chains presents a novel solution for the transition to upcoming PQC standards and post-quantum authentication.
- We propose nine mixed certificate chains as part of our migration strategy as well as six control certificate chains and assess their impact on the TLS 1.3 handshake in combination with the lattice-based KEM CRYSTALS-Kyber.[1]
- We conduct our experiments under realistic network conditions: Our setup consists of two client devices with different computing and storage capabilities; our three cloud-based servers are set up with increasing client-server distances.
- Since use cases like company networks or Internet of Things (IoT) applications increasingly require client authentication, we focus on mutual authentication for TLS in our experiments, which has not been investigated in previous studies.
- We demonstrate the feasibility of our proposed migration strategy even for embedded devices by reporting the following performance measurements: (i) connection establishment times, (ii) communication size, (iii) code size, and (iv) peak memory usage.

Outline. In Section 2, we introduce the reader to PQC in general and motivate our selection of PQC schemes for signatures as well as key establishment. Section 3 highlights related work. In Section 4 we describe the migration strategy based on mixed certificate chains and the integration of PQC into the TLS 1.3 handshake. We present and evaluate our results in Section 5. Section 6 concludes our work.

2 POST-QUANTUM CRYPTOGRAPHY

The goal of PQC is to provide cryptographic primitives that are able to withstand attacks using the computing power of quantum computers. While the mathematical problems of integer factorization and computing a discrete logarithm are the foundation of many of today's cryptographic schemes, they can be efficiently solved by a general quantum computer. However, there remain alternative mathematical problems where also a quantum computer does not efficiently provide a solution. Such problems, hence, are suitable candidates to construct PQC schemes.

It is important to note that while PQC schemes defend against quantum computers, they themselves do not require quantum technologies for their practical use — as opposed to, e.g., Quantum Key Distribution. For example, "classical" computing devices are capable of computing or verifying a PQC signature.

[1]The source-code of our integrations of PQC into the open-source library wolfSSL is publicly available at https://github.com/boschresearch/pq-wolfSSL.

PQC schemes are categorized into families based on their underlying hard mathematical problem: There are the families of lattice-based, code-based, hash-based, isogeny-based, and multivariate schemes. The underlying mathematical constructions result in very different properties of the cryptographic schemes in regard to efficiency, computing time, and memory consumption as well as message, key, and signature sizes, etc.

Lattice-based schemes are usually very efficient and have "medium" key, ciphertext, and signature sizes — larger than the sizes of the currently deployed RSA and elliptic curve (EC) cryptosystems but smaller or more balanced than most other PQC families. Some code-based and multivariate schemes have very large public keys; hash-based schemes, on the other hand, have very small public keys but large signature sizes. Today, there is a vast selection of schemes in each of the PQC families. This often allows to find a PQC scheme that is suitable for a specific application. However, PQC generally incurs a higher cost in some metric (computational cost, storage, or network bandwidth) compared to conventional schemes currently in use. This often is a challenge in particular for embedded devices that provide a restricted amount of resources.

2.1 PQC Standardization

Since December 2016, there has been an ongoing (as of 2021) standardization effort by the National Institute of Standards and Technology (NIST) for PQC schemes. The standardization process is conducted in form of an open "competition": The international research community is invited to submit, vet, and improve submissions from all major PQC families. In the initial first round, almost 80 PQC schemes had been submitted. This field has been narrowed down by NIST to only 15 schemes in the ongoing third round: The "Round 3 Finalists" are for digital signature algorithms (DSAs) CRYSTALS-Dilithium [7], Falcon [26] (both lattice-based), and Rainbow [23] (multivariate) as well as for public key encryption and key establishment Classic McEliece [1] (code-based), CRYSTALS-Kyber [6], NTRU [20], and Saber [9] (all lattice-based). The "Alternate Candidates" are for digital signature algorithms GeMSS [17] (multivariate), Picnic [19] (symmetric), and SPHINCS$^+$ [5] (hash-based) as well as for public key encryption and key establishment BIKE [3], HQC [39] (both code-based), FrodoKEM [2], NTRU Prime [10] (both lattice-based), and SIKE [31] (isogeny-based). As the above list shows, 7 out of the 15 remaining candidates belong to the family of lattice-based PQC with balanced performance profiles. Nevertheless, it is likely that NIST will standardize several of these candidates from different PQC families to provide schemes with different properties in regards to resource utilization and efficiency.

In addition to the NIST standardization efforts, the Internet Engineering Task Force (IETF) already published two Requests for Comments (RFCs) for PQC signatures: XMSS [30] and LMS [37]. Both of these schemes are also recommended by NIST to be used in particular applications such as firmware signatures [22]. As the security of hash-based signature schemes only depends on security assumptions of the underlying hash functions, they are well trusted and offer strong security guarantees even against attacks aided by quantum computers.

Since their one-time signature keys must not be reused, XMSS and LMS are *stateful* hash-based signature schemes. The construction of SPHINCS$^+$ eliminates the need for maintaining a state and thus it is a *stateless* signature scheme. Despite their strong security proofs, their adoption in everyday applications appears to be challenging mostly due to larger signatures and slow signing times.

2.2 Selection of Key Establishment and Signature Schemes

For the study in this work, we selected several PQC signature schemes for the evaluation of mixed certificate chains and one representative PQC key encapsulation mechanism for a complete post-quantum TLS handshake. Note that we only selected one KEM for key establishment since we are focusing on the migration towards post-quantum authentication in TLS. In turn, this reduces the dimensions of scheme combinations and allows for a clear evaluation of our migration strategy.

For the signature schemes in the certificate chain we selected hash-based schemes, which have large signatures but small public keys, and lattice-based schemes, which have moderate key and signature sizes. Hash-based signature schemes are very robust and well trusted. Due to their small public key size and fast verification times, they can easily be used in addition to conventional signature schemes, e.g., at the root CA level even in small embedded devices. More specifically, we selected the hash-based scheme SPHINCS$^+$ from the NIST standardization process and XMSS as representative of the IETF PQC-RFCs and the two lattice-based signature schemes CRYSTALS-Dilithium and Falcon. We did not select signature schemes from other PQC families because of recent cryptanalytic progress on multivariate schemes, i. e., GeMSS and Rainbow [11], as well as Picnic [24], a post-quantum signature scheme based on symmetric cryptography and zero-knowledge proofs.

To achieve a complete PQC-based TLS 1.3 handshake we selected CRYSTALS-Kyber for key establishment — a finalist in NIST's standardization process. As a representative of lattice-based KEMs it offers some of the smallest key sizes and can be implemented very efficiently [32]. Besides that, CRYSTALS-Kyber has already proven to be a suitable candidate for the integration into TLS in several other works [14, 44, 49]. It is also part of the PQC-CRYSTALS suite and thus closely related to CRYSTALS-Dilithium. Consequently, it is a natural pairing for CRYSTALS-Dilithium and also fits well to the other signature schemes. Henceforth, we omit the CRYSTALS-prefix and refer to both schemes as Kyber and Dilithium. An overview of all selected schemes including key, signature, ciphertext sizes, and benchmark results on our experimental platforms is given in Table 2 in Section 5.1.

3 RELATED WORK

Since NIST started its PQC standardization process, there have been many initiatives for testing and benchmarking the proposed schemes. The SUPERCOP[2] toolkit integrates and benchmarks the schemes for x86-64 CPU architectures and the subprojects of the mupq-project[3] for various embedded platforms, such as ARM Cortex-M4 (pqm4) [32] or the more efficient Cortex-Ax processors (pqax).

[2]https://bench.cr.yp.to/supercop.html.
[3]https://github.com/mupq.

The studies on integrating PQC into TLS started with the integration of lattice-based key exchange in TLS 1.2 [12] and were extended by multivariate signature schemes in [18]. The authors of [42] developed an experimental framework for the TLS 1.3 handshake which is able to emulate network connections with real conditions. Therewith, they studied the impact of connection latency, packet loss rate, and packet fragmentation on various post-quantum signature schemes and (hybrid) key exchange. The work presented in [49] further evaluates the integration of PQC under real network conditions. These works complement the previous public industry experiments by Google [13, 34] and the follow-up collaboration between Google and Cloudflare [33, 35]. While these were highly realistic experiments, they only focused on hybrid key exchange.

Considering the evaluation of post-quantum TLS on embedded systems, [14] integrated and evaluated SPHINCS$^+$ and Kyber on four different platforms in the TLS 1.2 handshake using the mbed TLS library, from the high performance class of ARM Cortex-A53 down to Cortex-M0+. The authors of [43] provided integrations of PQC into the industrial protocol OPC UA, which performs its TLS-like handshake using X.509 certificates for mutual authentication.

The combination of different signature schemes in the same certificate chain has been evaluated in two previous works. The authors of [48] introduced a novel post-quantum TLS variant relying on key encapsulation mechanisms for authentication — KEMTLS. They compared their handshake variant against size optimized certificate chains and KEMs resulting in combinations of multivariate, hash-based, and lattice-based signature schemes in addition to isogeny-based key establishment. The authors of [50] found that a combination of Dilithium at end entities and Falcon at CA levels outperforms the respective PQC control certificate chains in TLS 1.3 handshakes, but they did not consider post-quantum key establishment in their performance study. As NIST already declared it will only standardize one lattice-based signature scheme, we do not consider combinations of Dilithium and Falcon in our work [40].

In this work, however, we assess mixed certificate chains as part of a migration strategy towards post-quantum authentication by combining hash-based schemes at the root CA level with conventional cryptography or lattice-based schemes at subsequent levels. Furthermore, we consider two client platforms in our experiments (x86-64 CPU and Cortex-A53) as well as client authentication in TLS 1.3, which has not been investigated in previous studies. As opposed to the existing works that combine PQC signature schemes across different chain levels, we do not necessarily aim for performance improvements in terms of connection establishment time. Instead, our main goal is to explore the feasibility of our migration strategy by also taking implementation cost (code size and memory usage) into account and by evaluating the overall impact on the TLS 1.3 handshake protocol under realistic network conditions.

4 TOWARDS POST-QUANTUM AUTHENTICATION IN TLS 1.3

In this section we describe our approach for the transition to post-quantum authentication in TLS 1.3 which is based on mixed certificate chains. In order to assess our proposed strategy we also introduce the selected combinations of signature schemes.

Besides that, we briefly explain the integration of PQC into the handshake of TLS 1.3 and the corresponding integration into the cryptographic library wolfSSL.

4.1 Migration Strategy Based on Mixed Certificate Chains

Digital certificates play a key role in today's public key infrastructures (PKIs). They contain a public key alongside additional information related to the owner (subject) who possesses the corresponding secret key. In essence, a digital certificate ties the owner's identity to its public key via the signature of a certificate authority. X.509 certificates are the most common standard for digital certificates. They are primarily used for digital authentication in protocols like TLS, SSH, and IKEv2.

In a typical hierarchical PKI for TLS connections a trusted root CA holds a self-signed root certificate and issues certificates to ICAs. In turn, the task of ICAs is to sign other ICA or EE certificates ultimately creating a chain of trust. Validating EE certificates requires that (a) the root CA certificate is trusted and securely stored on the respective entities and that (b) all signatures in the certificate chain are verified using the public key of the issuer's certificate (see Fig. 1). In this chain of trust, the self signed root certificate serves as the main trust anchor. As a consequence, root CAs typically operate offline in high-security environments in order to protect their secret signing key. While the distribution of root CA certificates is straightforward in desktop environments, e. g., via regular software updates, it is considered a major challenge in IoT applications, since IoT devices often lack appropriate software/firmware update mechanisms despite their long lifespans [21, 27].

Moreover, root CA certificates typically have long validity periods, often between 10–25 years [28]. As a result, root CAs will already have to consider a transition to PQC in case their certificates expire in the near future. ICA and EE certificates, on the other hand, have much shorter validity periods (between 1–10 years) [28]. Besides that, they can be renewed via standardized mechanisms. For instance, EEs in WebPKIs, i. e., TLS servers, can automatically renew their certificates via the Automated Certificate Management Environment (ACME) [8].

Therefore, we expect the following two aspects to prevent a fast and seamless transition to post-quantum authentication in TLS:

(1) Despite the fact that root CAs need to initiate the transition to PQC in the near future, most PQC signature schemes have not received the same level of scrutiny as currently deployed public key cryptography. However, as the main trust anchor in a PKI root CAs require fully-trusted and well-established signature schemes.
(2) Not all components of a PKI need a transition to PQC simultaneously. As authentication cannot be broken retroactively, there is no need for end entities to deploy PQC for authentication right away. In addition, certificates at higher levels, i. e., ICA or EE, can be exchanged more easily.

Out of all proposed PQC signature schemes, hash-based schemes are arguably considered most mature for real-world deployment, because (a) most of their underlying constructions, e. g., Merkle trees, have been studied for decades, (b) their security relies only on minimal assumptions, i. e., (second-)preimage resistance, and (c) stateful hash-based schemes are already specified in IETF RFCs and also recommended by NIST. As a consequence, hash-based schemes already fulfill the requirements for signature schemes at the root CA level. However, because of their slow signing times and large signature sizes it is a challenge to deploy them at other levels of a PKI especially in embedded IoT devices [14]. In addition, stateful hash-based schemes require careful state synchronization in order to prevent any leakage of secret key material. But since root CAs operate in high-security environments, we assume they are capable of effectively applying state management techniques, such as introduced in [38]. Stateless hash-based signature schemes, on the other hand, mitigate the drawback of state synchronization, at the cost of bigger signature sizes (see Table 2).

In order to ease the transition to post-quantum authentication in TLS we propose the following two-step migration strategy, in which hash-based signature schemes play a key role. In the first step, we combine hash-based schemes at the root level with conventional signature schemes, e. g., ECDSA, at the subsequent levels of a hierarchical PKI. In this step, ICAs and EEs will only be required to implement the verify operation of the respective hash-based scheme. Note that we consider this only as intermediate step towards complete post-quantum authentication. As quantum computers capable of breaking currently deployed public key cryptography will not become available within the next few years, we do not require authentication to resist such attacks during this first step.

In the second step of our migration strategy, we combine hash-based schemes with lattice-based signature schemes in order to provide full post-quantum authentication. We consider this a valid strategy for the following three reasons:

(1) Due to the long life span of the root CA certificates deployed in the first step, a complex and time-consuming redistribution of root certificates can be avoided.
(2) More PQC signature schemes will have reached the same level of maturity as hash-based schemes by then. At the time of writing, lattice-based schemes are arguably considered most suitable for general purposes, but in case of any cryptanalytic breakthroughs alternatives will exist.
(3) Since hash-based schemes offer comparatively fast verification times we expect to see only a negligible performance impact when combined with either conventional cryptography in the first step or lattice-based schemes in the second step of our migration strategy.

Nevertheless, in the second step, ICAs and EEs need to fully implement the respective lattice-based signature scheme in addition to the verification of hash-based signatures. As a consequence, we do not only consider TLS metrics such as handshake duration and communication size in our evaluation but also the impact of our migration strategy on code size and memory.

4.2 Signature Scheme Combinations

As motivated in Section 2, we selected the two hash-based signature schemes XMSS (stateful; short: *XMS*) and SPHINCS$^+$ (stateless; short: *SP*), the two lattice-based schemes Dilithium (short: *Dil*) and Falcon (short: *Fal*), as well as the conventional scheme ECDSA (short: *EDS*) for the evaluation of our migration strategy. Moreover, we consider both performance variants of the SPHINCS$^+$ signature framework: its speed-optimized instantiation (short: *SPf*, 'f' for fast) and its size-optimized instantiation (short: *SPs*, 's' for small).

Since our focus is on the evaluation of post-quantum signature schemes in TLS, we are considering only two key establishment schemes, ECDHE (short: *EDH*) as conventional scheme and Kyber (short: *Kyb*) as representative for lattice-based post-quantum KEMs.

We chose the parameter sets of the evaluated PQC schemes to target NIST security level 1 with two exceptions: the specification of Dilithium does not offer a NIST level 1 set, instead we work with its parameter set that corresponds to security level 2; XMSS has no official NIST security level, since it is not part of the NIST PQC process. For the conventional ECC schemes ECDSA and ECDHE we are using the curve SECP256R1.

For our experiments we considered a certificate chain length of three, which is the predominant case in current deployments of TLS (about 40.0 %) [49]. Since we also target client authentication, server and client are both equipped with their own device certificate and the respective certificate chain. Although mutual authentication is not commonly used in WebPKIs, several use cases for it exist outside of it, e. g., company networks and IoT applications [44]. In fact, many industry-specific protocols that rely on security features of TLS even advocate the use of mutual authentication, such as MQTT [36] and OPC UA [45].

Following our selection of signature schemes and aforementioned prerequisites, we derive two groups for our evaluation:

(1) *Control*: This group contains six 'regular' certificate chains containing only a single signature scheme. As a reference we compare our migration strategy against a fully conventional certificate chain within a fully conventional TLS handshake. In order to assess the feasibility of our approach, we compare mixed certificate chains against PQC-based certificate chains that contain either only a single hash-based or a single lattice-based signature scheme.
(2) *Mixed Certificate Chain*: This group presents our two-step migration strategy and contains nine combinations of signature schemes within certificate chains. We selected the three hash-based schemes (XMS, SPf, SPs) as signature schemes at the root CA level. In accordance with our two-step process, we combine XMSS and the two SPHINCS$^+$ instantiations with the conventional signature scheme ECDSA representing our first transitional migration step. To complete the migration to full post-quantum authentication, we subsequently combine the selected hash-based schemes with lattice-based PQC schemes (Dilithium and Falcon).

The complete list of all 15 combinations (including their abbreviated notation) is given in Table 1. We present the measurement results of these combinations in Section 5.

4.3 Post-Quantum TLS 1.3 Handshake

The goal of TLS is to establish an encrypted communication channel between two endpoints. Its most recent version — TLS 1.3 — was introduced in 2018 as RFC 8446 [46]. Within global TLS connections, TLS 1.3 still plays a minor role with a share of approximately only 5.4 % [29]. But considering current adoption rates [47], we expect it will be the pre-dominant version by the time PQC has become state-of-the-art. As a result, we focus on TLS 1.3 in our work.

Table 1: Evaluated scheme combinations within TLS 1.3.

	Signature Scheme				
	Root CA	*Interm. CA*	*End Entity*	*KEX*	*Notation*[4]
Control	ECDSA	ECDSA	ECDSA	ECDHE	*EDS-EDH*
	Dilithium	Dilithium	Dilithium	Kyber	*Dil-Kyb*
	Falcon	Falcon	Falcon	Kyber	*Fal-Kyb*
	XMSS	XMSS	XMSS	Kyber	*XMS-Kyb*
	SPHINCS$^+$-f	SPHINCS$^+$-f	SPHINCS$^+$-f	Kyber	*SPf-Kyb*
	SPHINCS$^+$-s	SPHINCS$^+$-s	SPHINCS$^+$-s	Kyber	*SPs-Kyb*
Mixed Certificate Chain	XMSS	ECDSA	ECDSA	Kyber	*XMS+EDS-Kyb*
	XMSS	Dilithium	Dilithium	Kyber	*XMS+Dil-Kyb*
	XMSS	Falcon	Falcon	Kyber	*XMS+Fal-Kyb*
	SPHINCS$^+$-f	ECDSA	ECDSA	Kyber	*SPf+EDS-Kyb*
	SPHINCS$^+$-f	Dilithium	Dilithium	Kyber	*SPf+Dil-Kyb*
	SPHINCS$^+$-f	Falcon	Falcon	Kyber	*SPf+Fal-Kyb*
	SPHINCS$^+$-s	ECDSA	ECDSA	Kyber	*SPs+EDS-Kyb*
	SPHINCS$^+$-s	Dilithium	Dilithium	Kyber	*SPs+Dil-Kyb*
	SPHINCS$^+$-s	Falcon	Falcon	Kyber	*SPs+Fal-Kyb*

[4] We denote our control certificate chains as $DSA_{Root\&ICA\&EE}$-*KEX* and our mixed certificate chains as DSA_{Root}+$DSA_{ICA\&EE}$-*KEX*.

Compared to its predecessor, TLS 1.3 offers major security as well as performance improvements in its handshake protocol [46]. Legacy symmetric cryptography is no longer supported and it even mandates the use of authenticated encryption. Besides that, encryption of all handshake messages after the ServerHello message has been added. A new concept of cipher suites has also been introduced, separating authentication and key exchange mechanisms from the symmetric mechanisms protecting the established secure tunnel. Furthermore, its modular design has been improved, which facilitates the integration of new cryptographic schemes. This makes it easier to integrate PQC into the handshake of TLS 1.3, as highlighted in previous works [42, 49, 50].

Fig. 1 shows the resulting TLS 1.3 handshake after the integration of PQC. Regarding key exchange, a post-quantum KEM first needs to be transformed into an ephemeral KEX scheme as follows. The client creates an ephemeral key pair using the KEM's key generation function. Within the ClientHello message it advertises the selected KEM as part of the supported_groups extension alongside the generated KEM public key as part of the key_share extension. Having received the ClientHello message, the server performs the KEM's encapsulation operation resulting in a shared secret and the ciphertext. The server then sends the ciphertext back to the client within the key_share extension of the ServerHello message. As we require mutual authentication in our experiments, the server requests client authentication by sending a CertificateRequest message. Besides that, the server transmits its X.509 certificate chain (excluding the root CA certificate) as part of the Certificate message as well as a post-quantum signature over the handshake transcripts[5] within the CertificateVerify message. The client subsequently performs the KEM's decapsulation operation now sharing a secret with

[5] Handshake transcript is a hash value computed by hashing the concatenation of all handshake messages that have been sent so far.

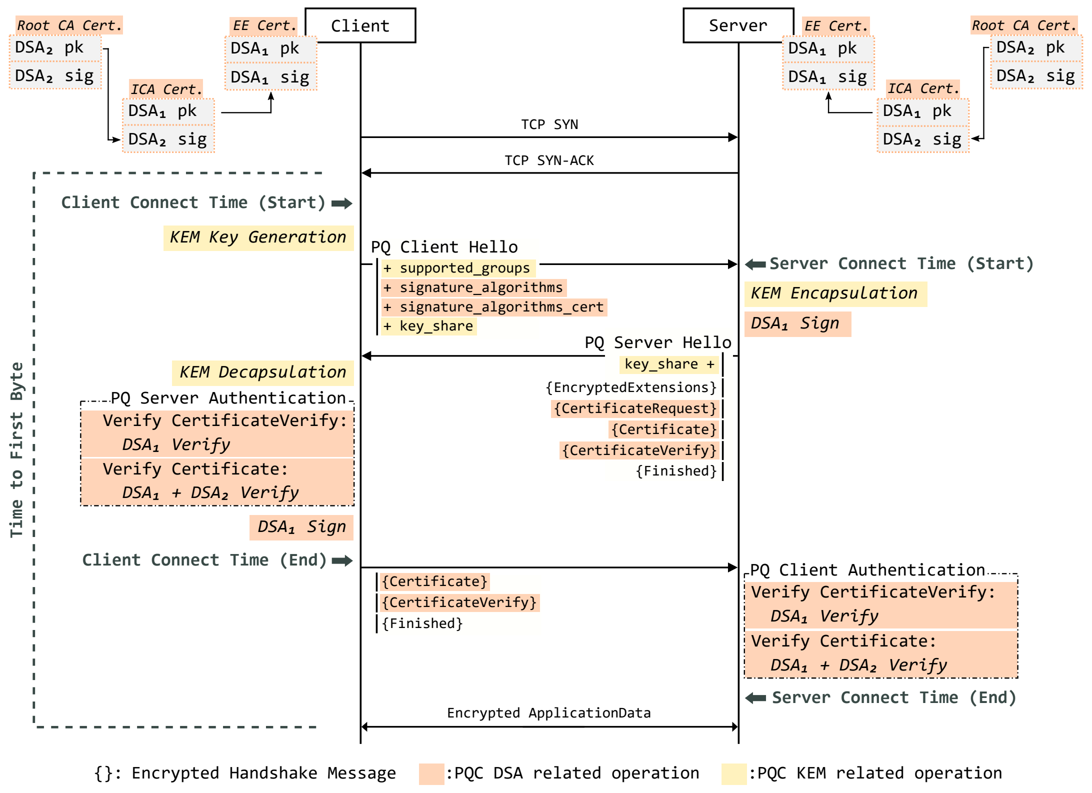


Figure 1: Post-quantum TLS 1.3 handshake illustrating the concept of mixed certificate chains.

the server. The client authenticates the server's identity by verifying the signature and the received certificate chain, performing a total of three PQC verify operations for a chain length of three. In response to the server's CertificateRequest message, the client also sends a signature over the handshake transcript (CertificateVerify) alongside its own certificate chain (excluding the root CA certificate) within the Certificate message. By verifying the signature over the handshake transcript and the entire certificate chain, the server authenticates the client, which completes the TLS 1.3 handshake.

4.4 Integration of PQC into wolfSSL

Existing integrations of PQC into TLS 1.3 libraries are aimed at desktop/server environments [33, 48, 51]. Work that targets embedded systems, on the other hand, only supports TLS 1.2 [14]. Therefore, we selected the open source TLS library wolfSSL (v4.7.0) for our integrations of PQC, because it is suitable for embedded systems and supports TLS 1.3.

WolfSSL consists of the cryptography engine wolfCrypt, which is responsible for all cryptography operations and services. As a result, the integration of PQC into wolfSSL requires: (a) to extend the wolfCrypt engine with PQC and (b) to make them accessible to wolfSSL's implementation of the TLS 1.3 handshake. We used the most recent reference implementations of the four selected NIST PQC schemes[6] and the latest reference implementation of XMSS[7]. Note that we did not make use of any architecture-specific optimizations (AVX2 and assembly) in our experiments.

Enabling PQC in the TLS handshake required to make the integrated schemes and newly defined Object Identifiers (OIDs) available in the respective modules of wolfSSL. To handle the larger key and signature size of the integrated PQC signature schemes, internal buffer sizes needed to be increased, such as MAX_X509_SIZE and MAX_CERT_VERIFY_SIZE. Currently, wolfSSL does not support fragmentation of the CertificateVerify messages for signatures larger than the maximum single record size of 2^{14} B (16 kB). Therefore, we implemented a fragmentation mechanism for messages containing a signature larger than the single record size, primarily relevant for the speed-optimized variant of SPHINCS^{+} (see Table 2).

5 MEASUREMENTS AND EVALUATION

We consider two types of client devices in our experiments: (i) *Notebook* equipped with an Intel Core i5-6300U quad-core processor running at 2.4 GHz and 8 GB of RAM; (ii) Raspberry Pi 3 Model B equipped with an ARM Cortex-A53 quad-core processor running at 1.2 GHz and 1 GB of RAM (denoted *Embedded* in the following). The first represents a typical client device in, e. g., company networks, whereas the latter represents an embedded client in typical IoT settings. Both devices are connected via their Ethernet interface to the local access point.

[6]https://csrc.nist.gov/Projects/post-quantum-cryptography/round-3-submissions.

[7]https://github.com/XMSS/xmss-reference.

Our servers are set up as remote Azure virtual machines (Standard D2as_v4) at three different locations (West Europe, East US, East Australia). They are equipped with two virtual CPUs based on an AMD EPYC7452 processor running at 2.35 GHz and 8 GB of RAM. The remote servers are located at increasing distances to our local clients based in West Europe, as the following round-trip-time (RTT) statistics show:

- Client ⟶ West Europe: 26.1 ms (mean RTT; hop count: *15*).
- Client ⟶ East US: 105 ms (mean RTT; hop count: *19*).
- Client ⟶ East Australia: 258 ms (mean RTT; hop count: *21*).

Client and Server Programs. Our client and server implement a mutually authenticated TLS 1.3 handshake using the full 1-RTT mode without pre-shared key resumption; the selected ciphersuite is TLS_AES_256_GCM_SHA384. Library and programs are compiled using gcc's O3 flag. Note that we do not consider caching of intermediate certificates, as well as extensions commonly found in WebPKI settings (SCT, OCSP, CRL, SAN), since a few extra byte of certificate data do not fundamentally change handshake performance [49, 50].

Measured Parameters and Objectives. In order to evaluate the feasibility of mixed certificate chains as migration strategy for TLS 1.3 towards PQC, we conduct several time-related as well as cost-related measurements. We introduce three measurement points in our client and server programs as outlined in Fig. 1. With *time to first byte (TTFB)* we measure the latency a client experiences from initiating the TLS handshake to receiving the first byte of encrypted application data (excluding the initial TCP handshake).[8] In order to assess the individual cryptographic load client and server experience during the TLS handshake, we also measure *client* and *server connect time* by observing the time when the handshake is initiated from their point of view up until sending their respective Finished message. Apart from these time-related measurements we also investigate the impact mixed certificate chains have on the following cost metrics: *communication size*, resulting *code size* of the PQC-enabled wolfSSL library, and *peak memory usage* of client and server programs for each of our test cases.

In our measurements the conventional use case *EDS-EDH* serves as baseline for the comparison between mixed certificate chains and other evaluated control certificate chains. Note that with our time-related measurements we do not expect to see performance improvements compared to the conventional and PQC-based control use cases, instead we aim to assess the feasibility of our approach as practical migration strategy towards enabling post-quantum authentication for TLS.

5.1 Standalone Performance of Evaluated Cryptographic Primitives

At first, we evaluated standalone performance characteristics for the signature and key establishment algorithms used in this work. We modified wolfCrypt's internal benchmark tool to integrate the newly added PQC schemes. Table 2 shows the averaged benchmark results on the three experimental platforms. In addition, we report

[8]To account for the time the server spends authenticating the client, we have the server send the first message of encrypted application data.

Table 2: Overview of evaluated cryptographic schemes including performance benchmarks on target platforms (time rounded to three significant figures).

Algorithm (Parameter)	*NIST Level*	*Sizes (byte)*		*Performance (ms)*			
					Embed.	*Notebook*	*Server*
				Key Encapsulation Schemes			
ECDHE (SECP256R1)	×	**sk:**	32	**gen:**	1.52	0.0920	0.0910
		pk:	65	**agmt:**	4.40	0.255	0.271
Kyber [6] (Kyber512)	1	**sk:**	1632	**gen:**	0.572	0.0380	0.0330
		pk:	800	**enc:**	0.772	0.0440	0.0370
		ct:	768	**dec:**	0.772	0.0490	0.0430
				Signature Schemes			
ECDSA (SECP256R1)	×	**sk:**	32	**gen:**	1.52	0.0920	0.0910
		pk:	65	**sign:**	1.94	0.116	0.119
		sig:	73	**vfy:**	4.85	0.285	0.301
Dilithium [7] (Dilithium-2)	2	**sk:**	2544	**gen:**	2.04	0.107	0.0880
		pk:	1312	**sign:**	11.9	0.414	0.389
		sig:	2420	**vfy:**	2.21	0.121	0.0990
Falcon [26] (Falcon-512)	1	**sk:**	1281	**gen:**	158	20.1	16.9
		pk:	897	**sign:**	35.7	5.90	4.91
		sig:	666	**vfy:**	0.435	0.0420	0.0310
SPHINCS$^+$ [5] (SHA-256-128s -simple)	1	**sk:**	64	**gen:**	473	114	93.6
		pk:	32	**sign:**	3540	866	710
		sig:	7856	**vfy:**	3.53	0.876	0.678
SPHINCS$^+$ [5] (SHA-256-128f -simple)	1	**sk:**	64	**gen:**	7.33	1.75	1.47
		pk:	32	**sign:**	183	43.3	36.4
		sig:	17,088	**vfy:**	10.2	2.46	2.05
XMSS [30] (XMSS-SHA2 -10-256)	–[9]	**sk:**	36	**gen:**	11,300	2190	1870
		pk:	64	**sign:**	50.1	9.70	8.26
		sig:	2500	**vfy:**	6.49	1.20	1.03

Notation: secret key (sk), public key (pk), ciphertext (ct), signature (sig), key generation (gen), key agreement (agmt), encapsulation (enc), decapsulation (dec), sign (sign), verify (vfy).

[9] Since XMSS is not part of NIST's PQC standardization process, it does not have an official NIST security level.

the size characteristics of all evaluated schemes: key size and ciphertext/signature size. Note that, for the ECC schemes (ECDHE and ECDSA) we enabled wolfSSL's optimized implementation based on its multi-precision math library.

Key Establishment Schemes. All three functions of the post-quantum KEM Kyber (key generation, encapsulation, and decapsulation) outperform ECDHE on every platform. In fact, ECDHE's key agreement is slower by at least a factor of five. As a result, Kyber's comparatively small public key and ciphertext sizes (≤1 kB) and fast performance make it a promising PQC scheme.

Signature Schemes. As expected the two lattice-based signature schemes Dilithium and Falcon perform better than their three hash-based counterparts on all platforms. In all evaluated hash-based schemes the sign operation is very expensive. For example, a single sign operation takes about 3.5 s using the size-optimized SPHINCS$^+$ (SPs) instantiation on the embedded target compared to only 1.9 ms with ECDSA. Considering verification, all hash-based schemes are

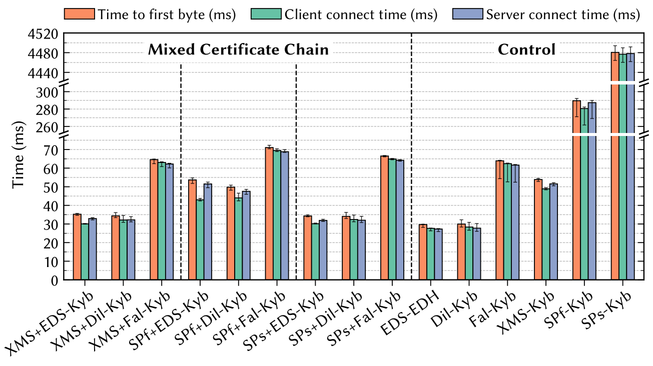


Figure 2: Time measurements in the local setting (median times in ms; error bars for 25th and 75th percentile): Embedded → Notebook.

within an acceptable range, whereas Dilithium and Falcon even outperform ECDSA.

Besides that, we observe that ECDSA has faster signing operations than verification as opposed to the two lattice-based schemes, where verification is considerably faster than signing. Since verification is required more often in a typical TLS handshake, we expect to see a feasible performance in case client or server make use of a lattice-based signature scheme.

Nevertheless, this benchmark shows that PQC signature schemes will generally affect the performance of TLS handshakes more than PQC key establishment (in case of fast lattice-based KEMs). Note that in signature schemes, key generation is required only in case a new long term signing key pair needs to be generated.

5.2 Connection Establishment Times

For each of the two client devices (Notebook and Embedded) and each of the 15 certificate chain test cases (9 mixed and 6 control certificate chains) we perform 1000 TLS 1.3 handshakes with each of the three remote servers. In addition, we perform the same experiment in a local network, where the embedded device acts as client and the notebook as server. This local experiment allows us to better analyze the impact of the actual cryptographic operations without hardly any network latency (mean RTT: 0.919 ms). In the following, we focus our analysis on median TTFB, since it includes all operations related to server as well as client authentication.

Local Server. Fig. 2 shows the results of our local experiment. The control combinations *EDS-EDH* and *Dil-Kyb* show the best performance with median TTFB at 29.7 ms for the former and 30.0 ms for the latter. Nevertheless, combining hash-based signature schemes with ECDSA — as part of the first step in our migration strategy — shows very promising results. Compared to the conventional control case *EDS-EDH* the increase in median TTFB remains feasible, especially in the following two combinations: *SPs+EDS-Kyb* (+4.64 ms) and *XMS+EDS-Kyb* (+5.57 ms).

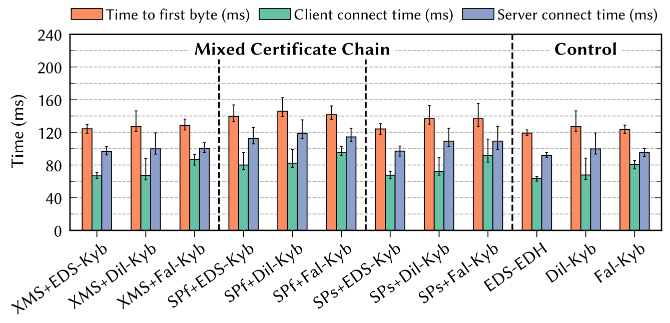
Considering the final step of our strategy, we observe a similar small increase in median TTFB. In that case the fastest two PQC-only mixed certificate chains are: *XMS+Dil-Kyb* with an increase of +4.69 ms and *SPs+Dil-Kyb* with an increase of +4.41 ms, which is also the smallest increase of all evaluated mixed certificate chains in this setting.

Furthermore, our approach of using SPHINCS$^+$ only at the root CA level gives a significant improvement compared to certificate chains from the control group that use SPHINCS$^+$ across the entire chain. In fact, we observe an average decrease in TTFB of 79.9 % in case of *SPf-Kyb* and 99.0 % in case of *SPs-Kyb*. The slow handshake times in the pure SPHINCS$^+$ settings are largely due to the slow signing operation of the two particular SPHINCS$^+$ signature schemes (see Table 2), especially in case of *SPs-Kyb* where we experienced a median TTFB of 4.48 s.

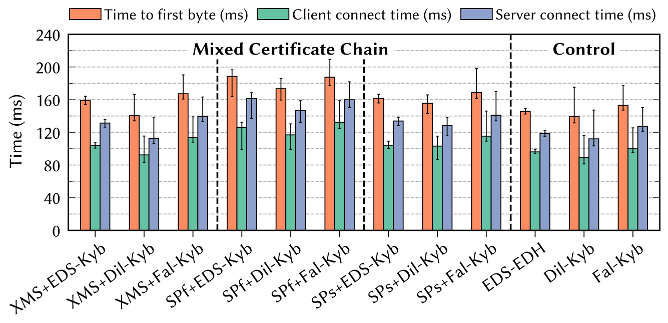
Remote Servers. Next, we analyze our mixed certificate chains under realistic network conditions where our two client devices connect to the aforementioned remote servers over a public network. Fig. 3 shows the six handshake related time measurements of this experiment. Despite the increase in message sizes due to larger certificates, signatures, and ciphertexts we see a success rate of 100 % in all performed handshake experiments. Note that we do not report measurements for the following three hash-based control combinations: SPf-Kyb, SPs-Kyb, and XMS-Kyb. As we showed in the local network setting SPHINCS$^+$-only certificate chains are impractical for most applications, because of their slow handshake times. In case of a certificate chain purely based on XMSS, we do not trust embedded clients and load-balanced server farms to securely manage the state of XMSS private keys.

Before we assess the individual impact of our two-step migration strategy, we look at the general feasibility of the evaluated mixed certificate chains for TLS 1.3. We observe a feasible increase in median TTFB across all nine evaluated mixed certificate chains compared to the ECC-based control handshake (*EDS-EDH*). For connections to the server located in West Europe this increase is +12.4 % (Notebook) and +14.4 % (Embedded) respectively. The same holds true for connections to the server in East US: +2.77 % (Notebook) and +3.66 % (Embedded). As expected, the increase becomes even less significant when connecting to the server located in East Australia: +1.52 % (Notebook) and +1.06 % (Embedded). This demonstrates that the impact of individual cryptographic operations on all reported connection establishment times decreases with longer client-server distances due to the higher round-trip times. As a consequence, we focus the subsequent evaluation of time-related handshake measurements when connecting to servers located in West Europe (Figs. 3a and 3b) and East US (Figs. 3c and 3d).

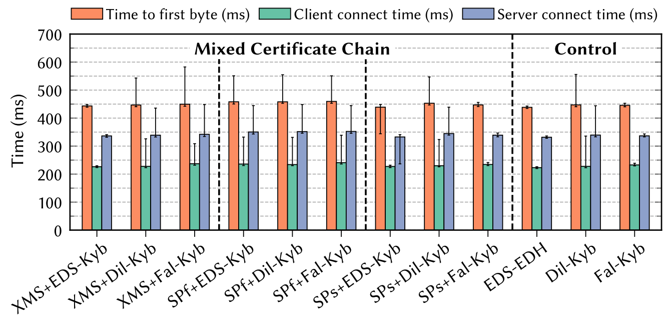
In case the notebook connects to the server located in West Europe (Fig. 3a) we observe the smallest increase in median TTFB compared to *EDS-EDH* with mixed certificate chains using ECDSA at ICA and EE level — the first step of our proposed strategy. Because of their comparatively small certificate chain sizes (see Table 5 in Appendix A), fast conventional sign operation for handshake signatures, and balanced verify operations, the combinations *SPs+EDS-Kyb* (+5.09 ms) and *XMS+EDS-Kyb* (+5.27 ms) seem promising transitional candidates, as in the local setting. Regarding the final step towards complete post-quantum security, the combination *XMS+Dil-Kyb* shows the fastest TTFB with an increase of +7.99 ms. For both migration steps, we observe the same behavior when connecting to the server in East US (Fig. 3c).



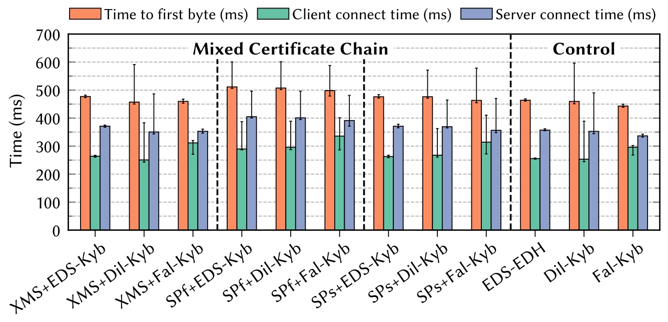
(a) Notebook (West Europe) ⟶ Server (West Europe).



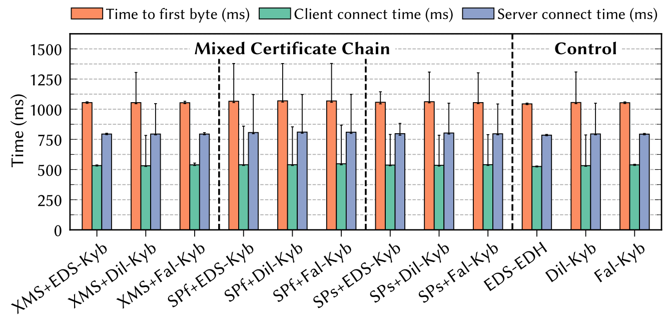
(b) Embedded (West Europe) ⟶ Server (West Europe).



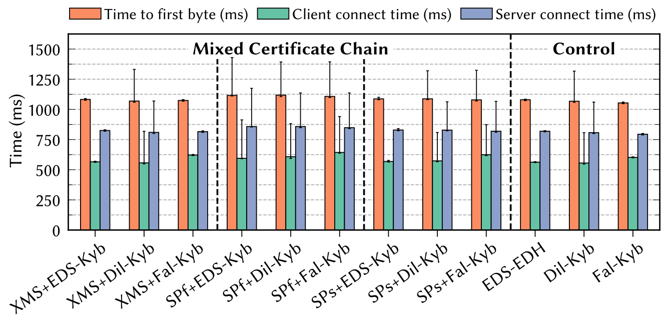
(c) Notebook (West Europe) ⟶ Server (East US).



(d) Embedded (West Europe) ⟶ Server (East US).



(e) Notebook (West Europe) ⟶ Server (East Australia).



(f) Embedded (West Europe) ⟶ Server (East Australia).

Figure 3: Time measurements under realistic network conditions at increasing client-server distances (median times in ms; error bars for 25th and 75th percentile).

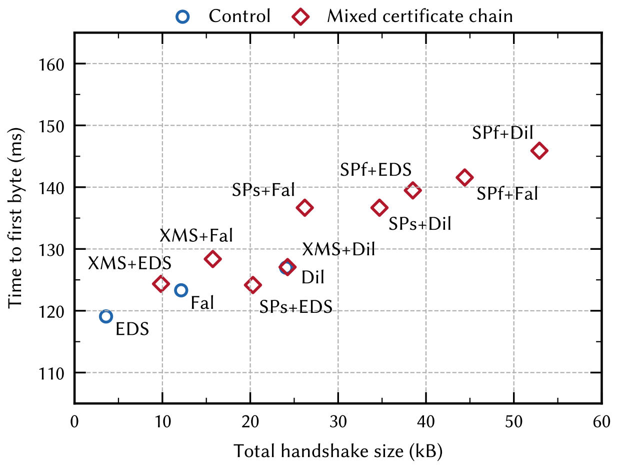
We find similar results, in case the embedded device acts as client. Again, the mixed certificate chains of the first migration step (hash-based schemes combined with ECDSA) show a feasible median TTFB compared to the conventional control combination *EDS-EDH* when connecting to the server based in West Europe (Fig. 3b): *XMS+EDS-Kyb* (+12.9 ms) and *SPs+EDS-Kyb* (+15.7 ms).

In the final step of our strategy, the combination *XMS+Dil-Kyb* even outperforms *EDS-EDH* (−5.39 ms). We attribute this behavior to scheme dependent computational fluctuations, especially in case of Dilithium [32], as well as higher traffic load caused by larger message sizes as indicated by the error bars in Fig. 3. These findings are also replicated for the server based in East US (Fig. 3d).

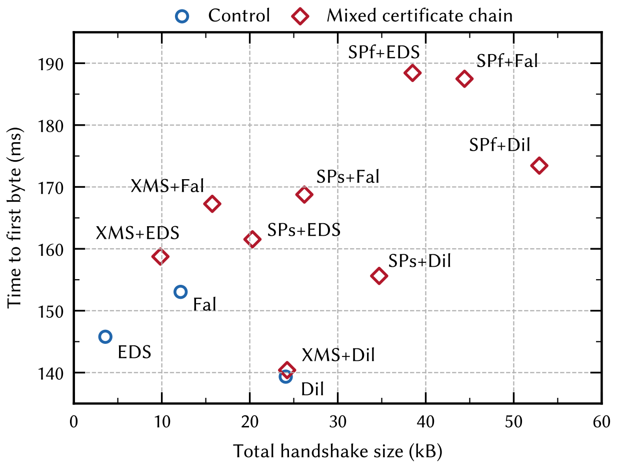
5.3 Communication Size

Fig. 4 shows the size of all messages that client and server exchange during their handshake (total handshake size) in contrast to median TTFB. In all evaluated test cases we observe an increase in communication size as a result of the significant increase in certificate chain size (see Table 5 in Appendix A) and, to a lesser degree, as a result of larger signatures and key exchange material (see Table 2).

In addition to its promising TTFB, the scheme combination *XMS+EDS-Kyb* of our first migration step leads to the smallest total handshake size (10.1 kB) among all investigated mixed certificate chains. On the other hand, our three combinations containing the SPHINCS$^+$ variant *SPf* at the root CA level show the highest total handshake sizes with a minimum of 39.4 kB (*SPf+EDS-Kyb*) up to 54.2 kB (*SPf+Dil-Kyb*).



(a) Notebook (West Europe) → Server (West Europe).



(b) Embedded (West Europe) → Server (West Europe).

Figure 4: Total handshake size vs. time to first byte of evaluated scheme combinations (KEX omitted for readability).

Regarding the mixed certificate chains that present our final migration step, the combination *XMS+Fal-Kyb* has the lowest total handshake size with 16.1 kB. The combination *XMS+Dil-Kyb*, however, has a slightly larger total handshake size of 24.8 kB, which is virtually the same as the purely Dilithium-based combination *Dil-Kyb* (24.7 kB), but compared to *XMS+Fal-Kyb* it has shown a more promising median TTFB in our time-related measurements.

5.4 Code Size

We compiled wolfSSL as static library for each of our test cases and linked it with the corresponding client or server programs. To allow for a fair comparison of the libraries' code size we only enabled the cryptographic operations relevant for the respective test case.

In Table 3 we report the code size of the wolfSSL library for each of our client device and all 15 scheme combinations. Note that ECC, i. e., ECDSA and ECDHE, is mandatory when building wolfSSL as TLS 1.3 client or server, therefore its functionality is included in all of the libraries. Nevertheless, we limit the impact of ECC on the resulting code size by only enabling the required elliptic curve SECP256R1. Other configuration options that we require in all library builds are the relevant symmetric ciphers, hash functions, and TLS 1.3 functionality including X.509 certificate handling; we disable all other non-relevant options. As we built the library optimized for speed (-O3) rather than size, we observe larger code sizes than typical embedded libraries (≤ 100 kB) [14]. We report total code size consisting of data, bss, and text sections. Due to the different platforms of our client devices (Cortex-A53 vs. Core i5) the code size of our libraries varies. However, we are more interested in the relative impact our mixed certificate chains have on final code size rather than absolute size. Besides that, most of the code size will end up in static flash memory which is typically not a limiting factor even in embedded systems.

Table 3: Total code size of wolfSSL library for evaluated scheme combinations (reported in kB; rounded to three significant figures).

	Embedded			*Notebook*		
Algorithm Combination	*Code Size (kB)*	*Overhead (kB)*	*Δ (%)*	*Code Size (kB)*	*Overhead (kB)*	*Δ (%)*
EDS-EDH	393	—	—	377	—	—
Dil-Kyb	633	240	+61.2	484	107	+28.5
Fal-Kyb	727	334	+85.1	569	192	+51.0
XMS(+EDS)-Kyb	602	209	+53.2	448	71.5	+19.0
XMS+Dil-Kyb	649	257	+65.3	503	126	+33.5
XMS+Fal-Kyb	743	350	+89.2	588	211	+56.0
SPf(+EDS)-Kyb	607	215	+54.6	457	79.7	+21.2
SPf+Dil-Kyb	655	262	+66.8	511	134	+35.7
SPf+Fal-Kyb	748	356	+90.6	596	219	+58.2
SPs(+EDS)-Kyb	607	214	+54.6	456	79.4	+21.1
SPs+Dil-Kyb	654	262	+66.7	511	134	+35.5
SPs+Fal-Kyb	748	355	+90.6	596	219	+58.1

On both client platforms, the integration of hash-based signature schemes and Kyber leads to the smallest overhead in code size, which automatically enables combinations of the initial migration step; with XMSS having the lowest impact: 53.2 % (Embedded) and 19.0 % (Notebook). As sign operations of hash-based schemes are not required on client and server side in our approach, the resulting code size can be decreased further by only implementing the corresponding verification operation. As Kyber and Dilithium use wolfSSL's implementation of SHA3, combining hash-based signature schemes with Dilithium for a post-quantum secure TLS handshake leads to acceptable overhead. In case of the embedded device code size increases in average by another 12.1 % and in case of the notebook by another 14.5 %. Due to Falcon's larger overall code size its impact is higher when combined with hash-based schemes from the first migration step, with an average additional increase of 36.0 % (Embedded) and 37.0 % (Notebook).

As servers are typically equipped with sufficient amount of memory, we enable all cryptographic algorithms relevant for our test cases in the server's library — resulting in a total code size of 798 kB.

Table 4: Peak memory usage of client program on our embedded platform for evaluated scheme combinations (reported in kB; rounded to three significant figures).

	Algorithm Combination	*Client: Embedded*			
		heap	*stack*	*total*	Δ (%)
Control	EDS-EDH	107	62.8	170	–
	Dil-Kyb	128	111	240	+41.2
	Fal-Kyb	119	102	221	+30.2
	XMS-Kyb	122	63.5	186	+9.49
	SPf-Kyb	167	61.3	228	+34.4
	SPs-Kyb	129	61.9	191	+12.4
Mixed Certificate Chain	XMS+EDS-Kyb	115	63.3	178	+5.19
	XMS+Dil-Kyb	129	111	240	+41.4
	XMS+Fal-Kyb	119	102	221	+30.2
	SPf+EDS-Kyb	131	68.3	199	+17.2
	SPf+Dil-Kyb	142	111	253	+49.0
	SPf+Fal-Kyb	135	102	236	+39.3
	SPs+EDS-Kyb	119	63.3	182	+7.43
	SPs+Dil-Kyb	133	111	244	+43.7
	SPs+Fal-Kyb	126	102	227	+34.0

5.5 Peak Memory Usage

We use Valgrind's massif tool[10] to measure peak heap and stack usage of our client and server programs for each mixed certificate chain and control group setting. Heap usage is mostly affected by dynamic memory allocations related to buffers for sending messages. Due to larger certificate chain sizes sent in the Certificate message as well as larger signatures sent in the CertificateVerify message, we see an increase in peak heap usage across all test cases on all platforms. Other notable allocations are related to loading the private key of the corresponding signature scheme, the DER-decoded certificate chain, the cryptographic material related to the ephemeral key exchange scheme, and other smaller wolfSSL context buffers. As a result, combinations of hash-based signature schemes and ECDSA, which are a part of our first migration step, show the smallest increase in peak heap usage benefiting from small public keys and signatures of ECDSA. Regarding our second migration step, the combination of hash-based schemes and Dilithium leads to the highest increase in heap usage because of Dilithium's larger keys and signature sizes with an average increase of 39.9 % across all platforms compared to the control combination *EDS-EDH*.

Note that peak stack usage is not influenced by the size of the cryptographic material. Instead, it depends largely on the implementation of the underlying hard mathematical problem. In our analysis we observe the highest peak stack usage when Dilithium is used in the TLS 1.3 handshake. Nevertheless, we expect future implementations of standardized PQC schemes to offer size-optimized implementations potentially reducing their stack usage.

However, as heap and stack both reside on RAM it is very important to achieve an overall low memory consumption for both, since this is typically a scarce resource — especially on embedded systems. Table 4 shows the results of our memory analysis for the embedded client. Peak memory usage of all our 15 test cases is below 256 kB and hence within acceptable range of typical embedded systems. For our memory analysis of the client program on the notebook and server program see Table 6 in Appendix A.

[10]https://valgrind.org.

5.6 Migration Strategy

We show that our proposed migration strategy based on mixed certificates is feasible in TLS 1.3. The combination of hash-based signature schemes, especially XMSS, with conventional ECDSA as the first step in our migration strategy gives promising results. On both evaluated client platforms XMS+EDS-Kyb shows fast connection establishment times, lowest overhead in communication and code size, as well as lowest memory usage of all evaluated mixed certificate chains. In case state management is not desired at the root CA level, *SPs+EDS-Kyb* is another well-balanced alternative, with similar results in time to first byte, code size, and memory usage but larger communication overhead.

Considering the final step in our migration strategy, we observe that the scheme combination *XMS+Dil-Kyb* is feasible for both client devices — even outperforming the control combination *EDS-EDH* on the embedded device. It also comes with acceptable overhead in code size; however, its impact on RAM is significant due to high peak stack usage of Dilithium's reference implementation. Nevertheless, we expect that future implementations of Dilithium will offer stack optimized implementations. Alternatively, the combination *XMS+Fal-Kyb* leads to slightly higher connection establishment times but still tolerable for short-lived TLS connections. However, it has smaller communication size and lower peak memory usage compared to *XMS+Dil-Kyb*.

While other scheme combinations can be used in our migration strategy, we see a combination of hash- and lattice-based schemes favorable for the following two reasons:

(1) Hash-based schemes at the root CA level already offer conservative security against current as well as future quantum-computer attacks. They have been analyzed for decades and are based only on minimal security assumptions. Furthermore, stateful hash-based schemes are the first PQC schemes to be specified in form of an IETF RFC.
(2) While lattice-based schemes are still fairly new and thus less tested, they have arguably attracted the most attention in academia over the last few years. Their balanced performance characteristics make them promising candidates for standardization.

6 CONCLUSION

In this paper, we presented a novel two-step migration strategy towards upcoming standards of PQC signature schemes based on the concept of mixed certificate chains. In a first intermediate step, we combined trusted hash-based signature schemes at the root CA level with the conventional scheme ECDSA to prepare for a seamless transition to PQC-based authentication in TLS. The final migration step aims for complete post-quantum security, where we evaluated the combination of hash-based signature schemes at the root CA level with lattice-based schemes at the ICA and EE level alongside the post-quantum KEM Kyber. In fact, using hash-based signature schemes only at the root CA level allows us to alleviate most of their drawbacks: slow signing performance, large signatures, and state management in case of XMSS.

Considering the first step of our proposed strategy, the combination *XMS+EDS-Kyb* showed promising results on both our client devices (Notebook and Embedded) with feasible connection establishment times, lowest overhead in communication and code size, as well as lowest memory usage of all evaluated mixed certificate chains. Moreover, XMSS is already specified in an IETF RFC and recommended by NIST as well as other standardization bodies.

The PQC-only combination *XMS+Dil-Kyb* displayed promising results as part of the final migration step. The measured time to first byte is feasible for both client platforms and has acceptable overhead in code size. Its impact on RAM, however, is significantly higher compared to other evaluated scheme combinations.

Furthermore, the reported results complement existing experimental studies [48–50]: We evaluated the impact of PQC signature schemes and PQC key establishment on the TLS 1.3 handshake by working with current round three reference implementations and by assessing mutual authentication under real network conditions.

While we focused on the network protocol TLS 1.3 in this work, it is of interest to investigate the impact of our proposed migration steps on other protocols such as SSH or IKEv2/IPsec in future work. We hope our practical results facilitate research into the concept of mixed certificate chains and ease the migration towards the next generation of cryptography.

ACKNOWLEDGMENTS

Partially funded by the German Federal Ministry of Education and Research (BMBF) under the project "FLOQI" (ID 16KIS1074). In addition, we gratefully acknowledge support by the German Federal Ministry of Education and Research and the Hessian Ministry of Higher Education, Research, Science and the Arts within their joint support of the National Research Center for Applied Cybersecurity ATHENE. We also thank Efstathia Katsigianni, Evangelos Karatsiolis, Johannes Roth, and Enrico Thomae for their helpful discussions and Shaker Aljallad for his implementation contributions.

REFERENCES

[1] M. R. Albrecht, D. J. Bernstein, T. Chuo, C. Cid, J. Gilcher, T. Lange, V. Maram, I. von Maurich, R. Miscozki, R. Niederhagen, K. G. Paterson, E. Persichetti, C. Peters, P. Schwabe, N. Sendrier, J. Szefer, C. J. Tjhai, M. Tomlinson, and W. Wang. 2020. Classic McEliece: conservative code-based cryptography. NIST PQC standardization: Round 3.

[2] E. Alkim, J. W. Bos, L. Ducas, P. Longa, I. Mironov, M. Naehrig, V. Nikolaenko, C. Peikert, A. Raghunathan, and D. Stebila. 2020. FrodoKEM: Learning With Errors Key Encapsulation. NIST PQC standardization: Round 3.

[3] N. Aragon, P. S. L. M. Barreto, S. Bettaieb, L. Bidoux, O. Blazy, J.-C. Deneuville, P. Gaborit, S. Ghosh, S. Gueron, T. Güneysu, C. A. Melchor, R. Misoczki, E. Persichetti, N. Sendrier, J.-P. Tillich, V. Vasseur, and G. Zémor. 2020. BIKE: Bit Flipping Key Encapsulation. NIST PQC standardization: Round 3.

[4] F. Arute et al. 2019. Quantum supremacy using a programmable superconducting processor. *Nature* 574, 7779 (2019), 505–510.

[5] J.-P. Aumasson, D. J. Bernstein, W. Beullens, C. Dobraunig, M. Eichlseder, S. Fluhrer, S.-L. Gazdag, A. Hülsing, P. Kampanakis, S. Kölbl, T. Lange, M. M. Lauridsen, F. Mendel, R. Niederhagen, C. Rechberger, J. Rijneveld, P. Schwabe, and B. Westerbaan. 2020. SPHINCS$^+$: Submission to the NIST post-quantum project. NIST PQC standardization: Round 3.

[6] R. Avanzi, J. Bos, L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, J. M. Schank, P. Schwabe, G. Seiler, and D. Stehlé. 2020. CRYSTALS-Kyber. NIST PQC standardization: Round 3.

[7] S. Bai, L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, P. Schwabe, G. Seiler, and D. Stehlé. 2020. CRYSTALS-Dilithium. NIST PQC standardization: Round 3.

[8] R. Barnes, J. Hoffman-Andrews, D. McCarney, and J. Kasten. 2019. Automatic Certificate Management Environment (ACME). RFC 8555.

[9] A. Basso, J. M. Bermudo Mera, J.-P. D'Anvers, A. Karmakar, S. S. Roy, M. Van Beirendonck, and F. Vercauteren. 2020. Saber: Mod-LWR based KEM. NIST PQC standardization: Round 3.

[10] D. J. Bernstein, B. B. Brumley, M.-S. Chen, C. Chuengsatiansup, T. Lange, A. Marotzke, B.-Y. Peng, N. Tuveri, C. van Vredendaal, and B.-Y. Yang. 2020. NTRU Prime. NIST PQC standardization: Round 3.

[11] W. Beullens. 2021. Improved Cryptanalysis of UOV and Rainbow. In *Advances in Cryptology – EUROCRYPT 2021*, A. Canteaut and F.-X. Standaert (Eds.). Springer, Cham, 348–373.

[12] J. W. Bos, C. Costello, M. Naehrig, and D. Stebila. 2015. Post-Quantum Key Exchange for the TLS Protocol from the Ring Learning with Errors Problem. In *2015 IEEE Symposium on Security and Privacy*. IEEE, 553–570.

[13] M. Braithwaite. 2016. Experimenting with Post-Quantum Cryptography. https://security.googleblog.com/2016/07/experimenting-with-post-quantum.html

[14] K. Bürstinghaus-Steinbach, C. Krauß, R. Niederhagen, and M. Schneider. 2020. Post-Quantum TLS on Embedded Systems: Integrating and Evaluating Kyber and SPHINCS+ with Mbed TLS. In *Asia Conference on Computer and Communications Security (ASIA CCS '20)*. ACM, 841–852.

[15] BSI. 2020. Migration zu Post-Quanten-Kryptografie. https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Krypto/Post-Quanten-Kryptografie Handlungsempfehlungen des BSI; available only in German.

[16] M. Campagna and E. Crockett. 2021. *Hybrid Post-Quantum Key Encapsulation Methods (PQ KEM) for Transport Layer Security 1.2 (TLS)*. Internet-Draft draft-campagna-tls-bike-sike-hybrid-06. IETF. https://datatracker.ietf.org/doc/html/draft-campagna-tls-bike-sike-hybrid-06 Work in progress.

[17] A. Casanova, J.-C. Faugère, G. Macario-Rat, J. Patarin, L. Perret, and J. Ryckeghem. 2020. GeMSS: A Great Multivariate Short Signature. NIST PQC standardization: Round 3.

[18] Y.-A. Chang, M.-S. Chen, J.-S. Wu, and B.-Y. Yang. 2014. Postquantum SSL/TLS for Embedded Systems. In *Service-Oriented Computing and Applications*. IEEE, 266–270.

[19] M. Chase, D. Derler, S. Goldfeder, D. Kales, J. Katz, V. Kolesnikov, C. Orlandi, S. Ramacher, C. Rechberger, D. Slamanig, X. Wang, and G. Zaverucha. 2020. The Picnic Signature Algorithm. NIST PQC standardization: Round 3.

[20] C. Chen, O. Danba, J. Hoffstein, A. Hülsing, J. Rijneveld, J. M. Schank, T. Saito, P. Schwabe, W. Whyte, K. Xagawa, T. Yamakawa, and Z. Zhang. 2020. NTRU. NIST PQC standardization: Round 3.

[21] V. Combs. 2020. Expiring security certificates may start shutting down IoT devices. https://www.techrepublic.com/article/expiring-security-certificates-may-start-shutting-down-iot-devices/

[22] D. A. Cooper, D. C. Apon, Q. H. Dang, M. S. Davidson, M. J. Dworkin, and C. A. Miller. 2020. *Recommendation for Stateful Hash-Based Signature Schemes: NIST Special Publication 800-208*. NIST.

[23] J. Ding, M.-S. Chen, M. Kannwischer, J. Patarin, A. Petzoldt, D. Schmidt, and B.-Y. Yang. 2020. Rainbow. NIST PQC standardization: Round 3.

[24] I. Dinur. 2021. Cryptanalytic Applications of the Polynomial Method for Solving Multivariate Equation Systems over GF(2). In *Advances in Cryptology – EUROCRYPT 2021*, A. Canteaut and F.-X. Standaert (Eds.). Springer, Cham, 374–403.

[25] ETSI. 2020. Quantum-Safe Cryptography (QSC). https://www.etsi.org/technologies/quantum-safe-cryptography

[26] P.-A. Fouque, J. Hoffstein, P. Kirchner, V. Lyubashevsky, T. Pornin, T. Prest, T. Ricosset, G. Seiler, W. Whyte, and Z. Zhang. 2020. Falcon: Fast-Fourier Lattice-based compact signatures over NTRU. NIST PQC standardization: Round 3.

[27] O. Garcia-Morchon, S. Kumar, and M. Sethi. 2019. Internet of Things (IoT) Security: State of the Art and Challenges. RFC 8576.

[28] S. Helme. 2020. The Impending Doom of Expiring Root CAs and Legacy Clients. https://scotthelme.co.uk/impending-doom-root-ca-expiring-legacy-clients/

[29] R. Holz, J. Amann, A. Razaghpanah, and N. Vallina-Rodriguez. 2019. The Era of TLS 1.3: Measuring Deployment and Use with Active and Passive Methods. *CoRR* abs/1907.12762 (2019).

[30] A. Hülsing, D. Butin, S.-L. Gazdag, J. Rijneveld, and A. Mohaisen. 2018. XMSS: Extended Merkle Signature Scheme. RFC 8391.

[31] D. Jao, R. Azarderakhsh, M. Campagna, C. Costello, L. De Feo, B. Hess, A. Hutchinson, A. Jalali, K. Karabina, B. Koziel, B. LaMacchia, P. Longa, M. Naehrig, G. Pereira, J. Renes, V. Soukharev, and D. Urbanik. 2020. Supersingular Isogeny Key Encapsulation. NIST PQC standardization: Round 3.

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

[33] K. Kwiatkowski and L. Valenta. 2019. The TLS Post-Quantum Experiment. https://blog.cloudflare.com/the-tls-post-quantum-experiment/

[34] A. Langley. 2018. Post-quantum confidentiality for TLS. https://www.imperialviolet.org/2018/04/11/pqconftls.html

[35] A. Langley. 2019. Real-world measurements of structured-lattices and supersingular isogenies in TLS. https://www.imperialviolet.org/2019/10/30/pqsivssl.html

[36] F. Maggi, R. Vosseler, and D. Quarte. 2018. The Fragility of Industrial IoT's Data Backbone: Security and Privacy Issues in MQTT and CoAP Protocols. https://documents.trendmicro.com/assets/white_papers/wp-the-fragility-of-industrial-IoTs-data-backbone.pdf

[37] D. McGrew, M. Curcio, and S. Fluhrer. 2019. Leighton-Micali Hash-Based Signatures. RFC 8554.

[38] D. McGrew, P. Kampanakis, S. Fluhrer, S.-L. Gazdag, D. Butin, and J. Buchmann. 2016. State Management for Hash-Based Signatures. In *Security Standardisation Research*, L. Chen, D. McGrew, and C. Mitchell (Eds.). Springer, 244–260.

[39] C. A. Melchor, N. Aragon, S. Bettaieb, L. Bidoux, O. Blazy, J. Bos, J.-C. Deneuville, A. Dion, P. Gaborit, J. Lacan, E. Persichetti, J.-M. Robert, P. Véron, and G. Zémor. 2020. Hamming Quasi-Cyclic (HQC). NIST PQC standardization: Round 3.

[40] D. Moody, G. Alagic, D. C. Apon, D. A. Cooper, Q. H. Dang, J. M. Kelsey, Y.-K. Liu, C. A. Miller, R. C. Peralta, R. A. Perlner, A. Y. Robinson, D. C. Smith-Tone, and J. Alperin-Sheriff. 2020. Status Report on the Second Round of the NIST Post-Quantum Cryptography Standardization Process.

[41] NIST. 2016. Submission Requirements and Evaluation Criteria for the Post-Quantum Cryptography Standardization Process. https://csrc.nist.gov/CSRC/media/Projects/Post-Quantum-Cryptography/documents/call-for-proposals-final-dec-2016.pdf

[42] C. Paquin, D. Stebila, and G. Tamvada. 2020. Benchmarking Post-quantum Cryptography in TLS. In *Post-Quantum Cryptography*, J. Ding and J.-P. Tillich (Eds.). Springer, 72–91.

[43] S. Paul and P. Scheible. 2020. Towards Post-Quantum Security for Cyber-Physical Systems: Integrating PQC into Industrial M2M Communication. In *ESORICS 2020*, L. Chen, N. Li, K. Liang, and S. Schneider (Eds.). Springer, 295–316.

[44] S. Paul, F. Schick, and J. Seedorf. 2021. TPM-Based Post-Quantum Cryptography: A Case Study on Quantum-Resistant and Mutually Authenticated TLS for IoT Environments. In *The 16th International Conference on Availability, Reliability and Security (ARES 2021)*. ACM.

[45] U. Pohlmann and A. Sikora. 2018. Practical security recommendations for building OPC UA applications. https://opcconnect.opcfoundation.org/2018/06/practical-security-guidelines-for-building-opc-ua-applications/

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

[47] J. A. Salowey, S. Turner, and C. A. Wood. 2019. TLS 1.3: One Year Later. https://www.ietf.org/blog/tls13-adoption

[48] P. Schwabe, D. Stebila, and T. Wiggers. 2020. Post-Quantum TLS Without Handshake Signatures. In *Computer and Communications Security (CCS '20)*, J. Ligatti, X. Ou, J. Katz, and G. Vigna (Eds.). ACM, 1461–1480.

[49] D. Sikeridis, P. Kampanakis, and M. Devetsikiotis. 2020. Assessing the Overhead of Post-Quantum Cryptography in TLS 1.3 and SSH. In *Emerging Networking EXperiments and Technologies (CoNEXT '20)*. ACM, 149–156.

[50] D. Sikeridis, P. Kampanakis, and M. Devetsikiotis. 2020. Post-Quantum Authentication in TLS 1.3: A Performance Study. In *NDSS 2020*. The Internet Society.

[51] D. Stebila and M. Mosca. 2017. Post-quantum Key Exchange for the Internet and the Open Quantum Safe Project. In *SAC 2016*, R. Avanzi and H. Heys (Eds.). Springer, 14–37.

[52] D. Steblia, S. Fluhrer, and S. Gueron. 2021. *Hybrid key exchange in TLS 1.3.* Internet-Draft draft-ietf-tls-hybrid-design-02. IETF. https://datatracker.ietf.org/doc/html/draft-ietf-tls-hybrid-design-02 Work in progress.

[53] H.-S. Zhong et al. 2020. Quantum computational advantage using photons. *Science* 370, 6523 (2020), 1460–1463.

A APPENDIX

Table 5: Certificate sizes of evaluated scheme combinations (reported in kB; rounded to three significant figures).

	Algorithm Combination	*Certificate Size* (kB)			*Chain Size (excl. root;* kB*)*	Δ (%)
		Root CA	*ICA*	*EE*		
Control	EDS-EDH	0.775	0.803	0.764	1.57	—
	Dil-Kyb	5.59	5.62	5.58	11.2	+615
	Fal-Kyb	2.71	2.74	2.69	5.43	+246
	XMS-Kyb	4.04	4.07	4.03	8.10	+417
	SPf-Kyb	23.3	23.3	23.3	46.6	+2870
	SPs-Kyb	11.1	11.1	11.1	22.2	+1320
Mixed Certificate Chain	XMS+EDS-Kyb	4.04	4.09	0.760	4.85	+209
	XMS+Dil-Kyb	4.04	5.72	5.58	11.3	+621
	XMS+Fal-Kyb	4.04	5.17	2.68	7.87	+402
	SPf+EDS-Kyb	23.3	23.4	0.764	24.1	+1440
	SPf+Dil-Kyb	23.3	25.0	5.58	30.6	+1850
	SPf+Fal-Kyb	23.3	24.5	2.70	27.2	+1630
	SPs+EDS-Kyb	11.1	11.2	0.760	11.9	+662
	SPs+Dil-Kyb	11.1	12.8	5.58	18.4	+1070
	SPs+Fal-Kyb	11.1	12.3	2.70	15.0	+855

Table 6: Peak memory usage of client and server programs for the evaluated scheme combinations (reported in kB; rounded to three significant figures).

	Algorithm Combination	*Client: Embedded*				*Client: Notebook*				*Server*			
		heap	stack	total	Δ (%)	heap	stack	total	Δ (%)	heap	stack	total	Δ (%)
Control	EDS-EDH	107	62.8	170	—	111	70.1	181	—	110	69.0	179	—
	Dil-Kyb	128	111	240	+41.2	137	111	248	+37.4	128	111	238	+33.3
	Fal-Kyb	119	102	221	+30.2	125	103	227	+25.9	119	103	221	+24.0
	XMS-Kyb	122	63.5	186	+9.49	124	64.1	188	+3.88	120	64.0	184	+2.86
	SPf-Kyb	167	61.3	228	+34.4	168	61.8	230	+27.2	167	61.4	228	+27.8
	SPs-Kyb	129	61.9	191	+12.4	130	61.9	192	+6.26	129	61.6	190	+6.61
Mixed Certificate Chain	XMS+EDS-Kyb	115	63.3	178	+5.19	119	70.2	189	+4.71	117	69.9	186	+3.86
	XMS+Dil-Kyb	129	111	240	+41.4	134	111	245	+35.5	126	111	237	+32.7
	XMS+Fal-Kyb	119	102	221	+30.2	125	102	227	+25.9	120	103	222	+24.5
	SPf+EDS-Kyb	131	68.3	199	+17.2	132	70.0	202	+11.7	131	69.2	200	+11.9
	SPf+Dil-Kyb	142	111	253	+49.0	147	111	258	+42.6	141	111	252	+40.8
	SPf+Fal-Kyb	135	102	236	+39.3	138	103	241	+33.3	134	102	236	+32.3
	SPs+EDS-Kyb	119	63.3	182	+7.43	123	70.2	193	+6.87	122	69.2	191	+6.89
	SPs+Dil-Kyb	133	111	244	+43.7	138	110	249	+37.7	132	111	242	+35.6
	SPs+Fal-Kyb	126	102	227	+34.0	129	102	231	+28.1	125	103	228	+27.4