IACR Communications in Cryptology
ISSN 3006-5496, Vol. 1, No. 2, 41 pages.
https://doi.org/10.62056/ahee0iuc

A Comprehensive Survey on Post-Quantum TLS

Nouri Alnahawi[2], Johannes Müller[1,3,4], Jan Oupický[1] and Alexander Wiesmaier[2]

[1] University of Luxembourg, Esch-sur-Alzette, Luxembourg
[2] Darmstadt University of Applied Sciences, Darmstadt, Germany
[3] LORIA, Nancy, France
[4] CNRS, Paris, France

Abstract. Transport Layer Security (TLS) is the backbone security protocol of the Internet. As this fundamental protocol is at risk from future quantum attackers, many proposals have been made to protect TLS against this threat by implementing post-quantum cryptography (PQC). The widespread interest in post-quantum TLS has given rise to a large number of solutions over the last decade. These proposals differ in many aspects, including the security properties they seek to protect, the efficiency and trustworthiness of their post-quantum building blocks, and the application scenarios they consider, to name a few.
Based on an extensive literature review, we classify existing solutions according to their general approaches, analyze their individual contributions, and present the results of our extensive performance experiments. Based on these insights, we identify the most reasonable candidates for post-quantum TLS, which research problems in this area have already been solved, and which are still open. Overall, our work provides a well-founded reference point for researching post-quantum TLS and preparing TLS in practice for the quantum age.

Keywords: TLS · post-quantum cryptography · post-quantum TLS

1 Introduction

Transport Layer Security (TLS) is the backbone security protocol of the Internet. This fundamental protocol protects the confidentiality, authenticity, and integrity of individual communication channels in potentially malicious network environments. TLS is one of the most widely used tool for securing application protocols, e.g., for browsing, email, instant messaging or voice-over-IP.

Like any other protocol that uses public key cryptography, the security of TLS is threatened by future quantum attackers. This is because quantum computers can efficiently break the underlying hardness assumptions of the public key cryptography on which the TLS handshake protocol is currently based, namely the integer factorization and discrete logarithm problems. Thus, future adversaries with such powerful machines can undermine the security of TLS by forging signatures to falsely authenticate themselves to users, and by deriving encryption keys to read secret messages, even from past TLS sessions.

To address this fundamental threat, and thus prevent a potential security and privacy disaster, numerous proposals have been made to replace the current quantum-insecure public-key cryptography in TLS with so-called *post-quantum cryptography (PQC)*. These are building blocks whose security is based on certain computational problems that are believed to be practically unsolvable even by scalable quantum computers. Many of the

E-mail: nouri.alnahawi@h-da.de (Nouri Alnahawi), johannes.mueller@loria.fr (Johannes Müller), jan.oupicky@uni.lu (Jan Oupický), alexander.wiesmaier@h-da.de (Alexander Wiesmaier)

Received: 2024-03-14 Accepted: 2024-06-03

proposals for post-quantum TLS are closely related to the NIST standardization process for PQC [NIS16], as TLS is often cited as a main motivation for developing cryptographic solutions that cannot be broken by quantum attackers.

While this level of interest is encouraging, the large number of contributions makes it difficult to understand the state of the art. Indeed, the existing proposals differ in many aspects: The security properties they seek to protect (confidentiality, authenticity or both), their underlying post-quantum hardness assumptions (lattice, code, isogeny, multivariate or hash-based), their specific application scenarios (unilateral or mutual authentication, pre-shared keys, etc.), the corresponding TLS versions (1.2 or 1.3), the different setups for their experiments and the resulting benchmarks (e.g., idealized or real networks), and their type of protection (purely post-quantum or combined with classical schemes).

This confusing state of affairs raises several questions: Which approaches actually achieve the desired properties? Which proposals are efficient? Which post-quantum assumptions are trustworthy? What application scenarios are realistic? Which research problems have been solved, and which are still open or need more attention? Answering these questions is crucial for making TLS post-quantum on a large scale in practice.

1.1 Our contributions

We provide the following contributions to address these questions.

Literature and internet research. To find relevant work, we searched for key words (variations of "post quantum", "TLS", "transport layer security", "authentication", "key exchange", "handshake") on Google Scholar, Crossref, DBLP, and the Internet Draft webpage of IETF. For each relevant work we identified, we ran a forward-backward search on Google Scholar and Crossref, i.e., we looked at which papers the work referred to and which papers referred to the work to find additional relevant information.

Taxonomy. We classify existing proposals for post-quantum TLS handshake according to the security properties they aim to protect: (1) post-quantum authentication only, (2) post-quantum key exchange only, and (3) post-quantum authenticated key exchange. For each of these three main categories, we identified different sub-categories to which we assigned individual publications/projects.

Evaluation. We provide a critical and comparative assessment of the state of affairs:

1. We summarize which proposals have proven insecure over time, which are considered secure according to the current state of research, and how their underlying hardness assumptions and thus their post-quantum security are related.

2. We examine which proposals deal with real-world scenarios and which do not.

3. We analyze the performance of all secure and practicable proposals and address the different security properties separately. We have carried out extensive experimental simulations, the benchmarks of which we present here. The special feature of our analysis is that we study all proposals in the same setting and that we examine proposals whose performance has not yet been tested.

Open and solved problems. We identify which research problems have been solved and which ones are still open or require further attention.

1.2 Overview

In Sec. 2.2, we describe how the TLS protocol works and explain how the security of TLS is affected by future quantum attackers. In Sec. 3, we give an overview of the PQC building blocks that are relevant for making the TLS handshake post-quantum. In Secs. 4, 5, and 6 we study the three different categories of post-quantum TLS that we have identified. In Sec. 7, we elaborate on how varying network conditions affect the performance of post-quantum TLS solutions. Finally, in Sec. 8, we summarize our main findings and conclude our paper. In App. A, we provide full details of our extensive performance analyses. In App. B, we elaborate on how to adapt symmetric cryptography in TLS for quantum security.

1.3 Related work

The performance of different post-quantum TLS proposals has been studied in various works, e.g., [DDV+20, SM16, PST20, SKD20b, DG22, SKD20a, KS19, Wel20, CPS19, ABG+23]. The work that comes closest to our contribution, both in terms of content and timing, is Wiggers' analysis in Section 11 of his dissertation [Wig24]. Wiggers' thesis was published around the same time (January 2024) that we published a first version of our paper on eprint (December 2023). Accordingly, the versions of the cryptographic schemes that we analyze in our paper and that Wiggers analyzes in his thesis are at the same current state. The main differences between Wiggers' work and ours are, first, that Wiggers' performance study of post-quantum TLS is embedded in a larger thesis, while our work is more standalone. Also, the focus of Wiggers' work is on his construction of a particular approach to post-quantum TLS, namely KEMTLS (Section 6.1), while our work is somewhat broader in scope and not tailored to a particular approach. We recommend any reader interested in post-quantum TLS to read Wiggers' thesis.

2 Description and quantum threat analysis of TLS

2.1 Description

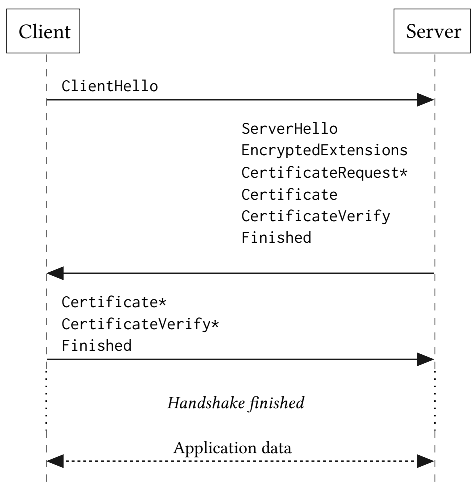


Figure 1: Typical TLS 1.3 handshake. Messages marked with * are optional depending on the authentication mode.

TLS is a protocol for establishing a secure channel between two peers, called the *client* and the *server*. The security features of TLS are confidentiality, authenticity and integrity of the messages transmitted. Our description and notation of TLS 1.3 follows the IETF specification [Res18].

Overview. TLS consists of two sub-protocols. In the *handshake protocol*, the peers authenticate themselves, negotiate cryptographic parameters and establish shared secrets. In the *record protocol*, the peers use these parameters and secrets to securely transmit the sensitive data.

Cryptographic building blocks. TLS employs the following cryptographic building blocks: *Cryptographic hash functions*, *digital signatures*, *message authentication codes (MACs)*, *key exchange protocols*, and *symmetric encryption schemes*. We assume that the reader is familiar with these basic cryptographic concepts and their security notions. Note that the only key exchange protocol supported by the TLS 1.3 standard [Res18] is (ephemeral) Diffie-Hellman.

Handshake protocol. Below we describe the most common variant of the TLS 1.3 handshake, i.e., where the peers do not have pre-shared keys (*non-PSK*), with a single roundtrip (*1-RTT*) and unilateral or mutual authentication with X.509 certificates [ITU19]. This handshake is illustrated in Fig. 1.

The client starts the handshake by sending `ClientHello`, which essentially includes the cryptographic algorithms and parameters that the client supports:

- The supported TLS cipher suites (consisting of an authenticated (symmetric) encryption scheme and a hash function) are included in the `cipher_suites` field using code points. A *code point* is an identifier of a cryptographic algorithm used in TLS.
- The `signature_algorithms` extension list the supported signature algorithms.
- The supported Diffie-Hellman (DH) groups for the key exchange protocol (EC)DHE are in the extension `supported_groups`.
- The client generates a key pair for each DH group and includes these public keys in the extension `key_share`. A *key share* is a public key that is exchanged between participants and is used to compute a shared secret.
- Optionally, the client can send an additional list of signature algorithms in the `signature_algorithms_cert` extension for certificate signatures.

The server computes the (unauthenticated) DH shared secret, which is used to derive subsequent symmetric keys for the record protocol, and responds as follows:

- `ServerHello` contains the server's choice of the parameters from `ClientHello`. The server generates its own key pair for the selected DH group and inserts the public key into the `key_share` extension.
- `EncryptedExtensions` contains extensions, which can be encrypted using the derived keys, such as `supported_groups`.
- If the server wants the client to also authenticate itself, it sends the optional `CertificateRequest` message.
- `Certificate` contains the server's certificate chain. The server's certificate must contain a public key for one of the signature schemes from `signature_algorithms`. If the client sent the extension `signature_algorithms_cert`, the certificates in the chain need to be signed with one of the signature algorithms listed in the extension.

- `CertificateVerify` contains a signature over the hash of all handshake messages up to that point. The private key used to compute the signature corresponds to the public key in the server's certificate.
- `Finished` contains a MAC over the hash of all handshake messages up to that point. The algorithm is HMAC with the hash function from the negotiated cipher suite.

If the server sent the `CertificateRequest` message, the client responds with its own `Certificate` and `CertificateVerify` messages. The final handshake message from the client is `Finished`, which includes a MAC over the entire handshake transcript.

After each side has received the `Finished` message and successfully verified all MACs and signatures, the handshake protocol is complete. The peers then start sending application data protected by the AEAD algorithm. TLS 1.3 derives the symmetric keys used in the AEAD/HMAC algorithms with a key derivation function called *key schedule*, which is determined by the negotiated hash function and takes the shared DH secret as input.

2.2 Quantum threat analysis

We explain how a *quantum attacker*, i.e., an adversary with access to a scalable quantum computer, can break the security properties of existing TLS versions.

Quantum attackers. Tab. 1 shows the general threat that future quantum attackers pose to the cryptographic building blocks that we commonly use in today's security protocols. For TLS, which uses RSA and ECDSA as signature algorithms and the Diffie-Hellman (DH) protocol for key exchange, this means that these building blocks would no longer guarantee their desired security properties. Below, we describe the implications for the confidentiality and authenticity of TLS. In App. B, we discuss whether and how to adapt the symmetric cryptographic primitives in TLS to make them post-quantum.

Confidentiality. A quantum attacker with access to the transcript of any TLS session can break the DH key exchange protocol and thus learn the peers' shared secret. With this information, the attacker can decrypt the data from the record protocol, which completely undermines the confidentiality of the channel.

It is important to note that future quantum attackers who have stored transcripts of past TLS sessions can break the confidentiality of those sessions retrospectively and obtain sensitive information that remain sensitive in the future. Such attacks are called "store now, decrypt later", and they illustrate the need for a timely transition to post-quantum key exchange in TLS.

Authenticity. A quantum attacker can forge signatures in certificates that peers use to authenticate themselves in TLS sessions. In particular, a quantum attacker can impersonate the Certificate Authority (CA) of the underlying PKI and thus break the authenticity of any TLS communication channel established with certificates from that PKI.

Unlike the confidentiality of today's TLS sessions, the authenticity of these channels will not be affected by future quantum attackers because authentication is complete at the end of the handshake phase. However, this does not mean that we can "sit back and relax", because for a variety of reasons it will take a long time to make the complete underlying PKIs post-quantum (see also Sec. 4.3).

3 Post-quantum cryptography

We give an overview of the state of the art in *post-quantum cryptography (PQC)*. The security of these cryptographic building blocks is based on hardness assumptions that even

Table 1: Quantum-Resistance of Classical Cryptography [CJL+16]

Primitive	**Algorithm**	**Quantum-Resistance**
Symmetric cipher	AES	Requires larger keys
Hash function	SHA-2	Requires larger output
	SHA-3	Requires larger output
Key exchange	(EC)DH	Not secure
Public key encryption	RSA	Not secure
	ElGamal	Not secure
Digital signature	RSA	Not secure
	(EC)DSA	Not secure

scalable quantum computers (presumably) cannot break.

As we shall see in the remainder of this paper, most post-quantum TLS proposals use post-quantum signatures and/or post-quantum key encapsulation mechanisms (KEMs) from the NIST PQC standardization process launched in 2016. Starting with 69 complete submissions [MAA+20], 15 candidates advanced to the third round. Each candidate can be placed into one of the following categories according to their underlying hardness assumption: lattice-based, code-based, hash-based, isogeny-based and multivariate. As we will see in the course of this paper, these categories offer different balances between size, speed and security. Following the third round, NIST announced in 2022 that the three signature schemes Dilithium [LDK+22], Falcon [PFH+22], and SPHINCS+ [HBD+22] as well as the KEM Kyber [SAB+22] will be standardized, and that it has selected the KEMs BIKE [ABB+22], Classic McEliece [ABC+22], HQC [AAB+22], and SIKE [JAC+20] for another round of evaluation.

As defined by NIST [NIS16], each post-quantum candidate has multiple parameter sets that provide different *security levels*, increasing from level 1 to 5. For example, an algorithm with security level 1 should be as hard to "break" as recovering an AES-128 key.

In the course of this standardization process, it has been demonstrated that several initial submissions do not achieve the desired security properties. This includes the signature schemes Picnic and qTesla [AAC+22], all multivariate signature schemes (Rainbow, GeMSS, MQDSS) [Beu22, TPD21, KZ20], and the only isogeny-based KEM SIKE [CD23].

Since, apart from the (comparatively less efficient) hash-based signature scheme SPHINCS+ [HBD+22], the only other two signature schemes to be standardized by NIST, Dilithium [LDK+22] and Falcon [PFH+22], require structured lattices, NIST has recently announced a new parallel standardization process for additional post-quantum signature schemes to diversify future PQC standards [NIS22].

4 Post-quantum authentication

We describe and compare existing proposals for post-quantum authentication in TLS. We distinguish between purely post-quantum and hybrid post-quantum approaches: *purely post-quantum* solutions (Sec. 4.1) use only post-quantum cryptography, while *hybrid post-quantum* solutions (Sec. 4.2) combine classical and post-quantum cryptography.

Authentication in TLS requires two cryptographic primitives (App. 2): digital signatures and HMACs. Since HMACs are generally considered to be quantum-secure (App. B), only the digital signatures used in TLS need to be updated to make authentication post-quantum. Signatures are used in TLS for two purposes: First, as "static" signatures of certificates in the certificate chain, and second, as "fresh" signatures of messages in the handshake protocol.

From a conceptual perspective, the use of new signature schemes, classical or post-quantum, in TLS is not complicated because the signature scheme is treated as a black box at the higher protocol level. For example, by using the reserved code points in TLS to agree and employ additional signature schemes that are not yet officially supported. The real challenge with post-quantum authentication in TLS is finding a cryptographic solution that provides a reasonable balance between post-quantum security and performance.

4.1 Purely post-quantum authentication

We describe works that replace classical signature schemes in TLS with post-quantum ones. We classify these papers based on the underlying post-quantum hardness assumptions.

In Tab. 2, we summarize the sizes of the most relevant post-quantum signature schemes and their impact on the computational performance of TLS. To this end, we simulated TLS with these signatures in a realistic and consistent environment to obtain meaningful and comparable benchmarks (see App. A for details).

Hash-based signatures. Numerous works [BSKNS20, PSS21, SM16, SKD20a, KS19, Wel20, SKD20b, DG22, PKLN22, TLF+22, SSW20] proposed the use of hash-based signatures in TLS and provided some benchmarks. Specifically, three hash-based signatures were studied: XMSS [HBG+18], Picnic[1] [ZCD+20], and SPHINCS+ [HBD+22].[2]

XMSS [HBG+18] is a *stateful* signature scheme that has been standardized in parallel with the PQC competition by NIST [CAD+20]. The main advantage of XMSS over the other signature schemes below is its performance, both in terms of computation and size. However, its statefulness limits its applicability in practice. In fact, since stateful signature schemes require private keys to be updated each time they are used, XMSS could well be used in the real world for creating certificates, as tested in [SSW20, PKLN22], but hardly for signing messages in individual TLS sessions. Note that the limitations of XMSS apply to any stateful signature scheme in this context.

Picnic [ZCD+20] is a *stateless* signature scheme whose performance in the TLS authentication phase has been tested in [PST20, Wel20]. These results show that Picnic's performance in terms of speed, key and signature sizes is similar to other hash-based signatures such as SPHINCS+ (see below). However, during the NIST competition, it was discovered that Picnic does not meet the originally claimed security level because the underlying block cipher is flawed [MAA+20, LSW+22, LIM21], and therefore NIST discarded Picnic.

SPHINCS+ [HBD+22] is the only *stateless* hash-based signature scheme that is planned to be standardized by NIST [AAC+22]. While its key sizes are small, its signature size is relatively large which limits its applicability in TLS. Specifically, while the public key size is only 32 bytes, the signature size of SPHINCS+ is 7856 or 17088 bytes at security level 1 [AAC+22]. Additionally, the operations of SPHINCS+ are orders of magnitudes slower than, e.g., those of RSA or Dilithium [SKD20a]. As a result, the TLS handshake with SPHINCS+ is significantly slower than with other signature schemes: [SKD20b, SKD20a] report that the handshake with SPHINCS+ takes $3-4\times$ the time of the handshake with classical (RSA/ECDSA) or post-quantum lattice-based (Dilithium/Falcon) signatures, and according to [DG22] the handshake with SPHINCS+ takes $2.5-15\times$ the time of the handshake with Dilithium. SPHINCS+ was evaluated on embedded devices in [BSKNS20, TLF+22]; [BSKNS20] reports that handshake with Kyber and SPHINCS+ takes $12-20\times$ the time of the classical handshake with ECDHE and ECDSA; [TLF+22] reports that the handshake with SPHINCS+ takes $714\times$ the time of the handshake with Dilithium. In our measurements, the TLS 1.3 handshake with SPHINCS+ at security level 1 took $1.58\times$ the time of the classical handshake with Ed25519. We suspect that the wide range of reported

[1] Picnic follows the MPC-in-the-head paradigm to implement digital signatures with symmetric cryptographic primitives. We have categorized Picnic as hash-based because that is its inner core.

[2] LMS [MCF19] is another hash-based signature standardized by NIST, but we do not include it in our analysis because its use has not yet been studied for post-quantum TLS.

SPHINCS+ performance is due to a combination of different setups and the fact that it has 12 parameter sets (with varying performance) at *each* security level (1, 3 and 5).

Lattice-based signatures. Various works [DDV+20, SM16, PST20, SKD20b, DG22, SKD20a, KS19, Wel20, MS22] studied the use of lattice-based signature schemes in TLS and tested their performance. Specifically, three signatures were studied: Dilithium [LDK+22], Falcon [PFH+22] and qTesla [BAA+19].

Table 2: Comparison of purely post-quantum signatures

Algorithm (sec. level)	**Public key** [bytes]	**Signature** [bytes]	**TLS slowdown coefficient** Our data	External sources
Falcon (L1)	897	666	0.98	0.75 − 1.04
Dilithium (L2)	1 312	2 420	1.02	0.95 − 1.03
SPHINCS+ (L1)	32	7 856	1.58	3 − 13.8*

Data sizes taken from [AAC+22] (for SPHINCS+ we take the small parameter choice, as defined in [HBD+22]). External sources: [SKD20b, DG22, KS19] (Falcon, Dilithium and SPHINCS+), [SKD20a] (SPHINCS+). "TLS slowdown coefficient" is defined as $\frac{t_{pq}}{t_c}$, where t_{pq} is the TLS handshake time with the post-quantum scheme and t_c is the time with a comparable classical scheme (RSA/ECDSA). We provide ranges for the external sources since their setups are not completely consistent. *We attribute the wide range to SPHINCS+ having multiple security level 1 parameter sets which differ in performance. Setup for our data (see Appendix A for more details): comparison with Ed25519, certificate chain length 2, and key exchange x25519; RTT 7ms, bandwidth 1.08Gb/s; SPHINCS+ parameter set: SHA2, simple, small.

Both Dilithium [LDK+22] and Falcon [PFH+22] are going to be standardized by NIST [CAD+20]. The security of Dilithium is based on the Module-LWE and Module-SIS problems and that of Falcon on the NTRU problem. The computational efficiency of these schemes is in the same order of magnitude as classical algorithms such as ECDSA, but their sizes are much larger. In fact, Falcon's keys and signatures are about 16× the sizes of ECDSA and Dilithium's sizes are about 39× the sizes of ECDSA (e.g., Tab. 1 in [SKD20b]). At the computational level, Falcon signature verification time is 0.3 − 0.5× Dilithium's time, but its signing time is about 10× Dilithium's time [PKLN22, DG22]. On the other hand, Falcon requires double-precision floating-point arithmetic, which is not always supported; in such cases these operations must be emulated which slows down the signing time by a factor of 10 according to [Por19]. Floating-point arithmetic also introduces side-channel vulnerabilities making Falcon insecure on some systems [Wes22, HW23]. As [KS19, PKLN22] have shown, Dilithium and Falcon offer similar performance in TLS, except in cases where Falcon's slower signing time creates a bottleneck, e.g., when a server (doing the signing) has to serve multiple clients. Our measurements show that Falcon's advantage over Dilithium grows with increasing security level: a TLS 1.3 handshake with Dilithium takes 1.04× the time of the handshake with Falcon at security levels 1 and 2, while the handshake with Dilithium takes 1.1× the time of the handshake with Falcon at security level 5.

The security of qTesla [BAA+19] is based on Ring-LWE. qTesla was submitted to the NIST competition but did not advance from the second to the third round because its security/efficiency balance was not convincing [MAA+20] and its closely related competitor Dilithium is based on a weaker hardness assumption[3] (namely Module-LWE) and offers better performance. In fact, [PST20, Wel20] have shown that qTesla typically performs worse than Dilithium in TLS handshake scenarios.

[3] If the security of a cryptographic primitive/security protocol X reduces to a weaker hardness assumption than another one Y, then X is at least as secure as Y.

Multivariate-based signatures. The use of multivariate signature schemes in the TLS handshake, such as GeMSS [CFM+20], Rainbow [DCP+20] and MQDSS [SCH+19], has been studied in some works [SKD20b, DG22, SSW20]. These schemes offer comparable or even better performance than hash-based or lattice-based ones, but all of them have been broken subsequently [Beu22, TPD21, KZ20]. As it is an open question if and how they can be fixed, we do not provide details on how the TLS handshake works with these schemes in this paper, but refer to those works that studied their deployment.

4.2 Hybrid post-quantum authentication

Hybrid post-quantum authentication uses a classical and a post-quantum signature scheme in combination, thus combining classical and post-quantum security.

Since the current TLS standard does not support the use of multiple signatures in parallel, the Open Quantum Safe (OQS) project [SM16, CPS19] extended the OpenSSL library as follows: classical and post-quantum keys and signatures are concatenated and regarded as regular keys and signatures, respectively, of a single signature scheme. This hybrid signature scheme is then used in TLS as usual. The approach is implemented for TLS 1.3 and currently supports the combination of classical algorithms RSA/ECDSA with Dilithium/Falcon/SPHINCS+.

Two works [Wel20, MS22] have tested the OQS implementation in several configurations. From their results, we can conclude that hybrid post-quantum authentication takes about $1.2 - 2\times$ the time of purely post-quantum authentication. The specific factor depends, of course, on the performance ratio between the classical and post-quantum signatures used. For example, according to [Wel20], a purely post-quantum TLS handshake using Dilithium (purely post-quantum authentication) and Kyber (purely post-quantum key exchange, Sec. 5.1) takes 18 ms, while a hybrid TLS handshake using RSA + Dilithium (hybrid post-quantum authentication) and ECDHE + Kyber (hybrid post-quantum key exchange, Sec. 5.2) takes 30 ms. Another result, using FrodoKEM instead of Kyber and Picnic instead of Dilithium, shows that the purely post-quantum handshake takes 64 ms and the hybrid handshake takes 75 ms. The main reason why the factor is about 1.7 in the first example but only 1.2 in the second is that the post-quantum primitives in the first example are more efficient than those in the second. We conducted an experiment to independently study the performance of solutions for hybrid post-quantum authentication. Our benchmarks (Tables 4 and 5 in App. A) confirm prior results. In particular, for hybrids with security level > 2, classical signatures become a bottleneck because the performance gap between Dilithium and Falcon at different security levels is not as significant as for ECDSA with P-256 and P-521.

4.3 Discussion

Observation 1. *Both purely and hybrid post-quantum authentication can be implemented efficiently.*

The benchmarks of our simulations confirm and refine the results of previous analyses (see Tab. 2). In particular, Dilithium and Falcon are the most reasonable proposals so far for quantum-proofing TLS authentication, both for use in single handshake sessions and for certificates, while XMSS is a viable alternative for certificates only. All three of these schemes will be or have been standardized by NIST. Both Dilithium and Falcon can be combined with classical signature schemes such as RSA or ECDSA for hybrid post-quantum authentication that is almost as efficient as purely post-quantum authentication.

We also find that the hash-based signature scheme SPHINCS+ is an interesting alternative, but depending on its mode it can significantly slow down the TLS handshake.

We note that the main bottleneck of post-quantum authentication in TLS is the larger key/signature size, which may limit its applicability especially in low-bandwidth networks or for some IoT applications.

We finally observe that both the purely post-quantum and hybrid post-quantum authentication approaches are crypto-agile, as they are not tailored to specific signature schemes. The purely and the hybrid (if realized as described above) post-quantum approaches can be implemented in TLS without any conceptual changes.

Observation 2. *Hybrid post-quantum solutions for authentication in TLS have not yet received the attention (in academia) that their practical relevance deserves.*

In the following, we elaborate on this observation. To begin with, we recall from above that hybrid post-quantum authentication has so far been studied in only two papers, while a dozen papers have analyzed purely post-quantum solutions. Furthermore, only standard TLS with its single signatures has been formally verified (e.g., [CHSv16]) or cryptographically analyzed (e.g., [DFGS15]), but not TLS with two (arbitrary) signatures.

We think that this state of affairs is unsatisfactory, since the hybrid approach can offer significant advantages in real world scenarios. First, from a security point of view, the hybrid approach is more robust, since even if one of the hardness assumptions, i.e., the classical or the post-quantum one, does not hold, the other can still serve as a backup. Second, hybrid solutions are also advantageous in the transition to the post-quantum era. For example, if a Trust Service Provider (TSP) [ETS21] wants to prepare to protect its authentication services against quantum attackers, it needs to find a post-quantum TLS solution that meets the relevant regulatory requirements. Since major standards such as eIDAS in Europe currently mandate the use of classical cryptography, the TSP must adopt a hybrid approach. For these reasons, post-quantum transition projects, such as HAPKIDO [pro23a] in the Netherlands, often follow the hybrid post-quantum approach, and some authorities, such as the German Federal Office for Information Security (BSI) [BSI21], require using hybrids during the migration to post-quantum cryptography.

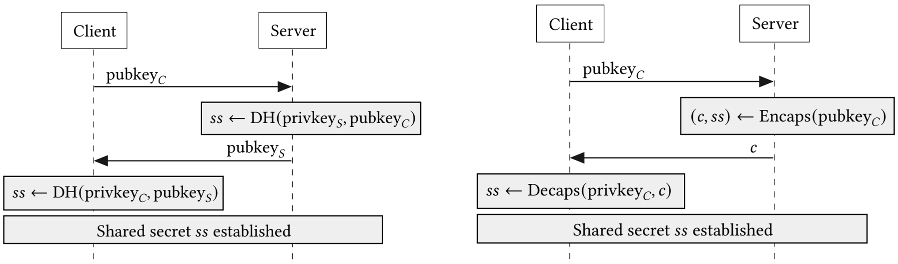
To bridge this gap between theory and practice, we recommend that the research community further explores hybrid solutions in terms of engineering and formal security.

Observation 3. *Although future quantum attackers cannot break the authentication of past TLS sessions, migration to post-quantum PKIs must begin soon.*

While future quantum attackers who gain access to the transcripts of today's TLS sessions can indeed break the secrecy of those sessions (Sec. 5), these attackers do not pose a threat to the authentication of today's TLS sessions for a simple reason. Authentication is complete at the end of the TLS handshake, and even if the server reveals its signature key afterwards, the client can still be sure that it is communicating with the correct server.

However, this does not mean that we can sit back and relax, because for a variety of reasons it will take a long time to fully secure the underlying PKIs against quantum attackers. A major constraint in this transition is the long lifetime of CA root certificates, typically 10 to 20 years, which should be secure for that time. Although in principle root certificates can be revoked at any time, in practice such a process would be cumbersome and error-prone, as it would require updating the entire PKI and all users' devices. Since current research (e.g., [MP22]) estimates that there is a high risk that quantum computers will be able to forge classically signed certificates in the next 10 to 20 years, work should start now on building PKIs that enable post-quantum authentication (in TLS).

We note that there are essentially two approaches to making existing PKIs post-quantum: creating a standalone, purely post-quantum PKI and using the classical and the post-quantum PKIs in parallel, or integrating post-quantum signatures into classical certificates. As this specific topic is beyond the scope of our paper, we refer the reader to [BHMS17, CPS19, KPDG18] for details.



(a) DH key exchange (b) KEM key exchange

Figure 2: Key exchange diagrams

5 Post-quantum key exchange

We describe and compare existing proposals for post-quantum key exchange in TLS. Again, as with post-quantum authentication (Sec. 4), we distinguish between purely post-quantum and hybrid post-quantum approaches. *Purely post-quantum* solutions (Sec. 5.1) use only post-quantum cryptography, while *hybrid post-quantum* ones (Sec. 5.2) combine classical and post-quantum cryptography.

We see that there is only one general purely post-quantum key exchange approach, while there are several approaches for hybrid post-quantum key exchange.

5.1 Purely post-quantum key exchange

All works in the literature that studied replacing the classical DH key exchange with post-quantum alternatives are based on *KEMs*. This approach was studied for TLS 1.2 in [BCNS15, BCD+16, SM16, CPS19, DDV+20, BSKNS20, PSS21] and for TLS 1.3 in [SM16, CPS19, SKD20a, Wel20, WZFG17, PKLN22, SSW20, DG22, BBCT22, TLF+22, MS22].

The idea of using KEMs for ephemeral key exchange in TLS works as follows (Fig. 2b):

1. The client creates an ephemeral KEM key pair and sends the public key to the server.

2. The server runs the encapsulation mechanism with the public key of the client as the input and obtains the ciphertext and the shared secret. The server sends the ciphertext to the client.

3. The client decapsulates the ciphertext with its private key and gets the shared secret.

Comparing this process with the original DH flow (Fig. 2a) shows that it is possible to use KEMs in TLS instead of DH key exchange without changing the overall protocol flow.[4] Thus, the main challenge for purely post-quantum key exchange is to find a cryptographic solution with a reasonable balance between quantum-security and performance.

In Tab. 3, we summarize the sizes of the most relevant post-quantum KEMs and their impact on the computational performance of TLS. As with the purely post-quantum authentication, we simulated TLS with these KEMs in a realistic and consistent environment to obtain meaningful and comparable benchmarks (see App. A for details).

Analogous to Sec. 4.1, we classify existing work on purely post-quantum key exchange based on the underlying hardness assumptions of the key exchange schemes.

[4] This approach is not entirely new, since RSA key exchange, which can be considered a KEM, could be used in TLS 1.2 for (static) key exchange.

Table 3: Comparison of purely post-quantum KEMs

Algorithm (sec. level)	**Public key** [bytes]	**Ciphertext** [bytes]	**TLS slowdown coefficient** Our data	External sources
Saber (L1)	672	736	1.07	0.89
Kyber (L1)	800	768	1.06	0.86 − 1.02
NTRU (L1)	930	930	1.13	1.57 − 1.64*
NTRU Prime (L2)	1 158	1 039	1.53	0.95
BIKE (L1)	1 540	1 572	2.55	N/A
HQC (L1)	2 249	4 481	1.23	N/A
FrodoKEM (L1)	9 616	9 720	2.53	1.14 − 1.29*

Data sizes taken from [AAC+22] (for NTRU Prime we take the streamlined parameter choice at security level 2, as defined in [BBC+20], because this parameter was used in the external source [BBCT22]). External sources: [DG22] (Saber, Kyber), [DM23b] (Kyber), [BCD+16] (NTRU, FrodoKEM), [BBCT22] (NTRU Prime). "TLS slowdown coefficient" is defined as $\frac{t_{pq}}{t_c}$, where t_{pq} is the TLS handshake time with the post-quantum scheme and t_c is the time with ECDHE. We provide ranges for the external sources since their setups are not completely consistent. "N/A" is used where we did not find external sources from which we could compute the TLS slowdown coefficient. * indicates that tested KEMs were only IND-CPA secure. Setup for our data (see Appendix A for more details): comparison with x25519, certificate chain length 2, and signature ECDSA (P-256); RTT 7ms, bandwidth 1.08Gb/s; FrodoKEM: AES, NTRU Prime: Streamlined.

Lattice-based KEMs. We identified three different groups of lattice-based KEMs that we characterize by their underlying hardness assumptions: variants of Learning-with-Errors (LWE) [BCD+16, BCNS15, PAA+19, SAB+22], Learning-with-Rounding (LWR) [DKR+20], and variants of NTRU [BBC+20, CDH+20].

Among the LWE KEMs, we distinguish between the following three categories, depending on which variant of LWE their security is based on. The security of Frodo [BCD+16] reduces to the "conservative" plain LWE assumption, that of Kyber [SAB+22] to the Module-LWE (MLWE) assumption, and that of NewHope [PAA+19] (and its predecessor BCNS [BCNS15]) to the Ring LWE (RLWE) assumption. All of these KEMs were submitted to the NIST standardization process.[5] NIST selected Kyber as a winner, while FrodoKEM and NewHope did not advance to subsequent rounds because Kyber's performance is significantly better than FrodoKEM's (see below), and because Kyber offers a similar performance to NewHope but requires a weaker hardness assumption for this (see [AAC+22, MAA+20] for details about NIST's selection).

The TLS performance of these LWE KEMs was extensively studied in the literature (see, e.g., [PSS21, BSKNS20, SKD20a, Wel20, PKLN22, DG22, SSW20, TLF+22, MS22]) and their characteristics can be summarized as follows. Overall, the performance of TLS with Kyber is similar to that of TLS with classical key exchange protocols such as ECDHE, and in some configurations even better. According to [SKD20a], the TLS 1.3 handshake with Kyber and Dilithium, at security level 1, takes roughly the same time as a classical handshake with ECDHE and RSA while at security level 3 it takes 2× the time (compared to the same classical scheme). Another work [DG22] finds that Kyber, at NIST security level 1, has almost identical performance to ECDHE with Curve25519 in TLS 1.3, and is even faster at security levels 3 (ECDHE with Curve448) and 5 (ECDHE with curve P-521) by a factor of 2 and 7 respectively. NewHope's performance is similar to that of Kyber [SKD20a] but it requires a stronger hardness assumption (RLWE vs. MLWE) and therefore Kyber can be considered superior to NewHope. The TLS handshake with

[5]The submitted versions of Frodo, called FrodoKEM [NAB+20], and of NewHope [PAA+19] differ from their original specifications [ADPS16, BCD+16] in that they provide not only IND-CPA but even IND-CCA security, as required by NIST.

FrodoKEM takes 2.7× the time a handshake with Kyber or NewHope takes according to [Wel20] and its public key and ciphertext sizes are about 10× the sizes of Kyber or NewHope [AAC+22]. The original FrodoKEM paper [BCD+16] reports that the TLS 1.2 handshake with FrodoKEM takes $1.14 - 1.29\times$ the time of a classical handshake with ECDHE but we note that this paper tested only the IND-CPA version. We measured the latest FrodoKEM IND-CCA version and we observed that the TLS 1.3 handshake with FrodoKEM at security level 1 takes 2.53× the time of the classical handshake with ECDHE (Curve25519). While these numbers show that Kyber's performance is significantly better than FrodoKEM's, they demonstrate that the more conservative FrodoKEM can still be a practical choice for use in TLS. Indeed, FrodoKEM is still recommended by some authorities (e.g., German Federal Office for Information Security) and is currently being standardized by ISO [Fro23].

The security of Saber [DKR+20] is based on the Module Learning-with-Rounding (MLWR) assumption. The use of Saber in TLS 1.3 was studied in [Wel20, DG22, MS22], all of which report that the key exchange phase in TLS with Saber is at least as efficient as with Kyber, both in terms of time and size. We also measured Saber's performance with the same outcome (App. A). Saber was also submitted to the NIST competition but, unlike Kyber, Saber did not advance to the 4th round of the NIST competition because its hardness assumption Module Learning-with-*Rounding* had not yet been researched as much as Kyber's Module Learning-with-*Errors* assumption [AAC+22].

There are two KEMs whose security reduces to the NTRU problem. The first KEM is based on a modification of NTRU, called NTRU Prime [BBC+20]. Its hardness assumption is intended to be more conservative than the original NTRU problem and than the underlying hardness assumptions of Kyber and Saber, respectively (see above), with only slightly larger key shares. In [BBCT22], the NTRU Prime KEM was integrated into TLS 1.3, essentially as described above, but with some performance optimizations which ensured that the resulting TLS 1.3 handshake with NTRU Prime takes 0.95× the time of the classical handshake with x25519. Our measurements show that the TLS 1.3 handshake with NTRU Prime's LPR variant at security level 1 takes 1.11× the time of the handshake with x25519 while the Streamlined variant at security level 1 takes 1.45× the time.

The security of the second KEM reduces to the classic NTRU problem and is therefore commonly referred to as NTRU [CDH+20]. The key shares of NTRU and NTRU Prime (see above) have essentially the same size. According to [SKD20a], the TLS 1.3 handshake with NTRU takes similar time as with Kyber (at security level 3) and takes about 2× the time of a classical ECDHE handshake; however, this result is inconsistent with the claim in the same paper that NTRU's algorithms take up to 100× the time of Kyber algorithms while having similar sizes. The performance of NTRU and Kyber in TLS has also been studied and compared in [Wel20] which reports that the handshake with NTRU takes about 3.25× time of the handshake with Kyber while being comparable to FrodoKEM. According to [MS22] the TLS 1.3 handshake with NTRU takes $1.05 - 1.07\times$ the time of a handshake with Kyber or Saber at multiple security levels which corresponds to our measurements presented in Appendix A.

Code-based KEMs. HQC [AAB+22] and BIKE [ABB+22] are the only code-based KEMs tested in TLS [Wel20, TLF+22, DG22].[6] HQC and BIKE were both submitted to the NIST competition and reached the 4th round [AAC+22]. BIKE's public key and ciphertext sizes are about 0.5× the sizes of HQC but its computations (key generation, encapsulation, decapsulation) take approximately 4.3× the time of HQC's computations [DM23a].[7]

[6] A reader may wonder if Classic McEliece [ABC+22] has been also integrated into TLS. We have not found such a work, and we suspect that this is because TLS does not support public keys larger than $2^{16} - 1$ bytes [Res18], but the smallest public key size for Classic McEliece is 4× this limit.

[7] Computed as sum of cycles for each operation at security level 1, x86 architecture, code type - performance mode, 2023-06-24.

According to [TLF+22], the TLS 1.3 handshake with BIKE takes 4.3× the time of the handshake with Kyber, and the handshake with HQC takes about 1.8× the time of the handshake with Kyber, whose performance is similar to that of ECDHE. Our measurements show that the TLS 1.3 handshake with HQC at sec. level 1 takes 1.45× the time of the classical handshake with x25519 while the handshake with BIKE at sec. level 1 takes 2.55× the time of the classical handshake.

Isogeny-based KEMs. SIKE [JAC+20] is the only isogeny-based KEM that was tested in TLS, in fact in [Wel20, DG22, SSW20, TLF+22]. The advantage of SIKE over code-based or lattice-based schemes (see above) is its key and ciphertext sizes which are at most 0.5× the sizes of lattice-based KEMs, such as Kyber [SAB+22] or Saber [DKR+20]. On the downside, a TLS 1.3 handshake with SIKE takes around 15× the time of the handshake with Kyber or Saber according to [Wel20], and even 31× according to [DG22]. The main problem with SIKE, however, is that it has been shown to be insecure [CD23].

5.2 Hybrid post-quantum key exchange

We identified three different approaches to hybrid key exchange in TLS. The first two run one classical and one post-quantum key exchange algorithm in parallel, and they differ in how they integrate the additional post-quantum key share into the overall protocol flow (Secs. 5.2.1 and 5.2.2). The third approach differs from the other two in that it uses one classical and $n \geq 1$ post-quantum algorithms in parallel (Sec. 5.2.3).

5.2.1 Concatenation

The main idea here is to concatenate the classical and post-quantum key material (public keys/ciphertexts) and treat them as single elements in TLS. This is conceptually similar to the hybrid authentication as described in Sec. 4.2.

More specifically, in the negotiation phase, one classical and one post-quantum key exchange protocol are negotiated. Then, in the key exchange phase, the classical and post-quantum key exchange algorithms are run in parallel on a concatenated input, the classical and post-quantum parts of which are first decoupled and then processed individually by the respective algorithms. The outputs of the two algorithms are concatenated to form the shared secret, which is then processed as in the usual TLS protocol. We note, however, that one of the key shares must be of known constant length to be able to decouple them; this requirement can be avoided, for example, by defining an encoding of the key shares.

The concatenation approach was first proposed for TLS 1.2 in [BCNS15] and later in [BCD+16]. The authors of [BCNS15] instantiated this approach with ECDHE and their Ring-LWE-based key exchange algorithm BCNS (Sec. 5.1). Their evaluation results show that the TLS handshake using this hybrid key exchange takes about 1.7× the time of the handshake using ECDHE and about 1.4× the time of the handshake using the purely post-quantum key exchange with BCNS. The results of [BCD+16] show that the handshake using a hybrid key exchange (ECDHE with NewHope [ADPS16]) takes about 1.2× the time of the handshake with ECDHE and about 1.4× the time of the handshake using the purely post-quantum key exchange NewHope.

Later, the concatenation approach was also proposed for the latest version 1.3 of TLS in [SFG24, KK18]. The OpenSSL fork from the Open Quantum Safe (OQS) project implements the concatenation approach [SM16, CPS19] for hybrid key exchange in TLS 1.3. The efficiency of this implementation in various settings was analyzed in [PST20, Wel20, SSW20]; we summarize their main insights next. First, [PST20] reports that the performance of a hybrid key exchange with ECDHE and Kyber in TLS 1.3 is essentially identical to a classical key exchange with ECDHE under *ideal network conditions*. In environments with a high packet loss, the additional bandwidth requirements of post-quantum algorithms make the hybrid key exchange protocol significantly less efficient than the classical one. Another main observation made in [PST20] is that SIKE [JAC+20], with key shares 0.4× the size of Kyber, performs significantly worse than Kyber under ideal network conditions due to its much slower computations, but in environments with high packet loss, SIKE outperforms Kyber, since SIKE's slow computations get negligible compared to Kyber's higher bandwidth requirements, which cause a higher packet retransmission rate.[8]

We conducted an experiment to study the performance of solutions for hybrid post-quantum key exchange; see Table 7 and Figure 6 in App. A for our detailed results. In particular, we observed that a hybrid post-quantum handshake with a combination of x25519 and Kyber at security level 1 took 1.25× the time of a classical handshake with x25519. Interestingly, the more secure combination of x25519 with Kyber at security level 3 took 1.28× the time of a classical handshake with x25519 and is therefore similarly fast; this combination is currently used in Google Chrome (see below).

Google conducted two hybrid post-quantum TLS experiments [Lan16, Lan18] that also implemented the concatenation approach. In the first experiment [Lan16], the authors tested a hybrid key exchange in TLS 1.2 with NewHope [ADPS16] and ECDHE, and they concluded that the tested hybrid post-quantum solution can be considered practical. The second experiment [Lan18] was conducted in collaboration with Cloudflare [KV19]. It tested two hybrid key exchange configurations in TLS 1.3 with NTRU [CDH+20] and ECDHE, and with SIKE [JAC+20] and ECDHE. The overall result is consistent with the results of [PST20] (see above), as the performance of the pair NTRU/ECDHE is similar to that of the classical ECDHE, while the performance of the pair SIKE/ECDHE is worse than NTRU/ECDHE under ideal network conditions, but better under poorer network conditions such as in mobile communications.

In 2022, Google announced that it had implemented this hybrid approach (using ECDHE with NTRU) in its modified TLS protocol called ALTS [Goo22]. Since August 2023, Google Chrome 116+ [Goo23b] supports post-quantum hybrid key exchange by a combination of x25519 with Kyber-768; see [WS23] for implementation details.

A recent development in this context are *unified KEM combiners*, which derive a hybrid secret from composite secrets (see, e.g., [GHP18, BBF+19]). For example, X-Wing [BCD+24] is a unified KEM combiner for x25519 and Kyber-768. The deployment of these tools to simplify the standardization and migration process has recently been discussed on the IETF mailing list [Wes24].

5.2.2 Separation

An alternative solution to concatenating key shares by using existing TLS data structures (Sec. 5.2.1) is to modify messages or to create extensions in order to transport additional post-quantum key shares in the hybrid key exchange. In this way, the classical and post-quantum key shares are separated more clearly in the TLS data structures.

The authors of [CC21] apply this idea to TLS 1.2 by extending `ClientHello` with a list of supported post-quantum KEMs, ordered by preference. In order to transport both the classical and the post-quantum keys, they modify `ServerKeyExchange` and `ClientKeyExchange` so that the post-quantum key share is embedded in a newly defined field, while the classical key share is embedded as in the TLS 1.2 standard. As in the previous approach (Sec. 5.2.1), the final shared secret is the concatenation of the resulting classical and post-quantum secrets.

The same approach has been proposed for TLS 1.3 in [SS17] but with some technical differences. First, the authors define new *group* code points for the post-quantum key exchange algorithms, so that the hybrid pair of key exchange algorithms can be negotiated as in standard TLS 1.3. Second, the authors define a new extension `additional_key_share`

[8] However, recall from Sec. 5.1 that SIKE has been shown to be insecure after these experiments.

of ClientHello and ServerHello. While the classic key shares are transported as in the TLS 1.3 standard, the post-quantum ones are sent using the new extension. Finally, the authors modified the key schedule (i.e., the key derivation algorithm) in TLS 1.3 to take as input the classical and the post-quantum shares.

AWS conducted an experiment using this approach in TLS 1.2 [Hop19]. Specifically, they tested ECDHE combined with BIKE [ABB+22] and SIKE [JAC+20]. Their results show that BIKE and SIKE provide different tradeoffs: BIKE consumes more bandwidth while SIKE has slower computations.

5.2.3 Multiple hybrid

The idea of this approach, which we call the *multiple hybrid*, is to use more than two key exchange algorithms in parallel to further reduce the attack surface.

In [SWZ16], this approach is applied to TLS 1.2. First, the authors extend ClientHello and ServerHello to negotiate the combination of multiple (post-quantum) key exchange algorithms and to transport all public keys. Second, they modify ServerKeyExchange and ClientKeyExchange to transport the additional ciphertexts in new dedicated fields. The classical key exchange phase remains unchanged. Finally, the shared secret is the concatenation of the classical secret and the additional (post-quantum) secrets.

A similar approach was proposed in [WZFG17] for TLS 1.3. The proposal modifies the standard key_share extension to accommodate hybrid key sharing, and also defines a new TLS extension hybrid_extension for ClientHello, which serves as a mapping between code points in supported_groups and the set of key exchange algorithms used in the hybrid key exchange. For example, if the client wants to use a hybrid key exchange with ECDHE, Kyber and NTRU, the client selects an unallocated code point in supported_groups and includes the description of the code point into hybrid_extension. The public keys are transported in the key_share extension, except that the KeyShareEntry corresponding to the hybrid code point contains a list of three KeyShareEntry values representing the public keys. The authors of [WZFG17] note that their modification is interoperable with standard TLS 1.3, since the modified KeyShareEntry is treated identically to a KeyShareEntry of an unsupported algorithm in TLS 1.3.

We have not found any publications that analyze the performance of this approach. While for one classical and one post-quantum algorithm, we assume that the performance is the same as in the concatenation approach, which has been tested extensively (Sec. 5.2.1), it is not obvious how this approach scales with more than one post-quantum key exchange algorithm in realistic settings.

5.3 Discussion

We summarize the state of the art and place these findings in a real-world context. Recall from Sec. 2.2 that future quantum attackers can, in particular, undermine the confidentiality of TLS by breaking the classical key exchange protocol, even of past TLS sessions ("store now, decrypt later" attacks).

Observation 4. *All existing proposals for purely and hybrid post-quantum key exchange in TLS use KEMs. Using KEMs for key exchange requires significant protocol changes, and the security of the resulting TLS protocol has not yet been formally verified.*

This observation, which follows directly from Secs. 5.1 and 5.2, shows that post-quantum key exchange is more challenging than post-quantum authentication (Sec. 4), since simply replacing the classical cryptography (e.g., ECDHE in TLS 1.3) does not work here. In fact, deploying a different cryptographic concept, namely KEMs, is the only known approach to efficient post-quantum key exchange.

From an engineering perspective, using KEMs for key exchange in TLS requires some adaptions of the handshake protocol, as opposed to simply using different signatures for post-quantum authentication.

To our knowledge, the security of the TLS protocol with (abstract) KEMs for key exchange has not yet been formally verified. Although we assume that the security of the TLS protocol is in principle preserved, we think that this research problem needs to be addressed before the modified TLS is used in real world scenarios. To illustrate the importance of a formal analysis, we recall that during the development of TLS 1.3, researchers formally verified the security of the drafts and identified subtle issues (see, e.g., [CHSv16]).

A specific open question that needs to be answered is whether the weaker notion of IND-CPA or the stronger (and commonly used) notion of IND-CCA secure KEMs is sufficient to preserve the security of the current TLS 1.3 standard. Since in some early works [BCNS15, BCD+16], the proposed KEMs are only secure against chosen plaintext attacks (IND-CPA), the authors of these papers changed the protocol flow of TLS 1.2, as otherwise the security of the resulting protocol could not be guaranteed (Sec. 5 in [BCNS15]). Essentially, in the modified version, the server signs the transcript at a later stage, so that it also contains the client's public key. Since in TLS 1.3, the server already signs the transcript in this way, it appears that IND-CPA secure KEMs could be used in TLS 1.3 without further modification to improve efficiency. The authors of KEMTLS [SSW20] prove its security under the IND-1CCA assumption, a restricted form of IND-CCA where the adversary can make only a single query to its decapsulation oracle.

Observation 5. *Both purely and hybrid post-quantum key exchange can be implemented efficiently.*

Let us first summarize the state of the art for purely post-quantum key exchange in TLS (Sec. 5.1); see Tab. 3 for the sizes of the most relevant post-quantum KEMs and their TLS performance. We find that the lattice-based KEMs FrodoKEM, Kyber and NTRU Prime, and the code-based KEMs HQC and BIKE are the most promising proposals so far for post-quantum key exchange in TLS. These KEMs are efficient, albeit to varying degrees, and based on reasonable hardness assumptions. In contrast, the security of NewHope relies on Ring-LWE, which may be significantly easier to solve than the hardness assumptions of the other lattice-based schemes, and the underlying hardness assumption of Saber is not yet sufficiently well understood. Since SIKE has been shown to be insecure, this KEM should not be used.

We have also seen that the concatenation approach for hybrid post-quantum key exchange (Sec. 5.2.1) provides a practical option, for example for the Kyber/ECDHE pair. Although, as to the best of our knowledge, the performance of the separation approach (Sec. 5.2.2) has not yet been studied in the literature, we assume that its overall efficiency is the same as that of the concatenation approach, since the cryptographic content is the same, only transferred in different places. As we already noted in Sec. 5.2.3, it is still an open research question to study the performance of using multiple post-quantum KEMs for higher robustness.

Observation 6. *While the concatenation approach to hybrid key exchange has received more attention, both the concatenation and separation approaches provide reasonable solutions with slightly different characteristics.*

Unlike the separation approach (Sec. 5.2.2), the concatenation approach (Sec. 5.2.1) is the only hybrid key exchange approach that has been implemented and tested in various TLS libraries (e.g., OpenSSL [SM16], s2n [AWS23]). Furthermore, the concatenation approach is currently an active IETF draft [SFG24], while the IETF drafts of the other two approaches have expired, namely [CC21, SS17] for the separation approach and [WZFG17, SWZ16] for the multiple hybrid approach.

The main difference between the separation and concatenation approaches is the level at which they operate. Because the concatenation approach solves the hybrid authentication problem at the cryptographic level, migration can be carried out by replacing the cryptographic components without changing the overall TLS protocol. Among other things, this allows the standard TLS key schedule to be maintained.

We are not convinced that the multiple hybrid approach (Sec. 5.2.3), which enables one to use more than two KEMs in parallel, offers any significant advantages in practice. Using one classical and one post-quantum KEM not only provides a high level of security, but it is also sufficient to address possible legal restrictions (Sec. 4.3). Thus, the limited added value of using more than two KEMs does not justify the computational overhead.

6 Post-quantum authenticated key exchange

We describe and discuss those proposals that simultaneously guarantee authenticity and secrecy of the TLS handshake, unlike the proposals in the previous two sections, which provide exactly one of these two properties.

6.1 KEMTLS

KEMTLS [SSW20] uses post-quantum KEMs to provide post-quantum authenticated key exchange (AKE) in TLS 1.3 without signing messages.[9] The main idea of KEMTLS is based on the observation that signing messages between the client and server is sufficient but not necessary for authentication. In fact, the handshake can be more efficient using a KEM instead of signatures. This is done by using a KEM to prove to the client that the server knows a secret key corresponding to its public verification key.

While KEMTLS is conceptually similar to the "TLS 1.3 candidate" OPTLS [KW16], which uses DH for AKE without signatures, KEMTLS is the first work that realizes this idea with post-quantum KEMs.

In the usual TLS 1.3 setup with unilateral authentication, the server has a static signature key pair and a certificate binding the signature public key to the server. In the corresponding KEMTLS setup, the server has a static KEM key pair and a certificate binding the static KEM public key to the server.

Description. We now describe the KEMTLS handshake protocol. Since KEMTLS is a modification of the standard TLS 1.3 protocol (Sec. 2.1), we only present the modifications.

1. Client → Server: The client sends `ClientHello` with KEM algorithms selection and corresponding ephemeral KEM public keys.

2. Server → Client: The server selects a KEM and runs encapsulation using the client's public key. The resulting KEM ciphertext is the server's key share in `ServerHello` and the shared secret is fed into the key schedule. The server's `Certificate` message contains a certificate with its static KEM public key. `CertificateVerify` is never sent. `Finished` is delayed and sent in the next message flow.

3. Client → Server: The client decapsulates the KEM ciphertext from `ServerHello` and the resulting shared secret is fed into the key schedule. The client runs encapsulation using the server's static public key. The client sends a new KEMTLS message `ClientKemCiphertext` containing the resulting KEM ciphertext. The resulting (second) shared secret is also fed into the key schedule. The client sends `Finished`.

[9] KEMTLS still uses digital signatures for certificates.

4. Server → Client: The server decapsulates the KEM ciphertext from `ClientKemCiphertext` and inputs the shared secret into the key schedule. The server sends its `Finished` message. Note that this message flow is extra compared to TLS 1.3.[10]

After the client receives the server's `Finished` message, the handshake is concluded. Note that compared to TLS 1.3, the `Finished` messages are computed using HMAC keys which are derived from the initial ephemeral key exchange and the static key exchange.

The modifications can be summarized as follows: the ephemeral key exchange uses KEMs instead of DH, the client sends `ClientKemCiphertext` instead of the server sending `CertificateVerify`, the server delays its `Finished` message and the TLS key schedule includes an extra round.

As mentioned above, the main concept of KEMTLS is to use KEMs instead of signatures for AKE, whereby KEMTLS trades the cost of creating and verifying one signature for the cost of encapsulating and decapsulating one message (plus a practically negligible key scheduling round).

Performance. The main advantage of KEMTLS is its smaller communication size, due to the fact that post-quantum KEMs have typically smaller bandwidth requirements than post-quantum signatures in those protocols presented in Sec. 4.

For example, [SSW20] reports that a KEMTLS handshake with Kyber [SAB+22] as KEM and Dilithium [LDK+22] as the signature scheme for certificates is about 0.85× the size of the TLS 1.3 handshake using post-quantum authentication with Dilithium (as in Sec. 4.1) and post-quantum key exchange with Kyber (as in Sec. 5.1). In another example, [SSW20] states that the size of the KEMTLS handshake is about 0.71× the size of the corresponding post-quantum TLS handshake when SIKE [JAC+20] is used as the KEM and Falcon [PFH+22] and XMSS [HBG+18] are used to create certificates.

Therefore, overall, the KEMTLS handshake can be faster in certain settings, as reported in [CFS+21, SSW20]. While the performance of KEMTLS is essentially the same as the separate post-quantum approach for the Kyber/Dilithium pair, the KEMTLS handshake takes 0.98× the the time of a TLS 1.3 handshake when used with the NTRU/Falcon pair. On the other hand, when KEMTLS is used with a computationally expensive KEM such as SIKE, it can take up to 1.36× the time of the corresponding separate post-quantum approach. The performance of KEMTLS was also studied on embedded devices in [GW22], which reports that "KEMTLS can reduce handshake time by up to 38%, can lower peak memory consumption and can save traffic volume compared to TLS 1.3".

Extensions. While the original KEMTLS paper [SSW20] focuses on unilateral authentication, it also describes in its appendix how to achieve mutual authentication by adding an extra round trip.

In [SSW21], a modification of KEMTLS with *pre-distributed keys* is proposed, called KEMTLS-PDK. Pre-distributed keys are similar to pre-shared keys in TLS 1.3, except that pre-shared keys are symmetric keys, while pre-distributed keys are public keys. This method significantly reduces the handshake size and can perform marginally better than purely post-quantum TLS (Sec. 4.1 and 5.1) with caching of certificates. Furthermore, it allows using post-quantum KEMs with large public keys in KEMTLS, such as Classic McEliece [ABC+22] (for authentication) combined Kyber (for ephemeral key exchange), thus providing hybrid confidentiality: even if Kyber was broken, confidentiality would be preserved as long as Classic McEliece is safe.

While the original KEMTLS paper [SSW20] only mentions the possibility of using hybrid KEMs to achieve hybrid AKE, this option was implemented in [GdNC+23], which demonstrated that the overhead of using hybrids is small (about 8%) and that hybrid KEMTLS is therefore a realistic option.

[10] The client can still send its application data together with the `Finished` message.

Security. We note that, as pointed out in [SSW20], the KEMs in KEMTLS need to be IND-CCA secure. In addition to its original pen-and-paper proof, the KEMTLS(-PDK) protocol was formally verified using Tamarin [CHSW22] and SAPIC+ [CJKK22].

Related work. A recent paper [CCSCD+23] proposes an alternative instantiation of OPTLS [KW16] using CSIDH [CLM+18] to provide post-quantum authenticated key exchange. Additionally, the paper proposes new parameters for CSIDH taking into account papers which called CSIDH's security into question, e.g. [Pei20]. Although conceptually similar to KEMTLS, the performance of this proposal is much worse than KEMTLS: in particular, the handshake of this protocol takes *at least* 117× the time of KEMTLS (using Kyber).

6.2 Mutually authenticated key exchange

While KEMTLS was primarily designed for unilateral authentication in TLS, where the server authenticates itself to the client, other works proposed post-quantum solutions for mutually authenticated key exchange in TLS, where the server and the client authenticate themselves to each other. We identified three different approaches in this group: (Specific) Ring-LWE AKE, password-authenticated key exchange (PAKE), and identity-based encryption (IBE). The security of all presented proposals reduces to the Ring-LWE problem.

6.2.1 Ring-LWE authenticated key exchange

The authors of [GLD+17] proposed a modification of the lattice-based AKE of [ZZD+15] for use in TLS 1.2. They define a ciphersuite for this primitive, but they do not specify exactly how this ciphersuite is integrated into TLS 1.2; for example, they do not define which TLS messages transport the public keys. In short, [GLD+17] focuses on the cryptographic aspects rather than on the protocol itself.

Since the underlying AKE protocol [ZZD+15] is designed for mutual authentication, employing it in TLS requires static keys for both parties, which need to either be pre-shared or part of a certificate. The distribution of these static keys not discussed in [GLD+17].

The authors of [GLD+17] compare the performance of their AKE in TLS 1.2 with two classical ciphersuites (FFDH with RSA and ECDH with RSA) and with the Ring-LWE key exchange protocol [BCNS15] (Sec. 5.1) in combination with RSA and ECDSA for authentication. The authors report that their AKE with a parameter set (supposedly) providing 80-bit security is faster than the 1024-bit FFDH with 2048-bit RSA, and their AKE with a 256-bit parameter set is slower than ECDH with a 256-bit parameter set and 2048-bit RSA. They also compared their AKE with [BCNS15] (combined with RSA/ECDSA) and observed that the TLS handshake with their AKE takes $1.12 - 1.18\times$ the time. We note, however, that these comparisons should be taken with a grain of salt, as the AKE provides mutual authentication while the others provide unilateral authentication.

6.2.2 Password-authenticated key exchange

Another paper [GDL+18] proposed to employ the PAKE protocol of [DAL+17] in TLS 1.2. Unlike standard AKE protocols, such as the one above, which uses pre-shared *keys*, PAKE protocols rely on pre-shared passwords for higher efficiency in exchange for weaker security.

The only performance data provided in [GDL+18] shows that the handshake with their PAKE takes on average about 4.9ms, when both the server and the client are on the same machine, which is an unrealistic setting. The authors also state that the transcript size of the PAKE key exchange is about $2 - 3\times$ the size of classical (EC)DHE/RSA key exchange.

6.2.3 Identity-based encryption

The authors of [BC20] took a completely different approach. They proposed to use an identity-based encryption (IBE) scheme together with a KEM for a mutually authenticated key exchange in TLS 1.3. In IBE schemes, the individual public key of each user can be publicly derived from a master public key and the user's identity. The master public key is generated by a trusted authority, which also distributes the individual private keys to the users. The main idea of [BC20] is to have the CA of the PKI create a master public key and then distribute the individual private keys to the clients and servers. The authors claim that this avoids the need for certificates. The authors of [BC20] implemented their approach in TLS 1.3 using a Ring-LWE IBE scheme in combination with NewHope [ADPS16] as the KEM (Sec. 5). They stated that their approach was about 4.5× more efficient in terms of size and time than alternative certificate-based solutions, without specifying which solutions they were referring to.

6.3 Discussion

Observation 7. *The application scenarios of the KEMTLS and of the AKE approaches are realistic, whereas those of the PAKE and the IBE approaches are not.*

The assumptions made by KEMTLS and AKE are those commonly considered for TLS: server certificates (KEMTLS only), pre-shared keys between clients and servers, or server and client certificates (KEMTLS & AKE). On the contrary, we are not convinced that the PAKE and IBE approaches make reasonable assumptions. First, the use of pre-shared *passwords*, as required by PAKE, is not common in TLS applications. Second, the IBE approach places too much trust in the CA, which not only holds a master secret key to authenticate users, as in standard TLS, but can also exploit this to read all messages.

For these reasons, we restrict our attention to the KEMTLS and AKE approaches in the following.

Observation 8. *Various post-quantum instantiations of KEMTLS and one specific instance of the AKE approach have been shown to be practically efficient.*

As mentioned in Sec. 6.1, the original KEMTLS paper presented detailed benchmarks of KEMTLS with various reasonable post-quantum instantiations, and [GdNC+23] showed that hybrid KEMTLS is also efficient. In contrast, the AKE approach has so far been instantiated only with a Ring-LWE primitive, whose performance has indeed been tested, but it remains an open question how the overall TLS handshake with that primitive performs in realistic network settings.

Observation 9. *While KEMTLS is well defined, well studied with different instantiations, and well analyzed at both cryptographic and protocol levels, the focus of the AKE approach is reduced to the cryptographic details of a single specific instance.*

This observation follows directly from our descriptions in Secs. 6.1 and 6.2.1 and implies that KEMTLS is so far the only viable and trustworthy solution for non-separate post-quantum AKE in TLS.

7 Non-ideal network conditions

From a practical point of view, it is important to understand how the different approaches to post-quantum TLS behave under non-ideal network conditions. In the previous sections, we focused on the individual local operations and created quasi-ideal network conditions in our experiments to minimize interfering factors (see App. A). In this section, we complement this perspective with insights on the behavior of solutions for post-quantum TLS under non-ideal network conditions. To this end, we summarize the results from [SKD20a, PST20, SSW20, Wig24, KV19] and draw a general conclusion.

Essentially, these studies show that the sizes of the various cryptographic components (such as the public key, the ciphertext, or the signature) are more relevant the worse the network conditions are, and the speeds of the cryptographic algorithms are more relevant the better the network conditions are. In general, this means that the balance between size and speed is critical in practice, since the exact network conditions are typically uncertain. This applies to both KEMs and digital signatures.

Let us take two examples to illustrate how size becomes a dominant factor under non-ideal network conditions. First, consider an Ethernet connection. Since the *maximum transmission unit* (MTU) of Ethernet is 1500 bytes, a `ClientHello` with a small public key (e.g. SIKE with 330 bytes) does not need to be fragmented into as many packets as with a larger public key (e.g. FrodoKEM with more than 9000 bytes). Second, consider packet loss scenarios. Fig. 1 of [PST20] shows that at low packet loss rates $< 5\%$ the TLS handshake with SIKE is much slower than with FrodoKEM, but at packet loss rates of $8 - 15\%$ SIKE becomes faster than FrodoKEM.

8 Conclusion

Based on our findings from Secs. 4, 5 and 6, we elaborate on the (research) problems that have been solved and those that are still open or need more attention.

8.1 Solved problems

The first major finding of our systematic review is a very positive one: there are a number of reasonable ways to prepare TLS for the era of quantum attackers. Since the TLS record layer can be adapted by changing the security parameters of the ciphers and hash functions (App. B), the main focus is on the TLS handshake.

Efficient purely post-quantum solutions. We have learned that there exist reasonably secure and efficient *purely* post-quantum solutions for authentication (Sec. 4.1), for key exchange (Sec. 5.1), and for AKE (Sec. 6.1) in TLS. Our extensive benchmarks (see App. A for full details) refine and confirm the results of previous performance studies. We observed all reasonable solutions for post-quantum key exchange (in the literature) use KEMs for this purpose.

Recall that there are two ways to build a purely post-quantum TLS handshake: first, by combining separate post-quantum solutions for authentication and key exchange; and second, by using a post-quantum solution that provides AKE directly. For the first approach, there exist several secure and efficient combinations (Secs. 4.3 and 5.3), while for the second approach, we found that KEMTLS is the only reasonable solution to date.

Efficient hybrid post-quantum solutions. We have seen that there also exist efficient *hybrid* post-quantum solutions for authentication (Sec. 4.2), for key exchange (Sec. 5.2), and for AKE (Sec. 6.1) in TLS. Although these proposals have some overhead, their efficiency is not much lower than that of purely post-quantum solutions.

We think that the benefits of hybrid solutions outweigh their practically limited performance overhead. First, it reduces a risk that should not be neglected by maintaining classical cryptographic components as a fallback mechanism. In fact, many post-quantum schemes and their underlying hardness assumptions are quite young, and it may well be that some of them turn out to be less secure than currently thought. For example, SIKE reached the fourth round of the NIST standardization process before it was shown to be completely insecure. Second, as we argued in Sec. 4.3, hybrid solutions using classical cryptographic building blocks may be advantageous in the transition to the quantum era because of their backward compatibility and compliance with current regulations.

However, to reduce the number of possible classical/post-quantum combinations, we recommend limiting the number of post-quantum algorithms supported in hybrid post-quantum TLS.

High-quality implementations. The development of post-quantum TLS has largely been driven by a few flagship projects, most notably the Open Quantum Safe (OQS) project, which published high quality implementations of post-quantum TLS. The OQS implementation is not only the most comprehensive, offering both post-quantum (hybrid) authentication and key exchange, but also provides easy-to-use benchmarking facilities. AWS's implementation of TLS with post-quantum key exchange [AWS23] is another notable publication.

8.2 Open problems

We identified the following three major open problems.

Formal analysis. As we already noted in Sec. 5.3, it is crucial to assure that not only the PQC building blocks are secure but also the overall TLS protocol that needs to be adopted. Stringent proofs in formal frameworks are the accepted method for ensuring this. Due to the complexity of the TLS protocol and the fact that multiple TLS sessions can run concurrently, computer-aided verification is particularly well suited for high assurance.

Except for the approaches for purely post-quantum authentication (Sec. 4.1), which do not change the TLS protocol, all other approaches require adapting the TLS protocol. However, to the best of our knowledge, KEMTLS is the only post-quantum TLS approach which has been formally analyzed, namely in symbolic frameworks with computer-aided tools [CHSW22, CJKK22] and in a cryptographic framework with "pen-and-paper proofs" [SSW20]. In particular, it is still an open research problem to formally analyze TLS with (two) KEMs for (hybrid) key exchange, as required by all post-quantum key exchange solutions (Sec. 5), and TLS with combined signatures, as required by hybrid post-quantum authentication solutions (Sec. 4.2).

Increasing diversity. As we observed above, there are reasonable solutions for post-quantum TLS, but their performance in low-bandwidth environments with significant packet loss is much worse than that of current TLS due to their larger sizes. Since SIKE was the only post-quantum proposal with a demonstrably better performance in such environments but it turned out to be insecure, it is desirable to develop alternative post-quantum solutions with low communication cost.

Adaption of standards. The most important step towards large-scale deployment of post-quantum TLS is the adaptation of the relevant standards. Specifically, we recommend that the next version of the TLS standard should provide the ability to use KEMs for key exchange to ensure post-quantum confidentiality. It should also allow more than one signature or key exchange protocol to be used in parallel to enable hybrid quantum-security. Again, to avoid subtle security issues, a formal analysis of future drafts will be crucial.

Acknowledgements

We thank the anonymous CiC reviewers for their detailed and constructive feedback.

The research work by Nouri Alnahawi has been funded by the German Federal Ministry of Education and Research and the Hessian State Ministry for Higher Education, Research and the Arts within their joint support of the National Research Center for Applied Cyber-Security ATHENE. Johannes Müller was supported by the Luxembourg National Research Fund (FNR), under the CORE Junior project FP2 (C20/IS/14698166/FP2 /Mueller). Jan Oupický was supported by the industrial partnership project between the interdisciplinary research center SnT and LuxTrust. This work benefited from funding managed by the French National Research Agency under the France 2030 programme with the reference ANR-22-PECY-0006.

References

[AAB+22] Carlos Aguilar-Melchor, Nicolas Aragon, Slim Bettaieb, Loïc Bidoux, Olivier Blazy, Jean-Christophe Deneuville, Philippe Gaborit, Edoardo Persichetti, Gilles Zémor, Jurjen Bos, Arnaud Dion, Jerome Lacan, Jean-Marc Robert, and Pascal Veron. HQC. Technical report, National Institute of Standards and Technology, 2022. available at https://csrc.nist.gov/Projects/post-quantum-cryptography/round-4-submissions.

[AAC+22] Gorjan Alagic, Daniel Apon, David Cooper, Quynh Dang, Thinh Dang, John Kelsey, Jacob Lichtinger, Carl Miller, Dustin Moody, Rene Peralta, Ray Perlner, Angela Robinson, Daniel Smith-Tone, and Yi-Kai Liu. Status Report on the Third Round of the NIST Post-Quantum Cryptography Standardization Process. Technical Report NISTIR 8413, National Institute of Standards and Technology, September 2022. doi:10.6028/NIST.IR.8413-upd1.

[ABB+22] Nicolas Aragon, Paulo Barreto, Slim Bettaieb, Loic Bidoux, Olivier Blazy, Jean-Christophe Deneuville, Phillipe Gaborit, Shay Gueron, Tim Guneysu, Carlos Aguilar-Melchor, Rafael Misoczki, Edoardo Persichetti, Nicolas Sendrier, Jean-Pierre Tillich, Gilles Zémor, Valentin Vasseur, Santosh Ghosh, and Jan Richter-Brokmann. BIKE. Technical report, National Institute of Standards and Technology, 2022. available at https://csrc.nist.gov/Projects/post-quantum-cryptography/round-4-submissions.

[ABC+22] Martin R. Albrecht, Daniel J. Bernstein, Tung Chou, Carlos Cid, Jan Gilcher, Tanja Lange, Varun Maram, Ingo von Maurich, Rafael Misoczki, Ruben Niederhagen, Kenneth G. Paterson, Edoardo Persichetti, Christiane Peters, Peter Schwabe, Nicolas Sendrier, Jakub Szefer, Cen Jung Tjhai, Martin Tomlinson, and Wen Wang. Classic McEliece. Technical report, National Institute of Standards and Technology, 2022. available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-4-submissions.

[ABG+23] Carlos Aguilar-Melchor, Thomas Bailleux, Jason Goertzen, David Joseph, and Douglas Stebila. TurboTLS: TLS connection establishment with 1 less round trip, 2023. URL: https://arxiv.org/abs/2302.05311, arXiv:2302.05311.

[ADPS16] Erdem Alkim, Léo Ducas, Thomas Pöppelmann, and Peter Schwabe. Post-quantum key exchange - A new hope. In Thorsten Holz and Stefan Savage, editors, *USENIX Security 2016: 25th USENIX Security Symposium*, pages 327–343. USENIX Association, August 2016.

[AWS23] AWS. Github s2n-TLS/PQ-Crypto. https://github.com/aws/s2n-tls/tree/v1.3.56/pq-crypto, July 2023.

[BAA+19] Nina Bindel, Sedat Akleylek, Erdem Alkim, Paulo S. L. M. Barreto, Johannes Buchmann, Edward Eaton, Gus Gutoski, Juliane Kramer, Patrick Longa, Harun Polat, Jefferson E. Ricardini, and Gustavo Zanon. qTESLA. Technical report, National Institute of Standards and Technology, 2019. available at https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-2-submissions.

[Bar20] Elaine Barker. Recommendation for Key Management Part 1: General. Technical Report NIST SP 800-57pt1r5, National Institute of Standards and Technology, May 2020. doi:10.6028/NIST.SP.800-57pt1r5.

[BBC+20] Daniel J. Bernstein, Billy Bob Brumley, Ming-Shing Chen, Chitchanok Chuengsatiansup, Tanja Lange, Adrian Marotzke, Bo-Yuan Peng, Nicola Tuveri, Christine van Vredendaal, and Bo-Yin Yang. NTRU Prime. Technical report, National Institute of Standards and Technology, 2020. available at https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-3-submissions.

[BBCT22] Daniel J. Bernstein, Billy Bob Brumley, Ming-Shing Chen, and Nicola Tuveri. OpenSSLNTRU: Faster post-quantum TLS key exchange. In Kevin R. B. Butler and Kurt Thomas, editors, *USENIX Security 2022: 31st USENIX Security Symposium*, pages 845–862. USENIX Association, August 2022.

[BBF+19] Nina Bindel, Jacqueline Brendel, Marc Fischlin, Brian Goncalves, and Douglas Stebila. Hybrid key encapsulation mechanisms and authenticated key exchange. In Jintai Ding and Rainer Steinwandt, editors, *Post-Quantum Cryptography - 10th International Conference, PQCrypto 2019*, pages 206–226. Springer, Heidelberg, 2019. doi:10.1007/978-3-030-25510-7_12.

[BC20] Utsav Banerjee and Anantha P. Chandrakasan. Efficient post-quantum TLS handshakes using identity-based key exchange from lattices. In *ICC 2020 - 2020 IEEE International Conference on Communications (ICC)*, pages 1–6, June 2020. doi:10.1109/ICC40277.2020.9148829.

[BCD+16] Joppe W. Bos, Craig Costello, Léo Ducas, Ilya Mironov, Michael Naehrig, Valeria Nikolaenko, Ananth Raghunathan, and Douglas Stebila. Frodo: Take off the ring! Practical, quantum-secure key exchange from LWE. In Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi, editors, *ACM CCS 2016: 23rd Conference on Computer and Communications Security*, pages 1006–1018. ACM Press, October 2016. doi:10.1145/2976749.2978425.

[BCD+24] Manuel Barbosa, Deirdre Connolly, João Diogo Duarte, Aaron Kaiser, Peter Schwabe, Karolin Varner, and Bas Westerbaan. X-wing. *IACR Communications in Cryptology*, 1(1), 2024. doi:10.62056/a3qj89n4e.

[BCNS15] Joppe W. Bos, Craig Costello, Michael Naehrig, and Douglas Stebila. Post-quantum key exchange for the TLS protocol from the ring learning with errors problem. In *2015 IEEE Symposium on Security and Privacy*, pages 553–570. IEEE Computer Society Press, May 2015. doi:10.1109/SP.2015.40.

[Ber09] Daniel J. Bernstein. Cost analysis of hash collisions: Will quantum computers make SHARCS obsolete? https://cr.yp.to/hash/collisioncost-20090823.pdf, 2009.

[Beu22] Ward Beullens. Breaking rainbow takes a weekend on a laptop. In Yevgeniy Dodis and Thomas Shrimpton, editors, *Advances in Cryptology – CRYPTO 2022, Part II*, volume 13508 of *Lecture Notes in Computer Science*, pages 464–479. Springer, Heidelberg, August 2022. doi:10.1007/978-3-031-15979-4_16.

[BHMS17] Nina Bindel, Udyani Herath, Matthew McKague, and Douglas Stebila. Transitioning to a quantum-resistant public key infrastructure. In Tanja Lange and Tsuyoshi Takagi, editors, *Post-Quantum Cryptography - 8th International Workshop, PQCrypto 2017*, pages 384–405. Springer, Heidelberg, 2017. doi:10.1007/978-3-319-59879-6_22.

[BHT98] Gilles Brassard, Peter Høyer, and Alain Tapp. Quantum cryptanalysis of hash and claw-free functions. In Claudio L. Lucchesi and Arnaldo V. Moura, editors, *LATIN 1998: Theoretical Informatics, 3rd Latin American Symposium*, volume 1380 of *Lecture Notes in Computer Science*, pages 163–169. Springer, Heidelberg, April 1998. doi:10.1007/bfb0054319.

[BSI21] BSI. Migration to Post Quantum Cryptography. https://www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/Crypto/Migration_to_Post_Quantum_Cryptography.pdf, 2021.

[BSI23] BSI. Cryptographic Mechanisms: Recommendations and Key Lengths. https://www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/Publications/TechGuidelines/TG02102/BSI-TR-02102-1.pdf, 2023.

[BSKNS20] Kevin Bürstinghaus-Steinbach, Christoph Krauß, Ruben Niederhagen, and Michael Schneider. Post-quantum TLS on embedded systems: Integrating and evaluating Kyber and SPHINCS+ with mbed TLS. In Hung-Min Sun, Shiuh-Pyng Shieh, Guofei Gu, and Giuseppe Ateniese, editors, *ASIACCS 20: 15th ACM Symposium on Information, Computer and Communications Security*, pages 841–852. ACM Press, October 2020. doi:10.1145/3320269.3384725.

[CAD+20] David A. Cooper, Daniel C. Apon, Quynh H. Dang, Michael S. Davidson, Morris J. Dworkin, and Carl A. Miller. Recommendation for Stateful Hash-Based Signature Schemes. Technical Report NIST SP 800-208, National Institute of Standards and Technology, October 2020. doi:10.6028/NIST.SP.800-208.

[CC21] Matt Campagna and Eric Crockett. Hybrid Post-Quantum Key Encapsulation Methods (PQ KEM) for Transport Layer Security 1.2 (TLS). Internet-Draft draft-campagna-tls-bike-sike-hybrid-07, Internet Engineering Task Force, September 2021. Work in Progress. URL: https://datatracker.ietf.org/doc/draft-campagna-tls-bike-sike-hybrid/07/.

[CCSCD+23] Fabio Campos, Jorge Chavez-Saab, Jesús-Javier Chi-Domínguez, Michael Meyer, Krijn Reijnders, Francisco Rodríguez-Henríquez, Peter Schwabe, and Thom Wiggers. Optimizations and practicality of high-security CSIDH. Cryptology ePrint Archive, Paper 2023/793, 2023. https://eprint.iacr.org/2023/793.

[CD23] Wouter Castryck and Thomas Decru. An efficient key recovery attack on SIDH. In Carmit Hazay and Martijn Stam, editors, *Advances in Cryptology – EUROCRYPT 2023, Part V*, volume 14008 of *Lecture Notes in Computer Science*, pages 423–447. Springer, Heidelberg, April 2023. doi:10.1007/978-3-031-30589-4_15.

[CDH+20] Cong Chen, Oussama Danba, Jeffrey Hoffstein, Andreas Hulsing, Joost Rijneveld, John M. Schanck, Peter Schwabe, William Whyte, Zhenfei Zhang, Tsunekazu Saito, Takashi Yamakawa, and Keita Xagawa. NTRU. Technical report, National Institute of Standards and Technology, 2020. available at https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-3-submissions.

[CFM+20] Antoine Casanova, Jean-Charles Faugère, Gilles Macario-Rat, Jacques Patarin, Ludovic Perret, and Jocelyn Ryckeghem. GeMSS. Technical report, National Institute of Standards and Technology, 2020. available at https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-3-submissions.

[CFS+21] Sofía Celi, Armando Faz-Hernández, Nick Sullivan, Goutam Tamvada, Luke Valenta, Thom Wiggers, Bas Westerbaan, and Christopher A. Wood. Implementing and measuring KEMTLS. In Patrick Longa and Carla Ràfols, editors, *Progress in Cryptology - LATINCRYPT 2021: 7th International Conference on Cryptology and Information Security in Latin America*, volume 12912 of *Lecture Notes in Computer Science*, pages 88–107, Bogotá, Colombia, October 2021. Springer, Heidelberg. doi:10.1007/978-3-030-88238-9_5.

[CHSv16] Cas Cremers, Marko Horvat, Sam Scott, and Thyla van der Merwe. Automated analysis and verification of TLS 1.3: 0-RTT, resumption and delayed authentication. In *2016 IEEE Symposium on Security and Privacy*, pages 470–485. IEEE Computer Society Press, May 2016. doi:10.1109/SP.2016.35.

[CHSW22] Sofía Celi, Jonathan Hoyland, Douglas Stebila, and Thom Wiggers. A tale of two models: Formal verification of KEMTLS via Tamarin. In Vijayalakshmi Atluri, Roberto Di Pietro, Christian Damsgaard Jensen, and Weizhi Meng, editors, *ESORICS 2022: 27th European Symposium on Research in Computer Security, Part III*, volume 13556 of *Lecture Notes in Computer Science*, pages 63–83. Springer, Heidelberg, September 2022. doi:10.1007/978-3-031-17143-7_4.

[CJKK22] Vincent Cheval, Charlie Jacomme, Steve Kremer, and Robert Künnemann. SAPIC+: protocol verifiers of the world, unite! In Kevin R. B. Butler and Kurt Thomas, editors, *USENIX Security 2022: 31st USENIX Security Symposium*, pages 3935–3952. USENIX Association, August 2022.

[CJL+16] Lily Chen, Stephen Jordan, Yi-Kai Liu, Dustin Moody, Rene Peralta, Ray Perlner, and Daniel Smith-Tone. Report on Post-Quantum Cryptography. Technical Report NIST IR 8105, National Institute of Standards and Technology, April 2016. doi:10.6028/NIST.IR.8105.

[CLM+18] Wouter Castryck, Tanja Lange, Chloe Martindale, Lorenz Panny, and Joost Renes. CSIDH: An efficient post-quantum commutative group action. In Thomas Peyrin and Steven Galbraith, editors, *Advances in Cryptology – ASIACRYPT 2018, Part III*, volume 11274 of *Lecture Notes in Computer Science*, pages 395–427. Springer, Heidelberg, December 2018. doi:10.1007/978-3-030-03332-3_15.

[CPS19] Eric Crockett, Christian Paquin, and Douglas 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.

[DAL+17] Jintai Ding, Saed Alsayigh, Jean Lancrenon, Saraswathy RV, and Michael Snook. Provably secure password authenticated key exchange based on RLWE for the post-quantum world. In Helena Handschuh, editor, *Topics in Cryptology – CT-RSA 2017*, volume 10159 of *Lecture Notes in Computer Science*, pages 183–204. Springer, Heidelberg, February 2017. doi:10.1007/978-3-319-52153-4_11.

[DCP+20] Jintai Ding, Ming-Shing Chen, Albrecht Petzoldt, Dieter Schmidt, Bo-Yin Yang, Matthias J. Kannwischer, and Jacques Patarin. Rainbow. Technical report, National Institute of Standards and Technology, 2020. available at https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-3-submissions.

[DDV+20] Alfonso Francisco De Abiega-L'Eglisse, Kevin Andrae Delgado-Vargas, Fernando Quetzalcoatl Valencia-Rodriguez, Victor Gerardo Gonzalez-Quiroga, Gina Gallegos-Garcia, and Mariko Nakano-Miyatake. Performance of New Hope and CRYSTALS-Dilithium Postquantum Schemes in the Transport Layer Security Protocol. *IEEE Access*, 8:213968–213980, 2020. doi:10.1109/ACCESS.2020.3040324.

[DFGS15] Benjamin Dowling, Marc Fischlin, Felix Günther, and Douglas Stebila. A cryptographic analysis of the TLS 1.3 handshake protocol candidates. In Indrajit Ray, Ninghui Li, and Christopher Kruegel, editors, *ACM CCS 2015: 22nd Conference on Computer and Communications Security*, pages 1197–1210. ACM Press, October 2015. doi:10.1145/2810103.2813653.

[DG22] Ronny Döring and Marc Geitz. Post-Quantum Cryptography in Use: Empirical Analysis of the TLS Handshake Performance. In *NOMS 2022-2022 IEEE/IFIP Network Operations and Management Symposium*, pages 1–5, April 2022. doi:10.1109/NOMS54207.2022.9789913.

[DKR+20] Jan-Pieter D'Anvers, Angshuman Karmakar, Sujoy Sinha Roy, Frederik Vercauteren, Jose Maria Bermudo Mera, Michiel Van Beirendonck, and Andrea Basso. SABER. Technical report, National Institute of Standards and Technology, 2020. available at https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-3-submissions.

[DM23a] Douglas Stebila and Michele Mosca. Open Quantum Safe benchmarking, KEM performance. https://web.archive.org/web/20230630133536/https://openquantumsafe.org/benchmarking/visualization/speed_kem.html, June 2023.

[DM23b] Douglas Stebila and Michele Mosca. Open Quantum Safe benchmarking, TLS Handshake performance. https://web.archive.org/web/20230630104243/https://openquantumsafe.org/benchmarking/visualization/handshakes.html, June 2023.

[ETS17] ETSI. Limits to Quantum Computing applied to symmetric key sizes. Technical Report ETSI GR QSC 006, ETSI, February 2017.

[ETS21] ETSI. Electronic Signatures and Infrastructures (ESI); General Policy Requirements for Trust Service Providers. Technical Report EN 319 401 V2.3.1, ETSI, May 2021.

[Fro23] FrodoKEM submitters. FrodoKEM website. https://frodokem.org/, 2023.

[GDL+18] Xinwei Gao, Jintai Ding, Lin Li, Saraswathy RV, and Jiqiang Liu. Efficient Implementation of Password-based Authenticated Key Exchange from RLWE and Post-Quantum TLS. *International Journal of Network Security*, 20(5):923–930, September 2018. doi:10.6633/IJNS.201809_20(5).14.

[GdNC+23] Alexandre Augusto Giron, João Pedro Adami do Nascimento, Ricardo Custódio, Lucas Pandolfo Perin, and Víctor Mateu. Post-quantum hybrid KEMTLS performance in simulated and real network environments. In Abdelrahaman Aly and Mehdi Tibouchi, editors, *Progress in Cryptology - LATINCRYPT 2023: 8th International Conference on Cryptology and Information Security in Latin America*, volume 14168 of *Lecture Notes in Computer Science*, pages 293–312, Quito, Ecuador, October 2023. Springer, Heidelberg. doi:10.1007/978-3-031-44469-2_15.

[GHP18] Federico Giacon, Felix Heuer, and Bertram Poettering. KEM combiners. In Michel Abdalla and Ricardo Dahab, editors, *PKC 2018: 21st International Conference on Theory and Practice of Public Key Cryptography, Part I*, volume 10769 of *Lecture Notes in Computer Science*, pages 190–218. Springer, Heidelberg, March 2018. doi:10.1007/978-3-319-76578-5_7.

[GLD+17] Xinwei Gao, Lin Li, Jintai Ding, Jiqiang Liu, R. V. Saraswathy, and Zhe Liu. Fast Discretized Gaussian Sampling and Post-quantum TLS Ciphersuite. In *Information Security Practice and Experience*, volume 10701, pages 551–565. Springer International Publishing, Cham, 2017. doi:10.1007/978-3-319-72359-4_33.

[Goo22] Google. Why Google now uses post-quantum cryptography for internal comms. https://cloud.google.com/blog/products/identity-security/why-google-now-uses-post-quantum-cryptography-for-internal-comms, 2022.

[Goo23a] Google. General-purpose machine family for compute engine. https://cloud.google.com/compute/docs/general-purpose-machines, 2023.

[Goo23b] Google. Protecting Chrome Traffic with Hybrid Kyber KEM. https://blog.chromium.org/2023/08/protecting-chrome-traffic-with-hybrid.html, August 2023.

[Gro96] Lov K. Grover. A fast quantum mechanical algorithm for database search. In *28th Annual ACM Symposium on Theory of Computing*, pages 212–219. ACM Press, May 1996. doi:10.1145/237814.237866.

[GW22] Ruben Gonzalez and Thom Wiggers. KEMTLS vs. Post-quantum TLS: Performance on Embedded Systems. In Lejla Batina, Stjepan Picek, and Mainack Mondal, editors, *Security, Privacy, and Applied Cryptography Engineering*, Lecture Notes in Computer Science, pages 99–117, Cham, 2022. Springer Nature Switzerland. doi:10.1007/978-3-031-22829-2_6.

[HBD+22] Andreas Hülsing, Daniel J. Bernstein, Christoph Dobraunig, Maria Eichlseder, Scott Fluhrer, Stefan-Lukas Gazdag, Panos Kampanakis, Stefan Kölbl, Tanja Lange, Martin M. Lauridsen, Florian Mendel, Ruben Niederhagen, Christian Rechberger, Joost Rijneveld, Peter Schwabe, Jean-Philippe Aumasson, Bas Westerbaan, and Ward Beullens. SPHINCS+. Technical report, National Institute of Standards and Technology, 2022. available at https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022.

[HBG+18] Andreas Huelsing, Denis Butin, Stefan-Lukas Gazdag, Joost Rijneveld, and Aziz Mohaisen. XMSS: eXtended Merkle Signature Scheme. RFC 8391, 2018. URL: https://www.rfc-editor.org/info/rfc8391, doi:10.17487/RFC8391.

[Hop19] Andrew Hopkins. Post-quantum TLS now supported in AWS KMS - AWS Security Blog, November 2019. URL: https://aws.amazon.com/blogs/security/post-quantum-tls-now-supported-in-aws-kms/.

[HW23] James Howe and Bas Westerbaan. Benchmarking and analysing the NIST PQC lattice-based signature schemes standards on the ARM Cortex M7. In Nadia El Mrabet, Luca De Feo, and Sylvain Duquesne, editors, *AFRICACRYPT 23: 14th International Conference on Cryptology in Africa*, volume 14064 of *Lecture Notes in Computer Science*, pages 442–462. Springer Nature, July 2023. doi:10.1007/978-3-031-37679-5_19.

[ITU19] ITU. Recommendation ITU-T X.509 (2019) | ISO/IEC 9594-8:2020, Information technology – Open Systems Interconnection – The Directory: Public-key and attribute certificate frameworks, October 2019.

[JAC+20] David Jao, Reza Azarderakhsh, Matthew Campagna, Craig Costello, Luca De Feo, Basil Hess, Amir Jalali, Brian Koziel, Brian LaMacchia, Patrick Longa, Michael Naehrig, Joost Renes, Vladimir Soukharev, David Urbanik, Geovandro Pereira, Koray Karabina, and Aaron Hutchinson. SIKE. Technical report, National Institute of Standards and Technology, 2020. available at https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-3-submissions.

[KK18] Franziskus Kiefer and Kris Kwiatkowski. Hybrid ECDHE-SIDH Key Exchange for TLS. https://datatracker.ietf.org/doc/draft-kiefer-tls-ecdhe-sidh/00/, November 2018.

[KPDG18] Panos Kampanakis, Peter Panburana, Ellie Daw, and Daniel Van Geest. The viability of post-quantum X.509 certificates. Cryptology ePrint Archive, Report 2018/063, 2018. https://eprint.iacr.org/2018/063.

[KS19] Panos Kampanakis and Dimitrios Sikeridis. Two PQ signature use-cases: Non-issues, challenges and potential solutions. Cryptology ePrint Archive, Report 2019/1276, 2019. https://eprint.iacr.org/2019/1276.

[KV19] Kris Kwiatkowski and Luke Valenta. The TLS Post-Quantum Experiment. https://blog.cloudflare.com/the-tls-post-quantum-experiment/, October 2019.

[KW16] Hugo Krawczyk and Hoeteck Wee. The OPTLS protocol and TLS 1.3. In *2016 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 81–96, 2016. doi:10.1109/EuroSP.2016.18.

[KZ20] Daniel Kales and Greg Zaverucha. An attack on some signature schemes constructed from five-pass identification schemes. In Stephan Krenn, Haya Shulman, and Serge Vaudenay, editors, *CANS 20: 19th International Conference on Cryptology and Network Security*, volume 12579 of *Lecture Notes in Computer Science*, pages 3–22. Springer, Heidelberg, December 2020. doi:10.1007/978-3-030-65411-5_1.

[Lan16] Adam Langley. ImperialViolet - CECPQ1 results. https://www.imperialviolet.org/2016/11/28/cecpq1.html, November 2016.

[Lan18] Adam Langley. ImperialViolet - CECPQ2. https://www.imperialviolet.org/2018/12/12/cecpq2.html, December 2018.

[LDK+22] Vadim Lyubashevsky, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Peter Schwabe, Gregor Seiler, Damien Stehlé, and Shi Bai. CRYSTALS-DILITHIUM. Technical report, National Institute of Standards and Technology, 2022. available at https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022.

[LIM21] Fukang Liu, Takanori Isobe, and Willi Meier. Cryptanalysis of full LowMC and LowMC-M with algebraic techniques. In Tal Malkin and Chris Peikert, editors, *Advances in Cryptology – CRYPTO 2021, Part III*, volume 12827 of *Lecture Notes in Computer Science*, pages 368–401, Virtual Event, August 2021. Springer, Heidelberg. doi:10.1007/978-3-030-84252-9_13.

[LSW+22] Fukang Liu, Santanu Sarkar, Gaoli Wang, Willi Meier, and Takanori Isobe. Algebraic meet-in-the-middle attack on LowMC. In Shweta Agrawal and Dongdai Lin, editors, *Advances in Cryptology – ASIACRYPT 2022, Part I*, volume 13791 of *Lecture Notes in Computer Science*, pages 225–255. Springer, Heidelberg, December 2022. doi:10.1007/978-3-031-22963-3_8.

[MAA+20] Dustin Moody, Gorjan Alagic, Daniel C Apon, David A Cooper, Quynh H Dang, John M Kelsey, Yi-Kai Liu, Carl A Miller, Rene C Peralta, Ray A Perlner, Angela Y Robinson, Daniel C Smith-Tone, and Jacob Alperin-Sheriff. Status Report on the Second Round of the NIST Post-Quantum Cryptography Standardization Process. Technical Report NIST IR 8309, National Institute of Standards and Technology, July 2020. doi:10.6028/NIST.IR.8309.

[MCF19] David McGrew, Michael Curcio, and Scott Fluhrer. Leighton-Micali Hash-Based Signatures. Request for Comments RFC 8554, Internet Engineering Task Force, April 2019. doi:10.17487/RFC8554.

[MP22] Michele Mosca and Marco Piani. 2022 Quantum Threat Timeline Report. Technical report, Global Risk Institute, 2022. URL: https://globalriskinstitute.org/mp-files/2022-quantum-threat-timeline-report-dec.pdf/.

[MS22] Dominik Marchsreiter and Johanna Sepúlveda. Hybrid Post-Quantum Enhanced TLS 1.3 on Embedded Devices. In *2022 25th Euromicro Conference on Digital System Design (DSD)*, pages 905–912, August 2022. doi:10.1109/DSD57027.2022.00127.

[MST21] John Preuß Mattsson, Ben Smeets, and Erik Thormarker. Quantum technology and its impact on security in mobile networks. *Ericsson Technology Review*, 2021(12):2–12, 2021. doi:10.23919/ETR.2021.9904724.

[NAB+20] Michael Naehrig, Erdem Alkim, Joppe Bos, Léo Ducas, Karen Easterbrook, Brian LaMacchia, Patrick Longa, Ilya Mironov, Valeria Nikolaenko, Christopher Peikert, Ananth Raghunathan, and Douglas Stebila. FrodoKEM. Technical report, National Institute of Standards and Technology, 2020. available at https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-3-submissions.

[NIS16] NIST. Submission Requirements and Evaluation Criteria for the Post-Quantum Cryptography Standardization Process. https://csrc.nist.ri p/groups/ST/post-quantum-crypto/documents/call-for-proposals-final-dec-2016.pdf, December 2016.

[NIS22] NIST. Call for Additional Digital Signature Schemes for the Post-Quantum Cryptography Standardization Process. https://csrc.nist.gov/csrc/media/Projects/pqc-dig-sig/documents/call-for-proposals-dig-sig-sept-2022.pdf, October 2022.

[NSA22] NSA. Commercial National Security Algorithm Suite 2.0. https://media.defense.gov/2022/Sep/07/2003071834/-1/-1/0/CSA_CNSA_2.0_ALGORITHMS_.PDF, 2022.

[PAA+19] Thomas Pöppelmann, Erdem Alkim, Roberto Avanzi, Joppe Bos, Léo Ducas, Antonio de la Piedra, Peter Schwabe, Douglas Stebila, Martin R. Albrecht, Emmanuela Orsini, Valery Osheter, Kenneth G. Paterson, Guy Peer, and Nigel P. Smart. NewHope. Technical report, National Institute of Standards and Technology, 2019. available at https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-2-submissions.

[Pei20] Chris Peikert. He gives C-sieves on the CSIDH. In Anne Canteaut and Yuval Ishai, editors, *Advances in Cryptology – EUROCRYPT 2020, Part II*, volume 12106 of *Lecture Notes in Computer Science*, pages 463–492. Springer, Heidelberg, May 2020. doi:10.1007/978-3-030-45724-2_16.

[PFH+22] Thomas Prest, Pierre-Alain Fouque, Jeffrey Hoffstein, Paul Kirchner, Vadim Lyubashevsky, Thomas Pornin, Thomas Ricosset, Gregor Seiler, William Whyte, and Zhenfei Zhang. FALCON. Technical report, National Institute of Standards and Technology, 2022. available at https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022.

[PKLN22] Sebastian Paul, Yulia Kuzovkova, Norman Lahr, and Ruben Niederhagen. Mixed certificate chains for the transition to post-quantum authentication in TLS 1.3. In Yuji Suga, Kouichi Sakurai, Xuhua Ding, and Kazue Sako, editors, *ASIACCS 22: 17th ACM Symposium on Information, Computer and Communications Security*, pages 727–740. ACM Press, May / June 2022. doi:10.1145/3488932.3497755.

[Por19] Thomas Pornin. New efficient, constant-time implementations of Falcon. Cryptology ePrint Archive, Report 2019/893, 2019. https://eprint.iacr.org/2019/893.

[pro23a] The HAPKIDO project. HAPKIDO: For quantum-safe Public Key Infrastructures. https://hapkido.tno.nl/, 2023.

[pro23b] The Open Quantum Safe project. Github repository: OQS liboqs library. https://github.com/open-quantum-safe/liboqs, 2023.

[pro23c] The Open Quantum Safe project. Github repository: OQS OpenSSL 3 provider. https://github.com/open-quantum-safe/oqs-provider, 2023.

[pro23d] The Open Quantum Safe project. Github repository: OQS OpenSSL fork. https://github.com/open-quantum-safe/openssl, 2023.

[pro23e] The OpenSSL project. Github repository: OpenSSL. https://github.com/openssl/openssl, 2023.

[PSS21] Sebastian Paul, Felix Schick, and Jan Seedorf. TPM-Based Post-Quantum Cryptography: A Case Study on Quantum-Resistant and Mutually Authenticated TLS for IoT Environments. In *Proceedings of the 16th International Conference on Availability, Reliability and Security*, ARES 21, pages 1–10. Association for Computing Machinery, August 2021. doi:10.1145/3465481.3465747.

[PST20] Christian Paquin, Douglas Stebila, and Goutam Tamvada. Benchmarking post-quantum cryptography in TLS. In Jintai Ding and Jean-Pierre Tillich, editors, *Post-Quantum Cryptography - 11th International Conference, PQCrypto 2020*, pages 72–91. Springer, Heidelberg, 2020. doi:10.1007/978-3-030-44223-1_5.

[Res18] Eric Rescorla. The Transport Layer Security (TLS) Protocol Version 1.3. RFC 8446, August 2018. URL: https://www.rfc-editor.org/info/rfc8446, doi:10.17487/RFC8446.

[SAB+18] N Smart, M Abdalla, E Bjørstad, C Cid, and B Gierlichs. Algorithms, key size and protocols report (2018). Technical Report Project 645421, ECRYPT - CSA, 2018. URL: https://www.ecrypt.eu.org/csa/documents/D5.4-FinalAlgKeySizeProt.pdf.

[SAB+22] Peter Schwabe, Roberto Avanzi, Joppe Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M. Schanck, Gregor Seiler, Damien Stehlé, and Jintai Ding. CRYSTALS-KYBER. Technical report, National Institute of Standards and Technology, 2022. available at https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022.

[SCH+19] Simona Samardjiska, Ming-Shing Chen, Andreas Hulsing, Joost Rijneveld, and Peter Schwabe. MQDSS. Technical report, National Institute of Standards and Technology, 2019. available at https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-2-submissions.

[SFG24] D. Stebila, S. Fluhrer, and S. Gueron. Internet-Draft: Hybrid key exchange in TLS 1.3. https://datatracker.ietf.org/doc/draft-ietf-tls-hybrid-design/10/, April 2024.

[SKD20a] Dimitrios Sikeridis, Panos Kampanakis, and Michael Devetsikiotis. Assessing the Overhead of Post-Quantum Cryptography in TLS 1.3 and SSH. In *Proceedings of the 16th International Conference on Emerging Networking EXperiments and Technologies*, CoNEXT ’20, pages 149–156, New York, NY, USA, 2020. Association for Computing Machinery. doi:10.1145/3386367.3431305.

[SKD20b] Dimitrios Sikeridis, Panos Kampanakis, and Michael Devetsikiotis. Post-quantum authentication in TLS 1.3: A performance study. In *ISOC Network and Distributed System Security Symposium – NDSS 2020*. The Internet Society, February 2020. doi:10.14722/ndss.2020.24203.

[SM16] Douglas Stebila and Michele Mosca. Post-quantum key exchange for the internet and the open quantum safe project. In Roberto Avanzi and Howard M. Heys, editors, *SAC 2016: 23rd Annual International Workshop on Selected Areas in Cryptography*, volume 10532 of *Lecture Notes in Computer Science*, pages 14–37. Springer, Heidelberg, August 2016. doi:10.1007/978-3-319-69453-5_2.

[SS17] John M. Schanck and Douglas Stebila. A Transport Layer Security (TLS) Extension For Establishing An Additional Shared Secret. https://datatracker.ietf.org/doc/draft-schanck-tls-additional-keyshare/00/, April 2017.

[SSW20] Peter Schwabe, Douglas Stebila, and Thom Wiggers. Post-quantum TLS without handshake signatures. In Jay Ligatti, Xinming Ou, Jonathan Katz, and Giovanni Vigna, editors, *ACM CCS 2020: 27th Conference on Computer and Communications Security*, pages 1461–1480. ACM Press, November 2020. doi:10.1145/3372297.3423350.

[SSW21] Peter Schwabe, Douglas Stebila, and Thom Wiggers. More efficient post-quantum KEMTLS with pre-distributed public keys. In Elisa Bertino, Haya Shulman, and Michael Waidner, editors, *ESORICS 2021: 26th European Symposium on Research in Computer Security, Part I*, volume 12972 of *Lecture Notes in Computer Science*, pages 3–22. Springer, Heidelberg, October 2021. doi:10.1007/978-3-030-88418-5_1.

[SWZ16] John M. Schanck, William Whyte, and Zhenfei Zhang. Quantum-Safe Hybrid (QSH) Ciphersuite for Transport Layer Security (TLS) version 1.2. https://datatracker.ietf.org/doc/draft-whyte-qsh-tls12/02/, 2016.

[TLF+22] George Tasopoulos, Jinhui Li, Apostolos P. Fournaris, Raymond K. Zhao, Amin Sakzad, and Ron Steinfeld. Performance Evaluation of Post-Quantum TLS 1.3 on Resource-Constrained Embedded Systems. In Chunhua Su, Dimitris Gritzalis, and Vincenzo Piuri, editors, *Information Security Practice and Experience*, Lecture Notes in Computer Science, pages 432–451. Springer International Publishing, 2022. doi:10.1007/978-3-031-21280-2_24.

[TPD21] Chengdong Tao, Albrecht Petzoldt, and Jintai Ding. Efficient key recovery for all HFE signature variants. In Tal Malkin and Chris Peikert, editors, *Advances in Cryptology – CRYPTO 2021, Part I*, volume 12825 of *Lecture Notes in Computer Science*, pages 70–93, Virtual Event, August 2021. Springer, Heidelberg. doi:10.1007/978-3-030-84242-0_4.

[Wel20] D. L. Weller. Incorporating post-quantum cryptography in a microservice environment. https://rp.os3.nl/2019-2020/p13/report.pdf, February 2020.

[Wes22] Bas Westerbaan. NIST's pleasant post-quantum surprise. https://blog.cloudflare.com/nist-post-quantum-surprise/, July 2022.

[Wes24] Bas Westerbaan. Re: [CFRG] [TLS] X-Wing: The go-to PQ/T hybrid KEM?, January 2024. URL: https://mailarchive.ietf.org/arch/msg/cfrg/R6LDb44AIKSNClPHeAp9JuLBS8s/.

[Wig24] Thom Wiggers. *Post-Quantum TLS*. PhD thesis, Radboud University, Nijmegen, The Netherlands, 2024. URL: https://thomwiggers.nl/publication/thesis/.

[WS23] Bas Westerbaan and Douglas Stebila. X25519Kyber768Draft00 hybrid post-quantum key agreement. https://datatracker.ietf.org/doc/draft-tls-westerbaan-xyber768d00/03/, September 2023.

[WZFG17] W. Whyte, Z. Zhang, S. Fluhrer, and O. Garcia-Morchon. Internet-Draft: Quantum-Safe Hybrid (QSH) Key Exchange for Transport Layer Security (TLS) version 1.3. https://datatracker.ietf.org/doc/draft-whyte-qsh-tls13/06/, October 2017.

[Zal99] Christof Zalka. Grover's quantum searching algorithm is optimal. *Physical Review A*, 60(4):2746–2751, October 1999. arXiv:quant-ph/9711070, doi:10.1103/PhysRevA.60.2746.

[ZCD+20] Greg Zaverucha, Melissa Chase, David Derler, Steven Goldfeder, Claudio Orlandi, Sebastian Ramacher, Christian Rechberger, Daniel Slamanig, Jonathan Katz, Xiao Wang, Vladmir Kolesnikov, and Daniel Kales. Picnic. Technical report, National Institute of Standards and Technology, 2020. available at https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-3-submissions.

[ZZD+15] Jiang Zhang, Zhenfeng Zhang, Jintai Ding, Michael Snook, and Özgür Dagdelen. Authenticated key exchange from ideal lattices. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology – EUROCRYPT 2015, Part II*, volume 9057 of *Lecture Notes in Computer Science*, pages 719–751. Springer, Heidelberg, April 2015. doi:10.1007/978-3-662-46803-6_24.

A Our post-quantum TLS 1.3 experiment

We conducted our own post-quantum TLS 1.3 experiment to measure the performance of purely and hybrid post-quantum authentication (Sections 4.1 and 4.2) and post-quantum key exchange (Sections 5.1 and 5.2.1). While our purely post-quantum results were already presented in Table 2 and Table 3, we provide more extensive results, which include hybrid post-quantum key exchange and authentication, and details about our methodology in this section.

A.1 Methodology

Hardware. We used two `e2-highcpu-16` machines hosted on Google Cloud, one for the server and one for the clients. These machines have a 16-core x86 CPU (AMD/Intel based on availability [Goo23a]) with 16GB of RAM. The RTT was 7ms and the bandwidth was 1.08Gb/s.

Software. The OS was Ubuntu 22.04.3 LTS. We used the modified OpenSSL libraries from the Open Quantum Safe project [SM16]. Specifically, we used the *oqsprovider* [pro23c] in combination with OpenSSL 3.2 alpha 2 [pro23e].[11]. For benchmarking purely and hybrid post-quantum signatures and hybrid KEMs, we used the latest oqsprovider 0.5.2 (compiled with liboqs 0.9.0 [pro23b]). For benchmarking purely post-quantum KEMs we had to use an older version (0.4.0) of the oqsprovider (compiled with liboqs 0.7.2) because the newest version only supports the algorithms which are still in the NIST PQC competition. Liboqs was compiled with default (auto) optimization target.

We provide the helper scripts used to re-run our experiments at https://github.com/PQTLSSOK/benchmarks with more details in the provided README and comments inside the scripts.

[11] We used the alpha version because it allows to benchmark post-quantum signatures in TLS 1.3 with the oqsprovider. Otherwise, it is necessary to use the deprecated OpenSSL 1.1.1 fork that is available at [pro23d].

Measurement and instantiation. We measured the TLS performance using the command `openssl s_time`, which outputs the number of connections per second. Each server process was established using the `openssl s_server` command. We ran 32 of these processes in parallel (32 servers and 32 clients) with 10 repetitions and `s_time` duration of 60s. When benchmarking signatures, the certificate chain length was 2 and the key exchange algorithm was x25519. When benchmarking key exchange, the certificate chain length was 2 and the signature algorithm was ECDSA with (P-256).

A.2 Results

The TLS slowdown coefficient is defined as $\frac{t_{pq}}{t_c}$, where t_{pq} is the mean of connections per second of the post-quantum scheme and t_c is the mean of connections per second of x25519 (for KEMs) or Ed25519 (for signatures). The specific parameter set is noted in parentheses, such as the security level and possibly other details.

The results of our evaluation of purely post-quantum signatures in TLS 1.3 are presented in Tab. 4. These results are visualized in Fig. 3. The results of our evaluation of purely post-quantum KEMs in TLS 1.3 are presented in Tab. 6. These results are visualized in Fig. 5. Note that we excluded from our results two NTRU level 5 parameter sets (NTRU-HPS-4096-1229, NTRU-HRSS-1373) because liboqs provides only their non-optimized implementation (compared to the rest).

The results of our evaluation of hybrid post-quantum signatures in TLS 1.3 are presented in Tab. 5. These results are visualized in Fig. 4. The results of our evaluation of hybrid post-quantum KEMs in TLS 1.3 are presented in Tab. 7. These results are visualized in Fig. 6.

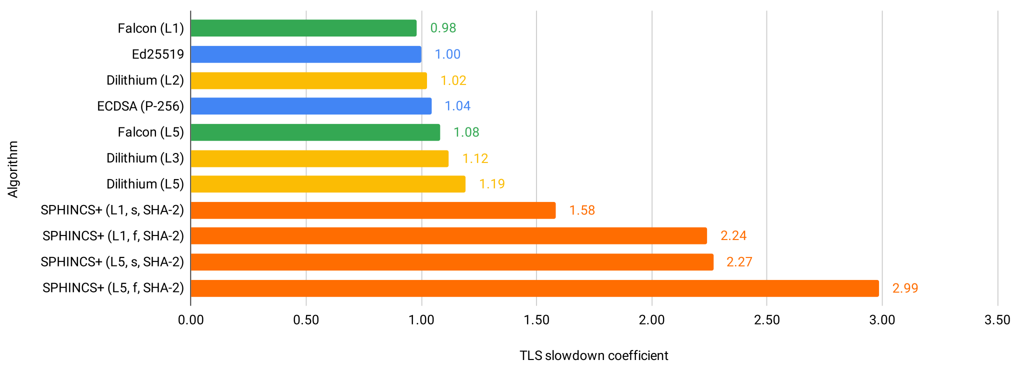


Figure 3: TLS slowdown coefficient for purely post-quantum signatures.

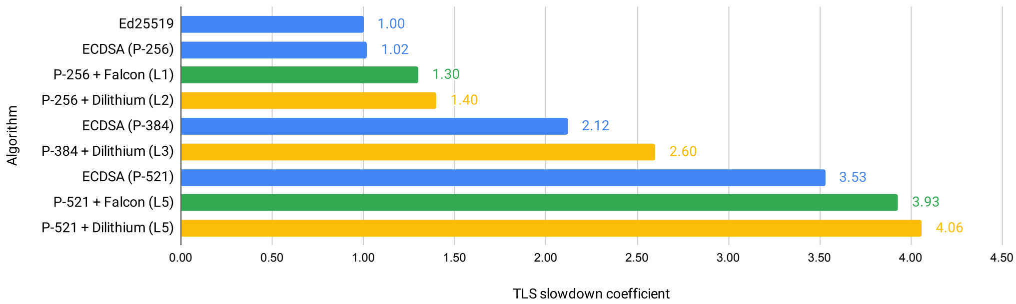


Figure 4: TLS slowdown coefficient for hybrid post-quantum signatures.

Table 4: Purely post-quantum signatures TLS performance

Algorithm (sec. level and params.)	**Connections/s** [mean ± st. dev.]	**TLS slowdown coefficient**
Falcon (L1)	892.63 ± 20.38	0.98
Ed25519 (control)	875.83 ± 17.64	1.00
Dilithium (L2)	854.54 ± 18.01	1.02
ECDSA (P-256)	839.76 ± 16.99	1.04
Falcon (L5)	809.83 ± 14.26	1.08
Dilithium (L3)	782.14 ± 17.19	1.12
Dilithium (L5)	734.33 ± 15.11	1.19
SPHINCS+ (L1, s, SHA-2)	553.04 ± 47.68	1.58
SPHINCS+ (L1, f, SHA-2)	390.84 ± 5.8	2.24
SPHINCS+ (L5, s, SHA-2)	390.29 ± 31.09	2.27
SPHINCS+ (L5, f, SHA-2)	296.22 ± 5.65	2.99

Table 5: Hybrid post-quantum signatures TLS performance

Algorithm (sec. level and params.)	**Connections/s** [mean ± st. dev.]	**TLS slowdown coefficient**
Ed25519	870.04 ± 23.45	1.00
ECDSA (P-256)	851.33 ± 17.85	1.02
P-256 + Falcon (L1)	669.76 ± 10.96	1.30
P-256 + Dilithium (L2)	620.09 ± 10.42	1.40
ECDSA (P-384)	410.24 ± 5.73	2.12
P-384 + Dilithium (L3)	334.10 ± 3.97	2.60
ECDSA (P-521)	246.24 ± 2.75	3.53
P-521 + Falcon (L5)	221.33 ± 3.30	3.93
P-521 + Dilithium (L5)	214.19 ± 2.30	4.06

P-256/P-384/P-521 refers to ECDSA using corresponding NIST curves.

Table 6: Purely post-quantum KEMs TLS performance

Algorithm (sec. level and params.)	**Connections/s** [mean ± st. dev.]	**TLS slowdown coefficient**
x25519 (control)	829.57 ± 15.48	1.00
Kyber (L1)	782.80 ± 14.03	1.06
Saber (L1)	776.79 ± 14.36	1.07
Kyber (L3)	760.13 ± 13.93	1.09
NTRU Prime (L1, LPR)	745.84 ± 15.26	1.11
NTRU Prime (L2, LPR)	746.60 ± 14.57	1.11
Saber (L3)	743.02 ± 14.89	1.12
Kyber (L5)	733.00 ± 15.24	1.13
NTRU (L1, HPS)	734.14 ± 13.45	1.13
Saber (L5)	733.30 ± 12.69	1.13
NTRU Prime (L3, LPR)	728.22 ± 14.31	1.14
NTRU Prime (L5, LPR)	720.67 ± 14.25	1.15
NTRU (L3, HPS)	700.24 ± 12.31	1.18
NTRU (L3, HRSS)	687.05 ± 13.80	1.21
HQC (L1)	676.69 ± 11.73	1.23
NTRU (L5, HPS)	661.54 ± 12.81	1.25
x448	582.92 ± 9.10	1.42
NTRU Prime (L1, S)	572.24 ± 10.83	1.45
HQC (L3)	553.22 ± 9.20	1.50
NTRU Prime (L2, S)	543.28 ± 8.47	1.53
NTRU Prime (L3, S)	488.73 ± 6.80	1.70
HQC (L5)	420.72 ± 7.95	1.97
NTRU Prime (L5, S)	347.78 ± 4.39	2.39
FrodoKEM (L1, AES)	328.00 ± 3.59	2.53
BIKE (L1)	325.66 ± 3.42	2.55
FrodoKEM (L3, AES)	222.30 ± 2.53	3.73
FrodoKEM (L5, AES)	141.68 ± 1.59	5.86
BIKE (L3)	133.63 ± 1.01	6.21
BIKE (L5)*	-	15.26

* BIKE security level 5 ciphersuite is not present in the older versions of liboqs [pro23b]. Therefore, we only present the slowdown coefficient which was computed from our other experiment which used the latest version (0.9.0).

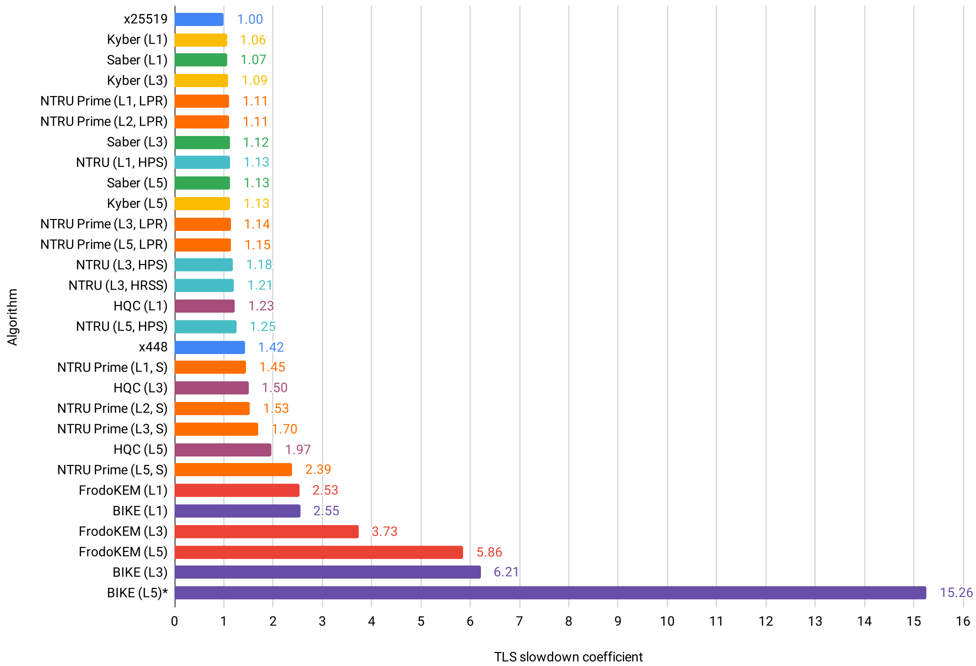


Figure 5: TLS slowdown coefficient for purely post-quantum KEMs.

Table 7: Hybrid post-quantum KEMs TLS performance

Algorithm (sec. level and params.)	**Connections/s** [mean ± st. dev.]	**TLS slowdown coefficient**
x25519	836.08 ± 13.27	1.00
P-256	735.71 ± 10.91	1.14
x25519 + Kyber (L1)	670.77 ± 9.18	1.25
x25519 + Kyber (L3)	653.46 ± 10.19	1.28
x25519 + HQC (L1)	647.19 ± 10.27	1.29
x448	571.88 ± 6.66	1.46
x448 + Kyber (L3)	423.81 ± 4.61	1.97
P-256 + Kyber (L1)	417.59 ± 4.59	2.00
P-256 + Kyber (L3)	411.93 ± 4.23	2.03
P-256 + HQC (L1)	392.19 ± 4.80	2.13
x448 + HQC (L3)	356.83 ± 3.23	2.34
x25519 + FrodoKEM (L1)	305.60 ± 2.68	2.74
x25519 + BIKE (L1)	300.34 ± 2.77	2.78
P-256 + FrodoKEM (L1)	233.12 ± 2.65	3.59
P-256 + BIKE (L1)	224.11 ± 2.38	3.73
x448 + FrodoKEM (L3)	178.15 ± 1.83	4.69
P-384	175.47 ± 1.72	4.76
P-384 + Kyber (L3)	143.59 ± 1.51	5.82
P-384 + HQC (L3)	130.47 ± 0.89	6.41
x448 + BIKE (L3)	101.95 ± 0.70	8.20
P-384 + FrodoKEM (L3)	100.40 ± 0.98	8.33
P-521	77.89 ± 0.44	10.73
P-521 + Kyber (L5)	72.55 ± 0.67	11.52
P-521 + HQC (L5)	69.41 ± 0.56	12.05
P-384 + BIKE (L3)	66.30 ± 0.34	12.61
P-521 + FrodoKEM (L5)	49.02 ± 0.57	17.06
P-521 + BIKE (L5)	30.27 ± 0.22	27.62

P-256/P-384/P-521 refers to ECDH using corresponding NIST curves.

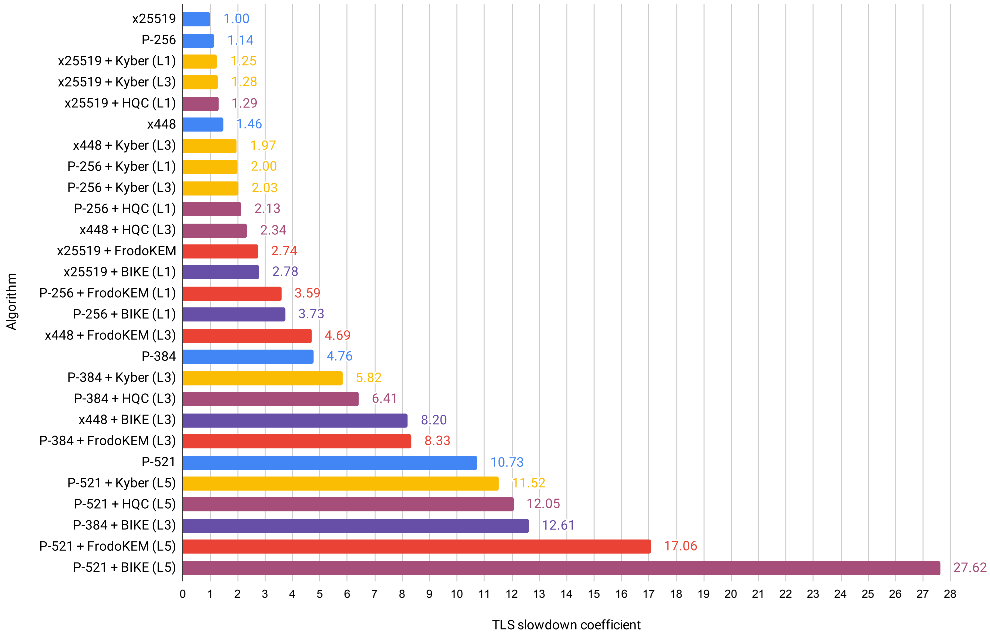


Figure 6: TLS slowdown coefficient for hybrid post-quantum KEMs.

B Symmetric primitives in post-quantum TLS

In this section, we discuss the quantum security of the symmetric ciphers and hash functions used in TLS 1.3. While all other publications on post-quantum TLS focused on the public-key cryptography in TLS, there has been no explicit evaluation of the quantum security of the symmetric ciphers and hash functions used in TLS 1.3.

B.1 Symmetric ciphers

TLS 1.3 [Res18] supports three symmetric ciphers: AES-128 (128-bit keys), AES-256 (256-bit keys), and ChaCha20 (256-bit keys).

Quantum security of symmetric key sizes. Grover's quantum search algorithm [Gro96] provides a quadratic speedup over classical exhaustive key search attack against symmetric ciphers such as AES, i.e., if a classical key search requires 2^{128} classical operations, a quantum key search requires $\sqrt{2^{128}} = 2^{64}$ *quantum* operations.

Although algorithms that can be broken in 2^{64} operations are usually considered insecure [SAB+18, Bar20, BSI23], notable works [NIS16, MST21, Wes22, BSI23] still consider AES-128 to be quantum-secure because it is difficult to parallelise Grover's algorithm [Zal99] and thus exploit its full theoretical power.

On the other hand, ETSI recommends the use of symmetric ciphers with 256-bit keys, since these primitives "will withstand quantum computer attacks until way after 2050. For these primitives, it is not necessary to look for alternatives with the same urgency as the search for quantum-safe asymmetric primitives." [ETS17]. The NSA also recommends the use of AES-256 [NSA22].

Conclusion. While symmetric ciphers with 128-bit keys are not as easily broken by Grover's algorithm as it might first appear, the conservative recommendation is to use 256-bit keys.

This means that all ciphers supported in TLS 1.3 can be considered quantum-secure, but it makes sense to use a symmetric cipher together with public key cryptographic primitives (KEMs, signatures) with similar security levels. Specifically, we recommend using AES-128 with post-quantum KEMs or signature schemes with NIST security level 1, and AES-256 or ChaCha20 for those primitives with NIST security level ≥ 2.

B.2 Hash functions

TLS 1.3 [Res18] supports two different hash functions: SHA-256 and SHA-384.

Quantum security of SHA-256 and SHA-384. The quantum security of hash functions is similar to the security of symmetric ciphers, since all known quantum attacks are based on Grover's algorithm [Gro96].

There are two types of attacks on hash functions that are improved by Grover's algorithm. First, while finding the pre-image of a hash function (with output size of n bits) using bruteforcing requires 2^n operations on a classical computer, Grover's quantum algorithm can reduce this complexity to $2^{\frac{n}{2}}$. Second, while finding a collision requires on average $2^{\frac{n}{2}}$ classical operations, the BHT quantum algorithm [BHT98] can reduce the complexity to $2^{\frac{n}{3}}$; however, the practical efficiency of the BHT algorithm is questionable since, according to [Ber09], the BHT algorithm has "fundamentally worse price-performance ratio" than classical collision search algorithms.

Conclusion. Both hash functions, SHA-256 and SHA-384, can be considered quantum-secure. Indeed, they correspond to the NIST security levels 2 and 4, respectively, [NIS16].