

Challenges and Strategies for Transitioning TLS to Post-Quantum Cryptography

Moritz Beckel, Holger Kinkelin*

**Chair of Network Architectures and Services*

School of Computation, Information and Technology, Technical University of Munich, Germany

Email: moritz.beckel@tum.de, kinkelin@net.in.tum.de

Abstract—With Harvest-Now-Decrypt-Later attacks, quantum computing already creates vulnerabilities in today’s systems and with increasing research on quantum computing, other attacks might soon become feasible against classical cryptography. Fortunately, post-quantum cryptography already exists, and it works with new mathematical problems that are not efficiently solvable by quantum computers. Transport Layer Security (TLS) is one of the most critical security protocols, so integrating post-quantum cryptography is imperative. In this work, we examine what parts of the TLS protocol are vulnerable to quantum-based attacks, and we evaluate different replacements in terms of standardization, security, and performance. Finally, we examine a strategy to efficiently migrate to a TLS implementation with post-quantum cryptography.

Index Terms—TLS, Post-quantum Cryptography

1. Introduction

TLS, responsible for encryption and authentication of endpoints, is one of the most used protocols on the internet, making security in this protocol especially important [1]. Unfortunately, this has not always been the case, as there have been several well-documented attacks over the last few years on the TLS protocol (e.g., the BEAST attack, which abuses issues of the TLS 1.0 implementation of Cipher Block Chaining) [2]. Even though mitigations for TLS vulnerabilities are proposed, there is still the problem of older TLS versions on the internet. Because of end-of-life software, many endpoints that do not support up-to-date implementations of TLS still exist, which is why it is so important to not only look at current vulnerabilities but also proactively mitigate future vulnerabilities [1].

One significant case is the capabilities of future quantum computers, which can break classical Public-Key Cryptography (PKC). A typical example for PKC is the RSA algorithm, which uses the integer factorization problem to make decryption without the key a hard problem [3]. However, in 1994, Peter Shor proposed an algorithm that can solve several mathematical problems in polynomial time, including the integer factorization problem [4]. Since the underlying mathematical problem of RSA is efficiently solvable, the cipher is no longer considered secure once quantum computers are powerful enough. To break an RSA key with 2048-bit length, a future quantum computer would need roughly 4000 qubits, though, since current quantum computers exhibit noise with increasing qubits, the real number is likely higher [5]. Current esti-

mates suggest the needed amount of qubits to be fewer than one million [6].

PKC is used in the TLS protocol to ensure the authenticity of a connection. Thus, a sufficiently powerful quantum computer can break authentication and even silently read alongside a TLS connection between two endpoints. What makes these kinds of attacks especially critical is that they already pose a threat to current TLS connections. With a Harvest-Now-Decrypt-Later attack, big institutions that have access to internet infrastructure can eavesdrop and store vast amounts of messages [7]. These encrypted messages can be recovered once a sufficiently powerful quantum computer is developed. Fortunately, several cryptographic ciphers are secure even with the use of a quantum computer. However, these so-called post-quantum cryptographic ciphers (PQC) have only recently been standardized and do not pose the historical strength that classical PKC has, as classical PKC has been well-studied and used for a long time [8].

We aim to explore the latest version of the TLS protocol (currently version 1.3) and analyze which operations can be broken using a quantum computer. We then evaluate the current state of PQC, the currently available algorithms, their performance, and the security impact on the TLS protocol. We examine the changes that need to be made to the protocol to efficiently support PQC and successfully mitigate attacks against adversaries with sufficient quantum computing power. Finally, we look at how to efficiently migrate to a TLS implementation with PQC and what is currently available.

Apart from TLS, there are many other areas where PQC must be integrated, which is why this topic has been heavily researched. Related research includes studies on the integration of PQC into several network protocols, such as DNSSEC [9] and IPSEC [10]. There are also approaches, such as POST [11], that aim to develop schemata to support PQC in the automotive industry.

2. Impact on the TLS protocol

TLS is a network protocol that works on top of the TCP protocol. Its main goal is to ensure confidentiality, authenticity, and integrity. We now examine the inner workings of TLS and single out those cryptographic operations/primitives that are not strong enough to withstand a post-quantum attack and need to be replaced by newer PQC. We will do so by observing a typical connection establishment of the TLS 1.3 protocol, which is defined in RFC 8446 [12]. As TLS is an extensive protocol with many extensions, we examine a connection where only

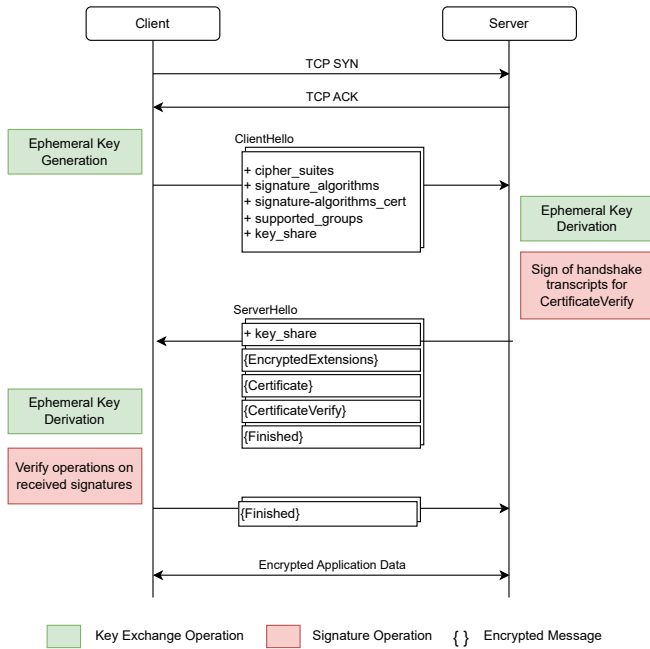


Figure 1: TLS 1.3 Handshake Overview.

the server authenticates itself, and the server and client do not have any pre-shared keys. A graphical overview is presented in Figure 1.

The TLS handshake starts once a TCP handshake is done and establishes a stateful connection. It begins by sending a *ClientHello* message, which contains several attribute fields. We omit most fields and only focus on the ones important for encryption. The first encryption-related field is the *cipher_suites* field, which contains a list of **symmetric ciphers** that the client supports. The next two fields are the *signature_algorithms* and *signature_algorithms_cert* fields that contain digital signature algorithms supported by the client. The final two fields are the *supported_groups* and *key_share* fields. These include all supported **key exchange algorithms** and the corresponding key material, which is used to share a secret key that both client and server can use for symmetric encryption. Upon receiving the message, the server selects a key exchange algorithm from the list of supported groups and calculates the shared secret.

The server responds with a *ServerHello* message that contains the *key_share* field with its own key material so that the client can deduce the shared secret as well. The message also contains the selected algorithm for symmetric encryption in the *cipher_suite* field. Afterward, the server sends the *EncryptedExtensions* message that contains additional extensions (e.g., the TCP segment size). Next, the server authenticates itself by sending the *Certificate* message that contains the server’s certificate chain and the *CertificateVerify* message, which shows that the previous messages were exchanged with the server. The server does so by calculating a **hash** over the handshake context and the certificate chain and then calculating a **digital signature** over the hash with the server’s private key. The client then verifies the digital signature with the server’s public key, and the client verifies the public key (contained in the server’s certificate) by going up the certificate chain and comparing the top with the client’s own

root certificate. The last two messages are the *Finished* messages, which both the client and the server send. They signal that neither server nor client has anything further to add to the handshake, and they contain a **Message Authentication Code (MAC)** of the entire handshake to ensure nothing was omitted. Once the *Finished* messages are exchanged, the TLS connection is established, and it is now possible to exchange user data. From now on, the shared secret is used for encryption, for both user data and status messages of the TLS protocol.

We can see that TLS makes use of several cryptographic operations: symmetric encryption, key exchange algorithms, digital signatures, hash functions, and MACs. As examined by Chen et al. [13], we can see that multiple operations are susceptible to quantum-capable attackers: Classical key exchange algorithms are based on the Diffie-Hellman algorithm, which uses the discrete logarithm problem for security [14]. However, with Shor’s Algorithm, quantum computers can efficiently solve this problem, making it insecure [4]. The same goes for digital signatures that typically use ECDSA and RSA algorithms, which are again based on the discrete logarithm problem or the integer factorization problem, that are weak against quantum computers [3], [4]. For symmetric encryption, the impact is not as drastic, as no efficient solution to the underlying problems has been found yet. Though Grover [15] discovered that it is possible to speed up general search algorithms quadratically with quantum computers, which means that we can also search faster for a symmetric key. However, if we double the key size, we can maintain the same security as before. The same applies to hash functions [7]. They are also considered secure, given that they have a larger output. Message authentication codes are based either on symmetric ciphers or hash functions, which means they are also considered as secure as symmetric encryption or hashing [16]. An overview of the different cryptographic operations and their security is shown in TABLE 1. Fortunately, we can see that vulnerable operations are primarily used in the handshake, as later everything is symmetrically encrypted with the shared secret, and they are limited to key exchange and digital signatures, which means we have to find alternatives for key exchange and digital signature algorithms.

Operation	Example Algorithm	Quantum-Resistance
Symmetric Cipher	AES128	Requires larger keys
Hash Function	SHA256	Requires larger output
MAC	HMAC	Requires larger output
Public Key Cipher	RSA-2048	Insecure
Digital Signature	RSA-2048	Insecure
Key Exchange	ECDHE	Insecure

TABLE 1: Resistance of Classical Cryptography to Quantum-Based Attacks [13]

3. Exchanging Keys with Post Quantum Cryptography

In the previous section, we showed how the TLS protocol works and what cryptographic operations it uses. We now look at PQC operations that replace classical key exchange and signature algorithms. In this section, we examine different key exchange algorithms regarding security and efficiency. We focus mainly on algorithms

that have been standardized, as they are the most evaluated and the most likely to be considered for implementation in libraries such as OpenSSL.

The National Institute of Standards and Technology (NIST) is the most well-known entity that tries to agree on common ciphers that will be later implemented in network protocols such as TLS. Starting in 2016, NIST held a competition to find suitable PQC ciphers. In several rounds [8], [17]–[19], ciphers were submitted and analyzed for weaknesses. Forty ciphers have been included, and a select few were standardized over the course of 6 years. NIST publishes them as a Federal Information Processing Standard (FIPS), which contains an exact definition of the used algorithms. Currently, three have been published [20]–[22] and two more have already been announced [23], [24]. NIST also specifies five security levels for the proposed ciphers [25]. The levels resemble the difficulty it takes to break the cipher in ascending order, and the difficulty is specified by comparing the cipher to symmetric cryptography, i.e., AES, and SHA. Starting with the first level, an adversary should require the same computational resources needed to break AES128 (SHA256, AES192, SHA384, AES256 with higher levels). Apart from NIST, other institutions are also trying to standardize PQC, mainly the International Standardization Organization (ISO), which includes other algorithms that NIST has not chosen [26].

At the moment, all institutions propose PQC suitable methods for exchanging keys. Instead of a Diffie-Hellman-based key exchange, all institutions agree on the use of the Key Encapsulation Mechanism (KEM): During a key exchange between a server and a client, the client first sends its public key. The Server then uses the public key to encapsulate (encrypt) a randomly-chosen secret and sends it back to the client. The client then decapsulates (decrypts) the key with its private key and gets hold of the shared secret. As KEM works very similarly to Diffie-Hellman, no changes in the TLS protocol flow need to be made in order to use it; however, KEM is not forward secure by default because it reuses the key pair. To achieve forward secrecy, the key pair needs to be regenerated for each connection [27].

We now examine ciphers that are being recommended by the institutions mentioned above. We compare security, performance, and power consumption. Security is defined by the underlying mathematical problem (e.g., integer factorization) and by the ciphertext indistinguishability property that formally verifies the security of a cryptosystem given that the previous mathematical problem holds. For ciphertext indistinguishability, NIST suggests Indistinguishability under adaptive chosen-ciphertext (IND-CCA2) [8]. Classifying the mathematical problems is not as simple because they inherently differ and require deep knowledge to do a proper comparison. Furthermore, there is no proof of hardness for these problems. They are only considered secure so long as no efficient solution has been found, which generally means the older a problem is, the more secure it is considered. For this reason, we only consider the recency of a used mathematical problem. For performance, we compare the execution time to Elliptic-Curve based Diffie-Hellman (ECDHE) in a TLS 1.3 handshake. We chose ECDHE as it is currently a supported algorithm in the TLS 1.3 standard and is widely used for

comparison in other work. We show the relative runtime overhead (e.g., 1.5x times higher execution time). Finally, we show the relative overhead for power consumption (e.g., 1.5x times higher power consumption):

Kyber (also called ML-KEM) was specified in FIPS 203 [22] and is currently the first choice for KEM ciphers. It is based on the Module-Learning-With-Error, a more recent specialization of a lattice problem, and currently has IND-CCA2 hardness [28]. Performance-wise, it is the best among lattice-based ciphers, and it is comparable to a classical key exchange with ECDHE for NIST level 1 when used in TLS 1.3, and for higher levels, Kyber can even outperform ECDHE [29]. Kyber also has similar power consumption in TLS 1.3 when compared to ECDHE [30].

HQC was chosen to be standardized as an alternative to Kyber by NIST because of the different underlying mathematical problem [24]. A standardization draft is expected to be released in 2026. It is based on the Quasi-Cyclic Syndrome Decoding problem, which is a specialization of a code-based problem and also has IND-CCA2 hardness [31]. Performance-wise, it is the best among code-based ciphers. On NIST level 1, HQC has a slight overhead of 1.45x - 1.8x when compared against ECDHE [7], [32]. HQC's power consumption is double that of ECDHE [30].

Additionally, NIST also lists other ciphers which still have not been disqualified but have not been chosen for standardization so far, including: Saber, NTRU and NTRU Prime, Classic McEliece, BIKE, and FrodoKEM [8], [19]. Most are not preferred either because of worse performance or stronger hardness assumptions of the ciphertext indistinguishability property. So far, they have been listed as alternative algorithms by NIST and are still being improved, so they remain an option if a cipher chosen for standardization is broken. A subset of alternative ciphers is also included in other standards:

FrodoKEM was specified in ISO 18033-2 [26] and is also recommended by BSI [33]. ISO considers it a conservative choice over Kyber as it uses the unmodified and well-studied Learning-With-Errors problem instead of the modified version used by Kyber [33], [34]. It has IND-CCA2 hardness and its performance overhead on NIST level 1 is 2.5x when compared with ECDHE [7]. FrodoKEM's Power consumption is roughly 14x times that of ECDHE [30].

Classic McEliece was also specified in ISO 18033-2 [26] and is recommended by BSI [33]. It is also considered a conservative choice as it is based on the Goppa-Code, which has the same maturity as RSA [33]. It has IND-CCA2 hardness [35]. The key size of Classic McEliece is significantly larger than that of other ciphers; in fact, it is too large to fit into the maximum size of $2^{16} - 1$ bytes of the TLS protocol's key share field, which means that it is not possible to use Classic McEliece in an unmodified TLS version [7]. There is work [36] that shows integrating Classic McEliece into the TLS protocol is possible, and a key share extension for the TLS protocol [37] has already been proposed. While there is currently no proper performance evaluation for the use in TLS, it was shown that the key generation of Classic McEliece is much slower than any other cipher mentioned here, and encapsulation and decapsulation are faster than HQC [38].

Now we have several choices as to what key exchange cipher we want to use. If we go with NIST's choices, we have two ciphers that perform similarly to old key exchange ciphers. However, their underlying mathematical problems are still relatively new and not as field-tested. On the other hand, ISO's choices rely on older, well-studied mathematical problems, which are less likely to be broken, but at the same time, they have a higher overhead. Fortunately, this overhead is manageable even on weaker hardware, e.g., it was shown by Wagner et al. [36] that it is possible to implement even Classic McEliece with its large key sizes on an embedded system. There is, however, a third option which offers a tradeoff between performance and security: Hybrid PQC.

Hybrid PQC makes use of both classical Diffie-Hellmann and PQC KEM ciphers to provide a resilient cipher, so long as quantum computing attacks are infeasible. Once post-quantum attacks are feasible, the PQC cipher will hopefully have matured enough for it to be considered as secure as the old cipher. The first proposal was made by Bos et al. [39], which concatenated the material of algorithms and put them in the same field. When a key is exchanged, both algorithms are run in parallel, and the result is concatenated again. This has the advantage that no changes to the TLS protocol need to be made, and it was shown that with Kyber and ECDHE, the performance is similar to a regular ECDHE key exchange [40]. There also exist other proposals, such as an Additional Shared Secret extension [41] in the TLS protocol to add a field for a second key or explicitly adding hybrid encryption fields to allow multiple cipher combinations [42]. This makes the separation explicit and would allow a TLS implementation to actively decide which cipher to use. The performance of these modifications has not been tested yet; however, the performance is likely similar to the concatenation approach because we execute the same algorithms. However, this can have an advantage if there are non-ideal network conditions. For connections with high packet loss, it was observed that some ciphers perform worse, and switching to other ciphers will result in better performance [40].

We now have several methods to implement a key exchange, which will securely generate a shared secret that we can use for symmetric encryption. However, adversaries still have the ability to spoof an active connection by issuing false certificates, which is why we also need to make use of post-quantum digital signatures.

4. Providing Certificates with Post Quantum Cryptography

Just like we did with key exchange algorithms, we examine different digital signature algorithms in terms of security and efficiency, and we will focus on ciphers that have been standardized. Functionally, post-quantum digital signatures behave exactly as classical signatures, which means they can act as a drop-in replacement in the TLS protocol. Again, we analyze security, efficiency, and power consumption, and we compare the signature algorithms to the Elliptic-Curve-based Digital Signature Algorithm (ECDSA) in a TLS 1.3 handshake. ECDSA is currently a supported algorithm in the TLS 1.3 standard and is widely used for comparison in other work. We

show the relative overhead of the execution time and the power consumption (e.g., 1.2x times higher execution time). For security of the cryptosystem, NIST suggests Existential Unforgeability under Chosen Message Attack (EUF-CMA) [25].

Dilithium (also called ML-DSA) was specified in FIPS 204 [20] and is currently the first choice for post-quantum digital signatures. It makes use of the Module-LWE and Module-SIS problems, which are based on the lattice problem [43]. Its hardness is SUF-CMA and meets the requirements of NIST [25], [43]. The performance of Dilithium is comparable to classical ECDSA, and its power consumption is also comparable [29], [30].

FALCON (also called FN-DSA) was chosen to be standardized by NIST as an alternative algorithm to Dilithium with similar performance [23]. A standardization draft is expected to be released in 2026. It makes use of the NTRU problem, which is based on the lattice problem [7], [44]. Falcon's hardness is proof of unforgeability in the QROM, which is deemed to be comparable to Dilithium [8], [45]. Performance-wise, it is also similar to ECDSA [7], [29], though the power consumption is roughly 2.7x times higher than with ECDSA [30].

SPHINCS+ (also called SLH-DSA) was specified in FIPS 205 [21] also as an alternative algorithm. It makes use of a hash-based problem, and its hardness is solely based on the security of hash functions, which pose adequate security [8]. The performance of SPHINCS+ is an order of magnitude slower than ECDSA; however, when comparing different works, there is no clear agreement on the slowdown [7]. Many different results have been published, likely because of different setups and the complexity of SPHINCS+, which has 12 parameters [7]. The same goes for energy consumption, which is much higher than other algorithms [30]. Because of this high overhead, SPHINCS+ is deemed a fallback in case a weakness is discovered in the other mainly lattice-based algorithms [8].

NIST and ISO also recommend the **XMSS/XMSS-MT** and **LMS/HSS** schemes [46], [47]. These are also hash-based and have better performance characteristics than SPHINCS+, but they are stateful, meaning that they can only sign a limited number of objects and require extra steps for the setup [7]. This makes them impractical for usage in the TLS protocol.

Similarly to key exchange algorithms, we can use hybrid algorithms and use classical signatures with post-quantum signatures simultaneously. This, however, requires changes to the TLS protocol and was implemented in [48]. The performance of hybrid signatures was shown to have an overhead of 1.2x to 2x times [49].

5. Implementing PQC into TLS

We now look at the TLS protocol again and examine what kind of attacks the TLS protocol is vulnerable to if we do not make use of PQC. We also look at how best to navigate the migration from classical cryptography to PQC.

Key exchange algorithms need to be migrated immediately because of the possibility of a Harvest-Now-Decrypt-Later attack, wherein an adversary that is on-path can eavesdrop and store messages, and once quantum

computers are powerful enough to break key exchange algorithms, the adversary can use them to recover the shared secret and decrypt all messages of the entire session. This attack requires an adversary to store a session from the beginning and acquire the key material from the TLS handshake, as this is the point where the TLS protocol is most vulnerable.

Authentication via certificates is less of a problem for current TLS connections: Adversaries can impersonate a peer by breaking its public keypair and issuing false certificates before the lifetime of the keypair ends. This attack also depends on the TLS handshake at the beginning of a TLS session. However, an adversary needs to actively participate in the connection, impersonate a peer, and only perform the attack while the session is ongoing, meaning it is not possible to perform this attack as long as we do not have a powerful enough quantum computer. Nevertheless, early adoption is still important as authentication with certificates is based on chaining with global root certificates on top of the chain having a life span of 20 years [50]. This means that if quantum-based attacks become feasible in this time span, root certificates that are trusted by and stored on the majority of devices need to be removed.

Because of these vulnerabilities, the current system needs to be migrated as soon as possible, which is why there are also recommendations by the IETF [50]. For key exchange algorithms, it is suggested to immediately transition to either hybrid or pure post-quantum key exchange, with hybrid being the preferred solution. From now on, TLS implementations should either provide both key shares or at least ask for post-quantum capabilities during a TLS handshake. Applications that depend on a library that implements TLS should test if their current setup supports post-quantum key exchange. For authentication, no immediate change is recommended. Instead, one should evaluate their system in terms of how often updates are possible and how accessible it is. If the system is deemed suitable, again, a hybrid solution is preferred over pure post-quantum certificates. To mitigate some of the overhead that hybrid certificates generate, new TLS extensions, such as TLS Cached Information Extension or TLS Certificate Compression, should be used. Once Quantum computing becomes feasible, hybrid signatures should be deprecated immediately because the current protocol design makes hybrid certificates vulnerable to downgrade attacks [51].

As an implementation, the most common library for TLS support is OpenSSL, which currently supports ML-KEM, ML-DSA, and SLH-DSA. With the provider architecture of OpenSSL and liboqs, it is also possible to access other algorithms such as FrodoKEM and HQC. A good alternative for embedded systems is WolfSSL. It currently also supports the same algorithms as OpenSSL. Nevertheless, it is important to carefully consider the security of implementations as they are not mature yet [50].

6. Conclusion and Future Work

We examined the TLS protocol and saw that key exchange and digital signature algorithms are weak and jeopardize security. Next, we analyzed alternatives for key

exchange and digital signature algorithms. For different algorithms and their underlying mathematical problems, it is mainly a tradeoff between security and performance, with mature problems having a higher overhead than newer problems. And finally, we showed guidelines on how to navigate the migration to PQC and what implementations to consider. Key exchange algorithms must be replaced immediately, while digital signature algorithms are less urgent. For now, hybrid algorithms are preferred over pure post-quantum algorithms, but classical public key cryptography must be deprecated once quantum-based attacks are feasible. Future work could encompass evaluating different implementations, such as OpenSSL and WolfSSL, in terms of formal verification and optimization for resource-constrained systems.

References

- [1] H. Lee, D. Kim, and Y. Kwon, "Tls 1.3 in practice: How tls 1.3 contributes to the internet," in *Proceedings of the Web Conference 2021*, 2021, pp. 70–79.
- [2] Y. Sheffer, R. Holz, and P. Saint-Andre, "Summarizing known attacks on transport layer security (tls) and datagram tls (dtls)," RFC 7457, 2015.
- [3] D. Mahto and D. K. Yadav, "Rsa and ecc: A comparative analysis," *International journal of applied engineering research*, vol. 12, no. 19, pp. 9053–9061, 2017.
- [4] P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," *SIAM review*, vol. 41, no. 2, pp. 303–332, 1999.
- [5] T. Häner, M. Roetteler, and K. M. Svore, "Factoring using $2n+2$ qubits with toffoli based modular multiplication," *arXiv preprint arXiv:1611.07995*, 2016.
- [6] C. Gidney, "How to factor 2048 bit RSA integers with less than a million noisy qubits," *arXiv preprint arXiv:2505.15917*, 2025.
- [7] N. Alnahawi, J. Müller, J. Oupický, and A. Wiesmaier, "A comprehensive survey on post-quantum tls," *IACR Communications in Cryptology*, 2024.
- [8] G. Alagic, D. Apon, D. Cooper, Q. Dang, T. Dang, J. Kelsey, J. Lichtinger, Y.-K. Liu, C. Miller *et al.*, "Status report on the third round of the nist post-quantum cryptography standardization process," 2022.
- [9] K. Shrishak and H. Shulman, "Negotiating pqc for dnssec," in *2021 51st Annual IEEE/IFIP International Conference on Dependable Systems and Networks-Supplemental Volume (DSN-S)*. IEEE, 2021, pp. 9–10.
- [10] S. Bae, Y. Chang, H. Park, M. Kim, and Y. Shin, "A performance evaluation of ipsec with post-quantum cryptography," in *International Conference on Information Security and Cryptology*, 2022.
- [11] H. Kinkel, F. Rezabek, M. Kempf, and G. Carle, "Post-quanten-sichere open-source schemata und technologien für automotive-anwendungen," 2024. [Online]. Available: <https://netintum.de/projects/post/>
- [12] E. Rescorla, "The transport layer security (tls) protocol version 1.3," RFC 8446, 2018.
- [13] L. Chen, S. Jordan, Y.-K. Liu, D. Moody, R. Peralta, R. A. Perlner, and D. Smith-Tone, *Report on post-quantum cryptography*. NIST, 2016, vol. 12.
- [14] U. M. Maurer and S. Wolf, "The diffie-hellman protocol," *Designs, Codes and Cryptography*, 2000.
- [15] L. K. Grover, "A fast quantum mechanical algorithm for database search," in *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, 1996, pp. 212–219.
- [16] R. Bhaumik, B. Chakraborty, W. Choi, A. Dutta, J. Govinden, and Y. Shen, "The committing security of macs with applications to generic composition," in *Annual International Cryptology Conference*, 2024.

- [17] D. Moody, G. Alagic, J. Alperin-Sheriff, D. Apon, D. Cooper, Q. Dang, Y.-K. Liu, C. Miller, R. Peralta, R. Perlner *et al.*, “Status report on the first round of the NIST post-quantum cryptography standardization process,” NIST, Tech. Rep., 2019.
- [18] G. Alagic, J. Alperin-Sheriff, D. Apon, D. Cooper, Q. Dang, J. Kelsey, Y.-K. Liu, C. Miller, D. Moody, R. Peralta *et al.*, “Status report on the second round of the NIST post-quantum cryptography standardization process,” *US Department of Commerce, NIST*, 2020.
- [19] G. Alagic, M. Bros, P. Ciadoux, D. Cooper, Q. Dang, T. Dang, J. Kelsey, J. Lichtinger, Y.-K. Liu, C. Miller *et al.*, *Status report on the fourth round of the NIST post-quantum cryptography standardization process*. US Department of Commerce, NIST, 2025.
- [20] NIST, “Module-lattice-based digital signature standard,” U.S. Department of Commerce, Washington, D.C., Tech. Rep. Federal Information Processing Standards Publications (FIPS) 204, 2024.
- [21] —, “Stateless hash-based digital signature standard,” U.S. Department of Commerce, Washington, D.C., Tech. Rep. Federal Information Processing Standards Publications (FIPS) 205, 2024.
- [22] —, “Module-lattice-based key-encapsulation mechanism standard,” U.S. Department of Commerce, Washington, D.C., Tech. Rep. Federal Information Processing Standards Publications (FIPS) 203, 2024.
- [23] C. Boutin. (2024) NIST releases first 3 Finalized Post-Quantum Encryption Standards. [Online]. Available: <https://www.nist.gov/news-events/news/2024/08/nist-releases-first-3-finalized-post-quantum-encryption-standards>
- [24] —. (2025) NIST selects HQC as fifth algorithm for Post-Quantum Encryption. [Online]. Available: <https://www.nist.gov/news-events/news/2025/03/nist-selects-hqc-fifth-algorithm-post-quantum-encryption>
- [25] NIST, “Submission requirements and evaluation criteria for the post-quantum cryptography standardization process,” 2016.
- [26] ISO/IEC 18033-2:2006, *ISO/IEC 18033-2:2006/DAmD 2 - Information technology — Security techniques — Encryption algorithms — Part 2: Asymmetric ciphers*. ISO, Geneva, Switzerland, 2025. [Online]. Available: <https://www.iso.org/standard/86890.html>
- [27] S. Fluhrer, Q. Dang, J. P. Mattsson, K. Milner, and D. Shiu, “ML-KEM Security Considerations,” Internet Engineering Task Force, Internet-Draft draft-sfluhrer-cfrg-ml-kem-security-considerations-03, May 2025, work in Progress. [Online]. Available: <https://datatracker.ietf.org/doc/draft-sfluhrer-cfrg-ml-kem-security-considerations/03/>
- [28] R. Avanzi, J. Bos, L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, J. M. Schanck, P. Schwabe, G. Seiler, D. Stehlé *et al.*, “Crystals-kyber algorithm specifications and supporting documentation,” *NIST PQC Round*, vol. 2, no. 4, pp. 1–43, 2019.
- [29] R. Döring and M. Geitz, “Post-quantum cryptography in use: Empirical analysis of the tls handshake performance,” in *NOMS 2022-2022 IEEE/IFIP Network Operations and Management Symposium*. IEEE, 2022, pp. 1–5.
- [30] G. Tasopoulos, C. Dimopoulos, A. P. Fournaris, R. K. Zhao, A. Sakzad, and R. Steinfeld, “Energy consumption evaluation of post-quantum tls 1.3 for resource-constrained embedded devices,” in *Proceedings of the 20th ACM International Conference on Computing Frontiers*, 2023, pp. 366–374.
- [31] P. Gaborit, C. Aguilar-Melchor, N. Aragon, S. Bettaiab, L. Bidoux, O. Blazy, J.-C. Deneuville, E. Persichetti, G. Zémor, J. Bos *et al.*, “Hamming Quasi-Cyclic (HQC),” 2025.
- [32] D. Sikeridis, P. Kampanakis, and M. 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*, 2020, pp. 149–156.
- [33] BSI TR-02102-1, *BSI TR-02102-1 Kryptographische Verfahren: Empfehlungen und Schlüssellängen*. BSI, Bonn, Germany, 2025. [Online]. Available: <https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Publikationen/TechnischeRichtlinien/TR02102/BSI-TR-02102.html>
- [34] “Frodokem: Learning with errors key encapsulation preliminary standardization proposal.” [Online]. Available: https://frodokem.org/files/FrodoKEM_standard_proposal_20241205.pdf
- [35] “Classic mceliece: conservative code-based cryptography: design rationale.” [Online]. Available: <https://classic.mceliece.org/mceliece-rationale-20221023.pdf>
- [36] J. Wagner and Y. Wang, New key share extension for classic mceliece algorithms. [Online]. Available: <https://www.ietf.org/archive/id/draft-wagner-tls-keysharepqc-03.html>
- [37] J. Roth, E. Karatsiolis, and J. Krämer, “Classic mceliece implementation with low memory footprint,” in *International Conference on Smart Card Research and Advanced Applications*, 2020.
- [38] O. Kuznetsov, S. Kandy, E. Frontoni, O. Smirnov *et al.*, “Trade-offs in post-quantum cryptography: A comparative assessment of bike, hqc, and classic mceliece,” in *CQPC*, 2023, pp. 1–11.
- [39] J. W. Bos, C. Costello, M. Naehrig, and D. Stebila, “Post-quantum key exchange for the tls protocol from the ring learning with errors problem,” in *2015 IEEE symposium on security and privacy*. IEEE, 2015, pp. 553–570.
- [40] C. Paquin, D. Stebila, and G. Tamvada, “Benchmarking post-quantum cryptography in TLS,” in *International Conference on Post-Quantum Cryptography*, 2020.
- [41] J. M. Schanck and D. Stebila, “A Transport Layer Security (TLS) Extension For Establishing An Additional Shared Secret,” Internet Engineering Task Force, Internet-Draft draft-schanck-tls-additional-keyshare-00, Apr. 2017, work in Progress. [Online]. Available: <https://datatracker.ietf.org/doc/draft-schanck-tls-additional-keyshare/00/>
- [42] W. Whyte, Z. Zhang, S. Fluhrer, and O. Garcia-Morchon, “Quantum-Safe Hybrid (QSH) Key Exchange for Transport Layer Security (TLS) version 1.3,” Internet Engineering Task Force, Internet-Draft draft-whyte-qsh-tls13-06, Oct. 2017, work in Progress. [Online]. Available: <https://datatracker.ietf.org/doc/draft-whyte-qsh-tls13/06/>
- [43] “Crystals-dilithium algorithm specifications and supporting documentation (version 3.1).” [Online]. Available: <https://pq-crystals.org/dilithium/data/dilithium-specification-round3-20210208.pdf>
- [44] P.-A. Fouque, J. Hoffstein, P. Kirchner, V. Lyubashevsky, T. Pornin, T. Prest, T. Ricosset, G. Seiler, W. Whyte, Z. Zhang *et al.*, “Falcon: Fast-fourier lattice-based compact signatures over ntru, specification v1. 2,” *NIST Post-Quantum Cryptography Standardization Round*, vol. 3, 2020.
- [45] D. Boneh, Ö. Dagdelen, M. Fischlin, A. Lehmann, C. Schaffner, and M. Zhandry, “Random oracles in a quantum world,” in *International conference on the theory and application of cryptology and information security*, 2011.
- [46] D. A. Cooper, D. C. Apon, Q. H. Dang, M. S. Davidson, M. J. Dworkin, C. A. Miller *et al.*, “Recommendation for stateful hash-based signature schemes,” *NIST Special Publication*, 2020.
- [47] ISO/IEC 14888-4:2024, *ISO/IEC 14888-4:2024 Information security — Digital signatures with appendix — Part 4: Stateful hash-based mechanisms*. ISO, Geneva, Switzerland, 2024. [Online]. Available: <https://www.iso.org/standard/80492.html>
- [48] E. Crockett, C. Paquin, and D. Stebila, “Prototyping post-quantum and hybrid key exchange and authentication in TLS and SSH,” Cryptology ePrint Archive, Paper 2019/858, 2019. [Online]. Available: <https://eprint.iacr.org/2019/858>
- [49] D. Marchsreiter and J. Sepúlveda, “Hybrid post-quantum enhanced tls 1.3 on embedded devices,” in *2022 25th Euromicro Conference on Digital System Design (DSD)*. IEEE, 2022, pp. 905–912.
- [50] T. Reddy.K and H. Tschofenig, “Post-Quantum Cryptography Recommendations for TLS-based Applications,” Internet Engineering Task Force, Internet-Draft draft-ietf-uta-pqc-app-00, Sep. 2025, work in Progress. [Online]. Available: <https://datatracker.ietf.org/doc/draft-ietf-uta-pqc-app/00/>
- [51] T. Reddy.K, T. Hollebeek, J. Gray, and S. Fluhrer, “Use of Composite ML-DSA in TLS 1.3,” Internet Engineering Task Force, Internet-Draft draft-reddy-tls-composite-mldsa-05, Jul. 2025, work in Progress. [Online]. Available: <https://datatracker.ietf.org/doc/draft-reddy-tls-composite-mldsa/05/>