PQUIP
Internet-Draft
Intended status: Informational
Expires: 2 January 2026

A. Banerjee
T. Reddy
D. Schoinianakis
Nokia
T. Hollebeek
DigiCert
M. Ounsworth
Entrust

1 July 2025

Post-Quantum Cryptography for Engineers
draft-ietf-pquip-pqc-engineers-13

Abstract

The advent of a cryptographically relevant quantum computer (CRQC) would render state-of-the-art, traditional public-key algorithms deployed today obsolete, as the mathematical assumptions underpinning their security would no longer hold. To address this, protocols and infrastructure must transition to post-quantum algorithms, which are designed to resist both traditional and quantum attacks. This document explains why engineers need to be aware of and understand post-quantum cryptography (PQC), detailing the impact of CRQCs on existing systems and the challenges involved in transitioning to post-quantum algorithms. Unlike previous cryptographic updates, this shift may require significant protocol redesign due to the unique properties of post-quantum algorithms.

About This Document

This note is to be removed before publishing as an RFC.

Status information for this document may be found at https://datatracker.ietf.org/doc/draft-ietf-pquip-pqc-engineers/.

Discussion of this document takes place on the pquip Working Group mailing list (mailto:pqc@ietf.org), which is archived at https://mailarchive.ietf.org/arch/browse/pqc/. Subscribe at https://www.ietf.org/mailman/listinfo/pqc/.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

This Internet-Draft will expire on 2 January 2026.

Copyright Notice

Table of Contents

1. Introduction

Quantum computing is no longer just a theoretical concept in computational science and physics; it is now an active area of research with practical implications. Considerable research efforts and enormous corporate and government funding for the development of practical quantum computing systems are currently being invested. At the time this document is published, cryptographically relevant quantum computers (CRQCs) that can break widely used public-key cryptographic algorithms are not yet available. However, there is

ongoing research and development in the field of quantum computing, with the goal of building more powerful and scalable quantum computers.

One common myth is that quantum computers are faster than conventional CPUs and GPUs in all areas. This is not the case; much as GPUs outperform general-purpose CPUs only on specific types of problems, so too will quantum computers have a niche set of problems on which they excel. Unfortunately for cryptographers, integer factorization and discrete logarithms, the mathematical problems underpinning much of classical public key cryptography, happen to fall within the niche that quantum computers are expected to excel at. As quantum technology advances, there is the potential for future quantum computers to have a significant impact on current cryptographic systems. Predicting the date of emergence of a CRQC is a challenging task, and there is ongoing uncertainty regarding when they will become practically feasible [CRQCThreat].

Extensive research has produced several post-quantum cryptographic algorithms that offer the potential to ensure cryptography’s survival in the quantum computing era. However, transitioning to a post-quantum infrastructure is not a straightforward task, and there are numerous challenges to overcome. It requires a combination of engineering efforts, proactive assessment and evaluation of available technologies, and a careful approach to product development.

PQC is sometimes referred to as "quantum-proof", "quantum-safe", or "quantum-resistant". It is the development of cryptographic algorithms designed to secure communication and data in a world where quantum computers are powerful enough to break traditional cryptographic systems, such as RSA and ECC. PQC algorithms are intended to be resistant to attacks by quantum computers, which use quantum-mechanical phenomena to solve mathematical problems that are infeasible for classical computers.

As the threat of CRQCs draws nearer, engineers responsible for designing, maintaining, and securing cryptographic systems must prepare for the significant changes that the existence of CRQCs will bring. Engineers need to understand how to implement post-quantum algorithms in applications, how to evaluate the trade-offs between security and performance, and how to ensure backward compatibility with current systems where needed. This is not merely a one-for-one replacement of algorithms; in many cases, the shift to PQC will involve redesigning protocols and infrastructure to accommodate the significant differences in resource utilization and key sizes between traditional and PQC algorithms.

This document aims to provide general guidance to engineers working on cryptographic libraries, network security, and infrastructure development, where long-term security planning is crucial. The document covers topics such as selecting appropriate PQC algorithms, understanding the differences between PQC key encapsulation mechanisms (KEMs) and traditional Diffie-Hellman and RSA style key exchanges, and provides insights into expected key, ciphertext and signature sizes and processing time differences between PQC and traditional algorithms. Additionally, it discusses the potential threat to symmetric cryptography and hash functions from CRQCs.

It is important to remember that asymmetric algorithms (also known as public key algorithms) are largely used for secure communications between organizations or endpoints that may not have previously interacted, so a significant amount of coordination between organizations, and within and between ecosystems needs to be taken into account. Such transitions are some of the most complicated in the tech industry and will require staged migrations in which upgraded agents need to co-exist and communicate with non-upgraded agents at a scale never before undertaken.

The National Security Agency (NSA) of the United States released an article on future PQC algorithm requirements for US national security systems [CNSA2-0] based on the need to protect against deployments of CRQCs in the future. The German Federal Office for Information Security (BSI) has also released a PQC migration and recommendations document [BSI-PQC] which largely aligns with United States National Institute of Standards and Technology (NIST) and NSA guidance, but differs on some of the guidance.

CRQCs pose a threat to both symmetric and asymmetric cryptographic schemes. However, the threat to asymmetric cryptography is significantly greater due to Shor's algorithm, which can break widely-used public key schemes like RSA and ECC. Symmetric cryptography and hash functions face a lower risk from Grover's algorithm, although the impact is less severe and can typically be mitigated by doubling key and digest lengths where the risk applies. It is crucial for the reader to understand that when the word "PQC" is mentioned in the document, it means asymmetric cryptography (or public key cryptography), and not any symmetric algorithms based on stream ciphers, block ciphers, hash functions, MACs, etc., which are less vulnerable to quantum computers. This document does not cover such topics as when traditional algorithms might become vulnerable (for that, see documents such as [QC-DNS] and others). It also does not cover unrelated technologies like quantum key distribution (QKD) or quantum key generation, which use quantum hardware to exploit quantum effects to protect communications and generate keys, respectively. PQC is based on conventional math (not on quantum mechanics) and software and can be run on any general purpose computer.

This document does not go into the deep mathematics or technical specification of the PQC algorithms, but rather provides an overview to engineers on the current threat landscape and the relevant algorithms designed to help prevent those threats. Also, the cryptographic and algorithmic guidance given in this document should be taken as non-authoritative if it conflicts with emerging and evolving guidance from the IRTF's Crypto Forum Research Group (CFRG).

There is ongoing discussion about whether to use the term "post-quantum", "quantum ready", or "quantum resistant", to describe algorithms that resist CRQCs, and a consensus has not yet been reached. It is important to clarify that "post-quantum" refers to algorithms designed to withstand attacks by CRQCs and classical computers alike. These algorithms are based on mathematically hard cryptographic problems that neither CRQCs nor classical computers are expected to break. This document uses any of these terms interchangeably to refer to such algorithms.

2. Terminology

Quantum computer: A computer that performs computations using quantum-mechanical phenomena such as superposition and entanglement.

Physical qubit: The basic physical unit in a quantum computer, which is prone to noise and errors.

Logical qubit: A fault-tolerant qubit constructed from multiple physical qubits using quantum error correction; it is the effective unit for reliable quantum computation.

Post-Quantum Cryptography (PQC): Cryptographic algorithms designed to be secure against quantum and classical attacks.

Cryptographically Relevant Quantum Computer (CRQC): A quantum computer with sufficient "logical qubits" to perform cryptographic attacks (e.g., break RSA/ECC).

3. Threat of CRQCs on Cryptography

When considering the security risks associated with the ability of a quantum computer to attack traditional cryptography, it is important to distinguish between the impact on symmetric algorithms and public-key ones. Dr. Peter Shor and Dr. Lov Grover developed two algorithms that changed the way the world thinks of security under the presence of a CRQC.

Quantum computers are, by their nature, hybrids of classical and quantum computational units. For example, Shor’s algorithm consists of a combination of quantum and classical computational steps. Thus, the term "quantum adversary" should be thought of as "quantum-enhanced adversary", meaning they have access to both classical and quantum computational techniques.

Despite the fact that large-scale quantum computers do not yet exist to experiment on, the theoretical properties of quantum computation are very well understood. This allows us to reason today about the upper limits of quantum-enhanced computation, and indeed to design cryptographic algorithms that are resistant to any conceivable form of quantum cryptanalysis.

3.1. Symmetric Cryptography

For unstructured data such as symmetric encrypted data or cryptographic hashes, although CRQCs can search for specific solutions across all possible input combinations (e.g., Grover’s algorithm), no quantum algorithm is known to break the underlying security properties of these classes of algorithms.

Grover’s algorithm is a quantum search algorithm that provides a theoretical quadratic speedup for searching an unstructured database, compared to traditional search algorithms. This has led to the common misconception that symmetric key lengths need to be doubled for quantum security. When you consider the mapping of hash values to their corresponding hash inputs (also known as pre-image), or of

ciphertext blocks to the corresponding plaintext blocks, as an unstructured database, then Grover's algorithm theoretically requires doubling the key sizes of the symmetric algorithms that are currently deployed today to counter the quadratic speedup and maintain current security level. This is because Grover's algorithm reduces the amount of operations to break 128-bit symmetric cryptography to 2^{64} quantum operations, which might sound computationally feasible. However, quantum operations are fundamentally different from classical ones as 2^{64} classical operations can be efficiently parallelized, 2^{64} quantum operations must be performed serially, making them infeasible on practical quantum computers.

Grover's algorithm is highly non-parallelizable and even if one deploys 2^c computational units in parallel to brute-force a key using Grover's algorithm, it will complete in time proportional to 2^{(128−c)/2}, or, put simply, using 256 quantum computers will only reduce runtime by a factor of 16, 1024 quantum computers will only reduce runtime by a factor of 32 and so forth (see [NIST] and [Cloudflare]). Due to this inherent limitation, the general expert consensus is that AES-128 remains secure in practice, and key sizes do not necessarily need to be doubled.

It would be natural to ask whether future research will develop a superior algorithm that could outperform Grover's algorithm in the general case. However, Christof Zalka has shown that Grover's algorithm achieves the best possible complexity for this type of search, meaning no significantly faster quantum approach is expected [Grover-search]

Finally, in their evaluation criteria for PQC, NIST is assessing the security levels of proposed post-quantum algorithms by comparing them against the equivalent traditional and quantum security of AES-128, 192, and 256. This indicates that NIST is confident in the stable security properties of AES, even in the presence of both traditional and quantum attacks. As a result, 128-bit algorithms can be considered quantum-safe for the foreseeable future. However, for compliance purposes, some organizations, such as the National Agency for the Security of Information Systems (ANSSI), recommend the use of AES-256 [ANSSI].

3.2. Asymmetric Cryptography

"Shor's algorithm" efficiently solves the integer factorization problem (and the related discrete logarithm problem), which underpin the foundations of the vast majority of public-key cryptography that the world uses today. This implies that, if a CRQC is developed, today's public-key cryptography algorithms (e.g., RSA, Diffie-Hellman and elliptic curve cryptography, as well as less commonly-used

variants such as ElGamal and Schnorr signatures) and protocols would need to be replaced by algorithms and protocols that can offer cryptanalytic resistance against CRQCs. Note that Shor's algorithm cannot run solely on a classical computer, it requires a CRQC.

For example, to provide some context, one would need around 20 million noisy qubits to break RSA-2048 in 8 hours [RSAShor] and [RSA8HRS] or 4099 stable (or logical) qubits to break it in 10 seconds [RSA10SC].

For structured data such as public keys and signatures, CRQCs can fully solve the underlying hard problems used in traditional cryptography (see Shor's algorithm). Because an increase in the size of the key-pair would not provide a secure solution (short of RSA keys that are many gigabytes in size [PQRSA]), a complete replacement of the algorithm is needed. Therefore, post-quantum public-key cryptography must rely on problems that are different from the ones used in traditional public-key cryptography (i.e., the integer factorization problem, the finite-field discrete logarithm problem, and the elliptic-curve discrete logarithm problem).

3.3. Quantum Side-channel Attacks

The field of cryptographic side-channel attacks potentially stands to gain a boost in attacker power once cryptanalytic techniques can be enhanced with quantum computation techniques [QuantSide]. While a full discussion of quantum side-channel techniques is beyond the scope of this document, implementers of cryptographic hardware should be aware that current best-practices for side-channel resistance may not be sufficient against quantum adversaries.

4. Traditional Cryptographic Primitives that Could Be Replaced by PQC

Any asymmetric cryptographic algorithm based on integer factorization, finite field discrete logarithms or elliptic curve discrete logarithms will be vulnerable to attacks using Shor's algorithm on a CRQC. This document focuses on the principal functions of asymmetric cryptography:

* Key agreement and key transport: Key agreement schemes, typically referred to as Diffie-Hellman (DH) or Elliptic Curve Diffie-Hellman (ECDH), as well as key transport, typically using RSA encryption, are used to establish a shared cryptographic key for secure communication. They are one of the mechanisms that can be replaced by PQC, as they are based on existing public key cryptography and is therefore vulnerable to Shor's algorithm. A CRQC can employ Shor's algorithm to efficiently find the prime factors of a large public key (in the case of RSA), which in turn

can be exploited to derive the private key. In the case of Diffie-Hellman, a CRQC has the potential to calculate the discrete logarithm of the (short or long-term) Diffie-Hellman public key. This, in turn, would reveal the secret required to derive the symmetric encryption key.

* Digital signatures: Digital signature schemes are used to authenticate the identity of a sender, detect unauthorized modifications to data, and underpin trust in a system. Similar to key agreement, signatures also depend on a public-private key pair based on the same mathematics as for key agreement and key transport, and hence a break in existing public key cryptography will also affect traditional digital signatures, hence the importance of developing post-quantum digital signatures.

* BBS signatures: BBS (Boneh-Boyen-Shacham) signatures are a privacy-preserving signature scheme that offers zero-knowledge proof-like properties by allowing selective disclosure of specific signed attributes without revealing the entire set of signed data. The security of BBS signatures relies on the hardness of the discrete logarithm problem, making them vulnerable to Shor's algorithm. A CRQC can break the data authenticity security property of BBS but not the data confidentiality (Section 6.9 of [I-D.irtf-cfrg-bbs-signatures]).

* Content encryption: Content encryption typically refers to the encryption of the data using symmetric key algorithms, such as AES, to ensure confidentiality. The threat to symmetric cryptography is discussed in Section 3.1.

5. Invariants of PQC: Necessitating Compliance Adjustments

In the context of PQC, symmetric-key cryptographic algorithms are generally not directly impacted by quantum computing advancements. Symmetric-key cryptography, which includes keyed primitives such as block ciphers (e.g., AES) and message authentication mechanisms (e.g., HMAC-SHA256), relies on secret keys shared between the sender and receiver and remains secure even in a post-quantum world. Symmetric cryptography also includes hash functions (e.g., SHA-256) that are used for secure message digesting without any shared key material. HMAC is a specific construction that utilizes a cryptographic hash function and a secret key shared between the sender and receiver to produce a message authentication code.

Grover's algorithm does not pose a practical threat to symmetric cryptography (see Section 3.1 for more details). As a result, CRQCs offer no substantial advantages in breaking symmetric-key algorithms compared to classical computers. However, for compliance purposes,

such as meeting the standards of CNSA 2.0 (Commercial National Security Algorithm Suite 2.0) [CNSA2-0], AES-256 must be used to ensure the highest level of security against both traditional and quantum threats.

6. NIST PQC Algorithms

At time of writing, NIST have standardized three PQC algorithms, with more expected to be standardised in the future ([NISTFINAL]). These algorithms are not necessarily drop-in replacements for traditional asymmetric cryptographic algorithms. For instance, RSA [RSA] and ECC [RFC6090] can be used as both a key encapsulation method (KEM) and as a signature scheme, whereas there is currently no post-quantum algorithm that can perform both functions. When upgrading protocols, it is important to replace the existing use of traditional algorithms with either a PQC KEM or a PQC signature method, depending on how the traditional algorithm was previously being used. Additionally, KEMs, as described in Section 10, present a different API than either key agreement or key transport primitives. As a result, they may require protocol-level or application-level changes in order to be incorporated.

6.1. NIST Candidates Selected for Standardization

6.1.1. PQC Key Encapsulation Mechanisms (KEMs)

* [ML-KEM]: Module-Lattice-based Key-Encapsulation Mechanism Standard (FIPS-203).

* [HQC]: This algorithm is based on the hardness of the syndrome decoding problem for quasi-cyclic concatenated Reed-Muller and Reed-Solomon (RMRS) codes in the Hamming metric. Reed-Muller (RM) codes are a class of block error-correcting codes commonly used in wireless and deep-space communications, while Reed-Solomon (RS) codes are widely used to detect and correct multiple-bit errors. HQC has been selected as part of the NIST post-quantum cryptography project but has not yet been standardized.

6.1.2. PQC Signatures

* [ML-DSA]: Module-Lattice-Based Digital Signature Standard (FIPS-204).

* [SLH-DSA]: Stateless Hash-Based Digital Signature (FIPS-205).

* [FN-DSA]: FN-DSA is a lattice signature scheme (FIPS-206) (Section 9.1 and Section 11.3).

7. ISO Candidates Selected for Standardization

At the time of writing, ISO has standardized three PQC KEM algorithms, which are mentioned in the following subsection.

7.1. PQC Key Encapsulation Mechanisms (KEMs)

* [FrodoKEM]: Key Encapsulation mechanism based on the hardness of learning with errors in algebraically unstructured lattices.

* [ClassicMcEliece]: Based on the hardness of syndrome decoding of Goppa codes. Goppa codes are a class of error-correcting codes that can correct a certain number of errors in a transmitted message. The decoding problem involves recovering the original message from the received noisy codeword.

* [NTRU]: Key encapsulation mechanism based on the "N-th degree Truncated polynomial Ring Units" (NTRU) lattices.

8. Timeline for Transition

The timeline, and driving motivation for transition differs slightly between data confidentiality (e.g., encryption) and data authentication (e.g., signature) use-cases.

For data confidentiality, one is concerned with the so-called "harvest now, decrypt later" attack where a malicious actor with adequate resources can launch an attack to store sensitive encrypted data today that they hope to decrypt once a CRQC is available. This implies that, every day, sensitive encrypted data is susceptible to the attack by not implementing quantum-safe strategies, as it corresponds to data possibly being deciphered in the future.

For authentication, it is often the case that signatures have a very short lifetime between signing and verifying (such as during a TLS handshake) but some authentication use-cases do require long lifetimes, such as signing firmware or software that will be active for decades, signing legal documents, or signing certificates that will be embedded into hardware devices such as smartcards. Even for short-lived signatures use cases, the infrastructure often relies on long-lived root keys which can be difficult to update or replace on in-field devices.

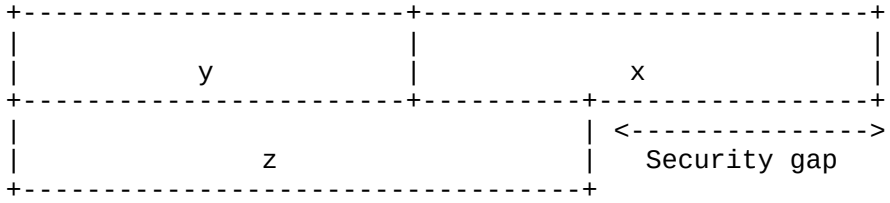

Figure 1: Mosca model

These challenges are illustrated nicely by the so-called Mosca model discussed in [Threat-Report]. In Figure 1, "x" denotes the time that systems and data need to remain secure, "y" the number of years to fully migrate to a PQC infrastructure, and "z" the time until a CRQC that can break current cryptography is available. The model assumes either that encrypted data can be intercepted and stored before the migration is completed in "y" years, or that signatures will still be relied upon for "x" years after their creation. This data remains vulnerable for the complete "x" years of their lifetime, thus the sum "x+y" gives us an estimate of the full timeframe that data remain insecure. The model essentially asks how one is preparing IT systems during those "y" years (in other words, how one can minimize those "y" years) to minimize the transition phase to a PQC infrastructure and hence minimize the risks of data being exposed in the future.

Finally, other factors that could accelerate the introduction of a CRQC should not be under-estimated, like for example faster-than-expected advances in quantum computing and more efficient versions of Shor's algorithm requiring fewer qubits. Innovation often comes in waves, so it is to the industry's benefit to remain vigilant and prepare as early as possible. Bear in mind also that while the industry tracks advances from public research institutions such as universities and companies that publish their results, there is also a great deal of large-budget quantum research being conducted privately by various national interests. Therefore, the true state of quantum computer advancement is likely several years ahead of the publicly available research.

Organizations should also consider carefully and honestly what their migration timeline "y" actually is. If you think only of the time between receiving a patch from your technology vendor, and rolling that patch out, then "y" might seem as short as a few weeks. However, this represents the minority of migration cases; more often, a PQC migration will involve at least some amount of hardware replacement. For example, performance-sensitive applications will need CPUs with PQC hardware acceleration. Security-sensitive applications will need PQC TPMs, TEEs, Secure Enclaves, and other cryptographic co-processors. Smartcard applications will require

replacement of the cards as well as of the readers which can come in many form-factors: tap-for-entry door and turnstile readers, PIN pad machines, laptops with built-in smartcard readers, and many others.

Included in "y" is not only the deployment time, but also preparation time: integration, testing, auditing, and re-certification of cryptographic environments. Consider also upstream effects that contribute to "y", including lead-times for your vendors to produce PQC-ready products, which may itself include auditing and certification delays, time for regulating bodies to adopt PQC policies, time for auditors to become familiar with the new requirements, etc. If you measure the full migration time "y" from when your vendors begin implementing PQC functionality, to when you switch off your last non-PQC-capable device, then "y" can be quite long; likely measured in years for even most moderately-sized organizations, this long tail should not discourage early action.

Organizations responsible for protecting long-lived sensitive data or operating critical infrastructure will need to begin transitioning immediately, particularly in scenarios where data is vulnerable to HDNL attacks. PQ/T or PQ key exchange is relatively self-contained, typically requiring changes only to the cryptographic library (e.g., OpenSSL). In contrast, migrating to post-quantum or PQ/T digital signatures involves broader ecosystem changes, including updates to certificates, CAs, Certificate Management Protocols, HSMs, and trust anchors. Starting early with hybrid key exchange deployments allows organizations to gain operational experience, while prototyping and planning for PQ/T or PQ digital signature integration helps identify ecosystem-wide impacts early. This phased approach reduces long-term migration risks and ensures readiness for more complex updates.

9. PQC Categories

The post-quantum cryptographic schemes standardized by NIST, along with the ongoing Round 4 candidates, can be categorized into three main groups: lattice-based, hash-based, and code-based. Other approaches, such as isogeny-based, multivariate-based, and MPC-in-the-Head-based cryptography, are also being explored in research and standardization efforts. NIST has been calling for additional digital signature proposals to be considered in the PQC standardization process which has completed two rounds in October, 2024 [AddSig].

9.1. Lattice-Based Public-Key Cryptography

Lattice-based public-key cryptography leverages the simple construction of lattices (i.e., a regular collection of points in a Euclidean space that are evenly spaced) to create "trapdoor" problems. These problems are efficient to compute if you possess the secret information but challenging to compute otherwise. Examples of such problems include the shortest vector, closest vector, short integer solution, learning with errors, module learning with errors, and learning with rounding problems. All of these problems feature strong proofs for worst-to-average case reduction, effectively relating the hardness of the average case to the worst case.

Lattice-based schemes usually have good performances and average size public keys and signatures (average within the PQC primitives at least; they are still several orders of magnitude larger than e.g., RSA or ECC signatures), making them the best available candidates for general-purpose use such as replacing the use of RSA in PKIX certificates.

Examples of this class of algorithms include ML-KEM, FN-DSA, ML-DSA and FrodoKEM.

It is noteworthy that lattice-based encryption schemes require a rounding step during decryption which has a non-zero probability of "rounding the wrong way" and leading to a decryption failure, meaning that valid encryptions are decrypted incorrectly; as such, an attacker could significantly reduce the security of lattice-based schemes that have a relatively high failure rate. However, for most of the NIST PQC proposals, the number of required oracle queries to force a decryption failure is above practical limits, as has been shown in [LattFail1]. More recent works have improved upon the results in [LattFail1], showing that the cost of searching for additional failing ciphertexts after one or more have already been found, can be sped up dramatically [LattFail2]. Nevertheless, at the time this document is published, the PQC candidates by NIST are considered secure under these attacks and constant monitoring as cryptanalysis research is ongoing.

9.2. Hash-Based Public-Key Cryptography

Hash based PKC has been around since the 1970s, when it was developed by Lamport and Merkle. It is used to create digital signature algorithms and its security is based on the security of the underlying cryptographic hash function. Many variants of hash-based signatures (HBS) have been developed since the 70s including the recent XMSS [RFC8391], HSS/LMS [RFC8554] or BPQS [BPQS] schemes. Unlike many other digital signature techniques, most hash-based

signature schemes are stateful, which means that signing necessitates the update and careful tracking of the state of the secret key. Producing multiple signatures using the same secret key state results in loss of security and may ultimately enable signature forgery attacks against that key.

Stateful hash-based signatures with long service lifetimes require additional operational complexity compared with other signature types. For example, consider a 20-year root key; there is an expectation that 20 years is longer than the expected lifetime of the hardware that key is stored on, and therefore the key will need to be migrated to new hardware at some point. Disaster-recovery scenarios where the primary node fails without warning can be similarly tricky. This requires careful operational and compliance consideration to ensure that no private key state can be reused across the migration or disaster recovery event. One approach for avoiding these issues is to only use stateful HBS for short-term use cases that do not require horizontal scaling, for example signing a batch of firmware images and then retiring the signing key.

The SLH-DSA algorithm, which was standardized by NIST, leverages the HORST (hash to obtain random subset with trees) technique and remains the only standardized hash based signature scheme that is stateless, thus avoiding the complexities associated with state management. SLH-DSA is an advancement on SPHINCS which reduces the signature sizes in SPHINCS and makes it more compact.

9.3. Code-Based Public-Key Cryptography

This area of cryptography started in the 1970s and 80s based on the seminal work of McEliece and Niederreiter which focuses on the study of cryptosystems based on error-correcting codes. Some popular error correcting codes include Goppa codes (used in McEliece cryptosystems), encoding and decoding syndrome codes used in Hamming quasi-cyclic (HQC), or quasi-cyclic moderate density parity check (QC-MDPC) codes.

Examples include all the unbroken NIST Round 4 finalists: Classic McEliece, HQC (selected by NIST for standardization), and BIKE.

10. KEMs

A Key Encapsulation Mechanism (KEM) is a cryptographic technique used for securely exchanging symmetric key material between two parties over an insecure channel. It is commonly used in hybrid encryption schemes, where a combination of asymmetric (public key) and symmetric encryption is employed. The KEM encapsulation results in a fixed-length symmetric key that can be used with a symmetric algorithm,

typically a block cipher, in one of two different ways:

* Derive a data encryption key (DEK) to encrypt the data

* Derive a key encryption key (KEK) used to wrap a DEK

These techniques are often referred to as "hybrid public key encryption (HPKE)" [RFC9180] mechanism.

The term "encapsulation" is chosen intentionally to indicate that KEM algorithms behave differently at the API level from the key agreement or key encipherment / key transport mechanisms that are in use today. Key agreement schemes imply that both parties contribute a public / private key pair to the exchange, while key encipherment / key transport schemes imply that the symmetric key material is chosen by one party and "encrypted" or "wrapped" for the other party. KEMs, on the other hand, behave according to the following API primitives [PQCAPI]:

* def kemKeyGen() -> (pk, sk)

* def kemEncaps(pk) -> (ss, ct)

* def kemDecaps(ct, sk) -> ss

where pk is the public key, sk is the secret key, ct is the ciphertext representing an encapsulated key, and ss is the shared secret. The following figure illustrates a sample flow of a KEM-based key exchange:

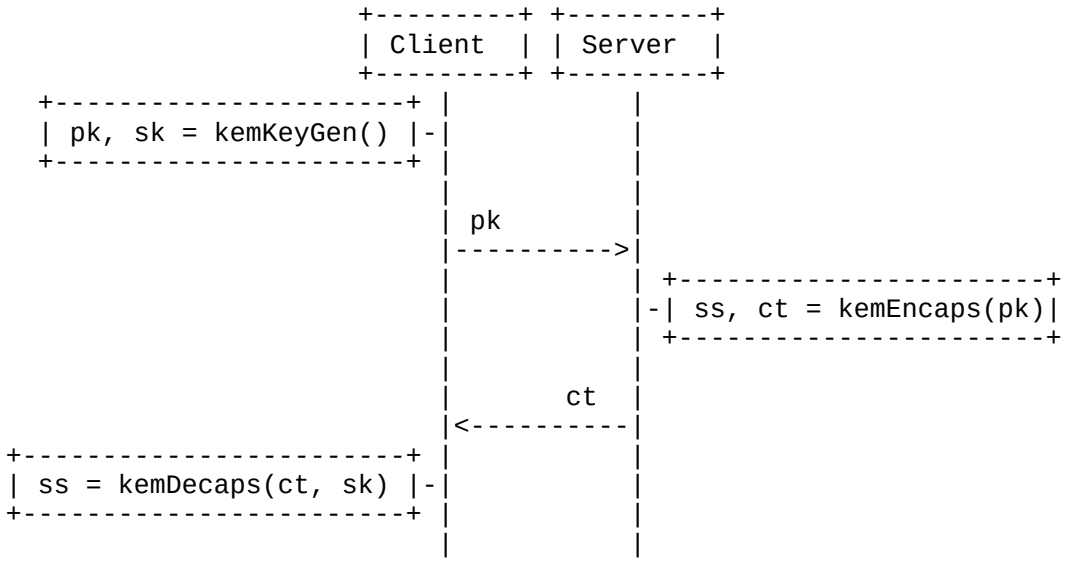

Figure 2: KEM based key exchange

10.1. Authenticated Key Exchange

Authenticated Key Exchange (AKE) with KEMs where both parties contribute a KEM public key to the overall session key is interactive as described in [I-D.draft-ietf-lake-edhoc]. However, single-sided KEM, such as when one peer has a KEM key in a certificate and the other peer wants to encrypt for it (as in S/MIME or OpenPGP email), can be achieved using non-interactive HPKE [RFC9180]. The following figure illustrates the Diffie-Hellman (DH) Key exchange:

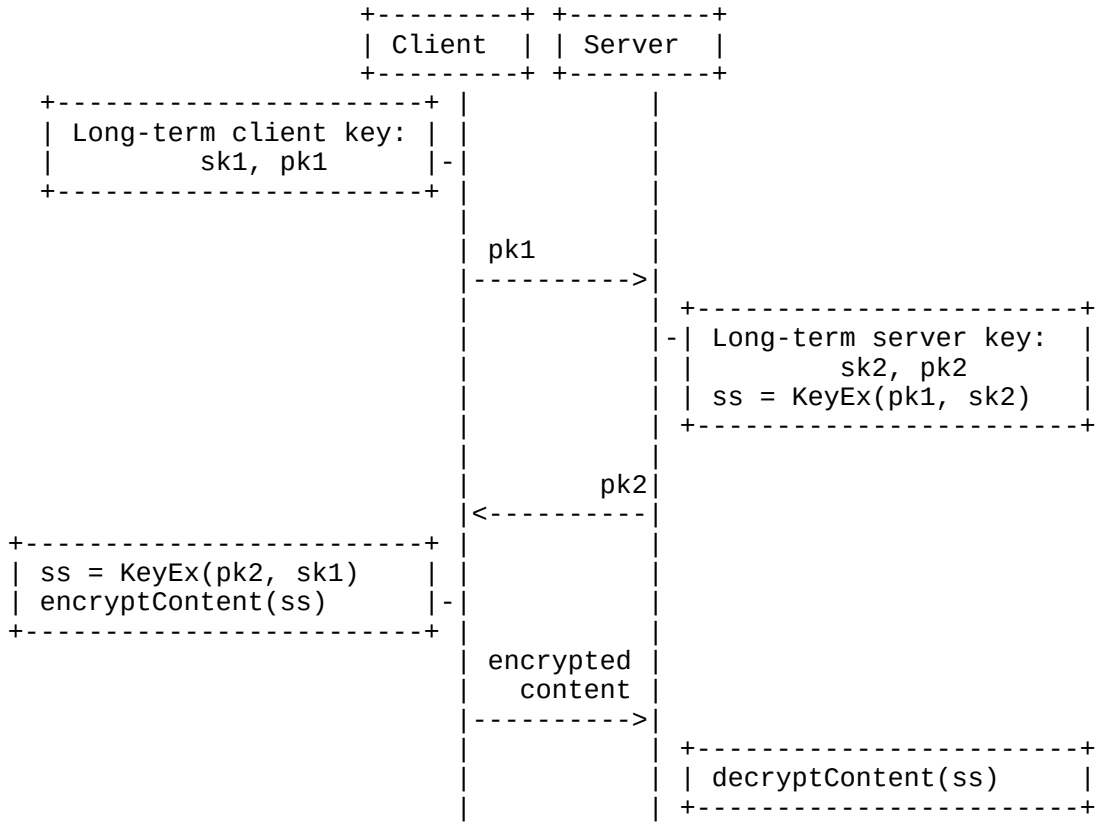

Figure 3: Diffie-Hellman based AKE

What's important to note about the sample flow above is that the shared secret ss is derived using key material from both the Client and the Server, which classifies it as an AKE. There is another property of a key exchange, called Non-Interactive Key Exchange (NIKE) which refers to whether the sender can compute the shared secret ss and encrypt content without requiring active interaction (an exchange of network messages) with the recipient. Figure 3 shows a Diffie-Hellman key exchange which is an AKE, since both parties are using long-term keys which can have established trust (for example, via certificates), but it is not a NIKE, since the client needs to wait for the network interaction to receive the receiver's public key pk2 before it can compute the shared secret ss and begin content encryption. However, a DH key exchange can be an AKE and a NIKE at the same time if the receiver's public key is known to the sender in advance, and many Internet protocols rely on this property of DH-based key exchanges.

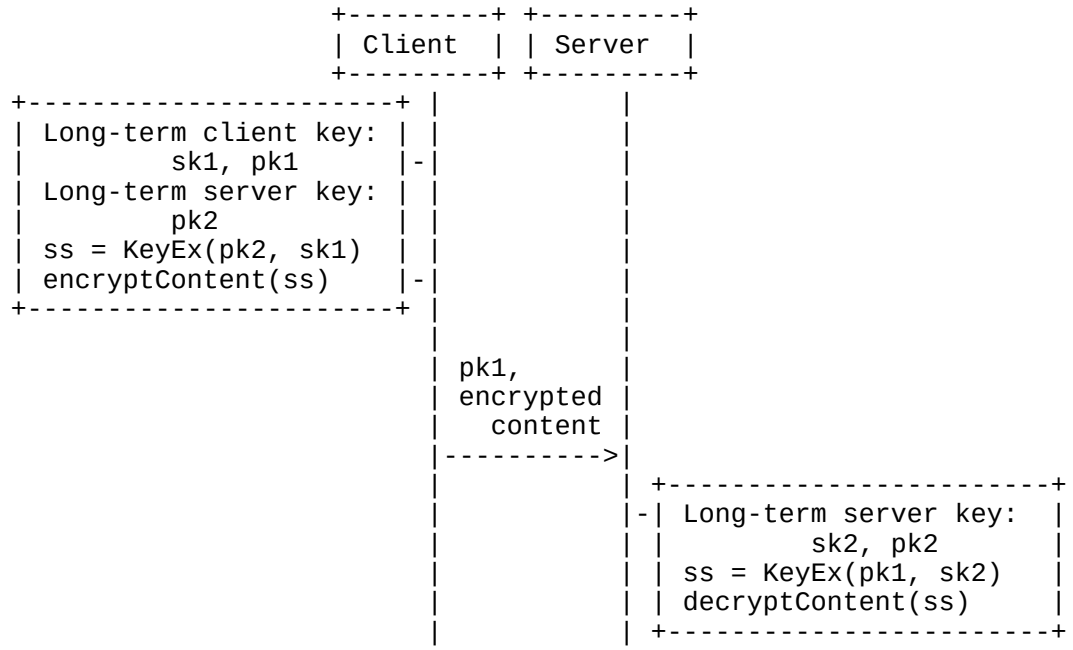

Figure 4: Diffie-Hellman based AKE and NIKE simultaneously

The complication with KEMs is that a KEM Encaps() is non-deterministic; it involves randomness chosen by the sender of that message. Therefore, in order to perform an AKE, the client must wait for the server to generate the needed randomness and perform Encaps() against the client key, which necessarily requires a network round-trip. Therefore, a KEM-based protocol can either be an AKE or a NIKE, but cannot be both at the same time. Consequently, certain Internet protocols will necessitate a redesign to accommodate this distinction, either by introducing extra network round-trips or by making trade-offs in security properties.

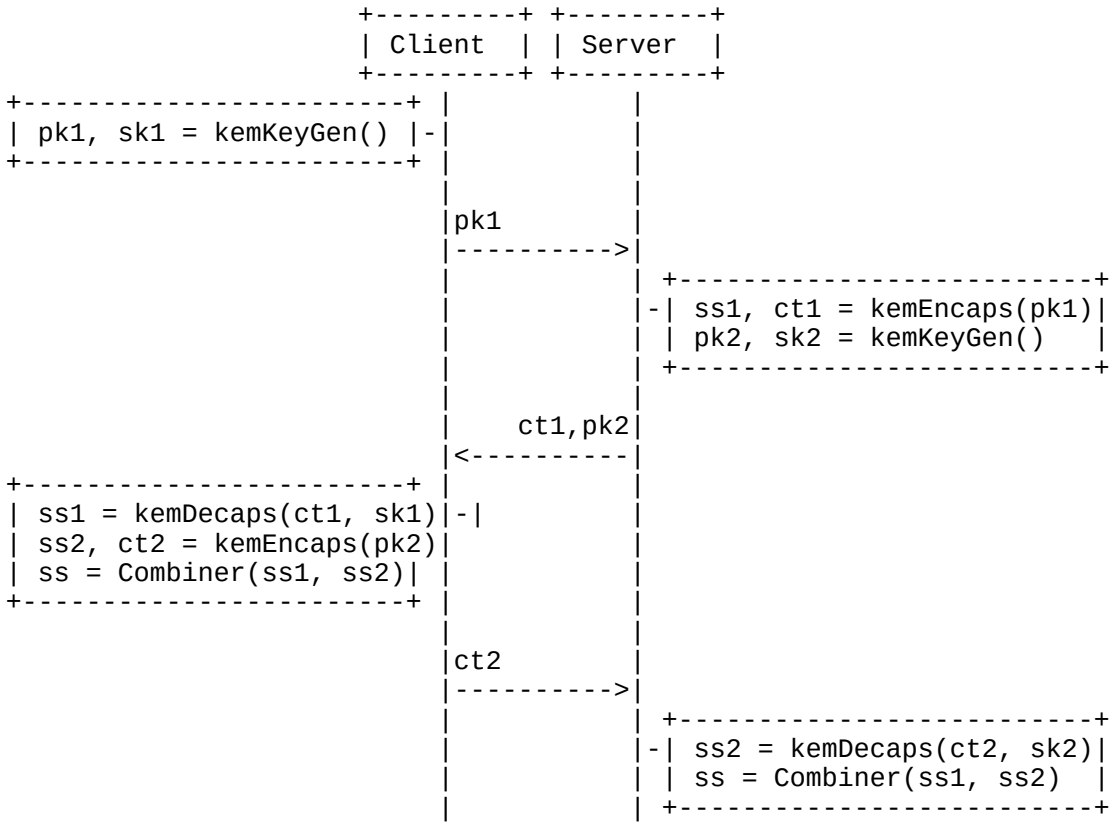

Figure 5: KEM based AKE

Here, Combiner(ss1, ss2), often referred to as a KEM Combiner, is a cryptographic construction that takes in two shared secrets and returns a single combined shared secret. The simplest combiner is concatenation ss1 || ss2, but combiners can vary in complexity depending on the cryptographic properties required. For example, if the combination should preserve IND-CCA2 of either input even if the other is chosen maliciously, then a more complex construct is required. Another consideration for combiner design is so-called "binding properties" introduced in [KEEPINGUP], which may require the ciphertexts and recipient public keys to be included in the combiner. KEM combiner security analysis becomes more complicated in hybrid settings where the two KEMs represent different algorithms, for example, where one is ML-KEM and the other is ECDH. For a more thorough discussion of KEM combiners, see [KEEPINGUP], [I-D.draft-ounsworth-cfrg-kem-combiners], and [I-D.draft-connolly-cfrg-xwing-kem].

10.2. Security Properties of KEMs

10.2.1. IND-CCA2

IND-CCA2 (INDistinguishability under adaptive Chosen-Ciphertext Attack) is an advanced security notion for encryption schemes. It ensures the confidentiality of the plaintext and resistance against chosen-ciphertext attacks. An appropriate definition of IND-CCA2 security for KEMs can be found in [CS01] and [BHK09]. ML-KEM [ML-KEM] and Classic McEliece provide IND-CCA2 security.

Understanding IND-CCA2 security is essential for individuals involved in designing or implementing cryptographic systems and protocols in order to evaluate the strength of the algorithm, assess its suitability for specific use cases, and ensure that data confidentiality and security requirements are met. Understanding IND-CCA2 security is generally not necessary for developers migrating to using an IETF-vetted key establishment method (KEM) within a given protocol or flow. IND-CCA2 is a widely accepted security notion for public key encryption mechanisms, making it suitable for a broad range of applications. IETF specification authors should include all security concerns in the "Security Considerations" section of the relevant RFC and not rely on implementers being experts in cryptographic theory.

10.2.2. Binding

KEMs also have an orthogonal set of properties to consider when designing protocols around them: binding [KEEPINGUP]. This can be "ciphertext binding", "public key binding", "context binding", or any other property that is important to not be substituted between KEM invocations. In general, a KEM is considered to bind a certain value if substitution of that value by an attacker will necessarily result in a different shared secret being derived. As an example, if an attacker can construct two different ciphertexts which will decapsulate to the same shared secret; or can construct a ciphertext which will decapsulate to the same shared secret under two different public keys, or can substitute whole KEM exchanges from one session into another, then the construction is not ciphertext binding, public key binding, or context binding respectively. Similarly, protocol designers may wish to bind protocol state information such as a transaction ID or nonce so that attempts to replay ciphertexts from one session inside a different session will be blocked at the cryptographic level because the server derives a different shared secret and is thus is unable to decrypt the content.

The solution to binding is generally achieved at the protocol design level: It is recommended to avoid using the KEM output shared secret directly without integrating it into an appropriate protocol. While KEM algorithms provide key secrecy, they do not inherently ensure source authenticity, protect against replay attacks, or guarantee freshness. These security properties should be addressed by incorporating the KEM into a protocol that has been analyzed for such protections. Even though modern KEMs such as ML-KEM produce full-entropy shared secrets, it is still advisable for binding reasons to pass it through a key derivation function (KDF) and also include all values that you wish to bind; then finally you will have a shared secret that is safe to use at the protocol level.

10.3. HPKE

Modern cryptography has long used the notion of "hybrid encryption" where an asymmetric algorithm is used to establish a key, and then a symmetric algorithm is used for bulk content encryption.

HPKE (hybrid public key encryption) [RFC9180] is a specific instantiation of this which works with a combination of KEMs, KDFs and AEAD (authenticated encryption with additional data) schemes. HPKE includes three authenticated variants, including one that authenticates possession of a pre-shared key and two optional ones that authenticate possession of a key encapsulation mechanism (KEM) private key. HPKE can be extended to support hybrid post-quantum KEM [I-D.draft-connolly-cfrg-xwing-kem]. ML-KEM does not support the static-ephemeral key exchange that allows HPKE based on DH based KEMs and its optional authenticated modes as discussed in section 1.5 of [I-D.draft-connolly-cfrg-xwing-kem].

11. PQC Signatures

Any digital signature scheme that provides a construction defining security under a post-quantum setting falls under this category of PQC signatures.

11.1. Security Properties of PQC Signatures

11.2. EUF-CMA and SUF-CMA

EUF-CMA (existential unforgeability under chosen message attack) [GMR88] is a security notion for digital signature schemes. It guarantees that an adversary, even with access to a signing oracle, cannot forge a valid signature for an arbitrary message. EUF-CMA provides strong protection against forgery attacks, ensuring the integrity and authenticity of digital signatures by preventing unauthorized modifications or fraudulent signatures. ML-DSA, FN-DSA,

and SLH-DSA provide EUF-CMA security.

SUF-CMA (strong unforgeability under chosen message attack) builds upon EUF-CMA by requiring that an adversary cannot produce a different valid signature for a message that has already been signed by the signing oracle. Like EUF-CMA, SUF-CMA provides robust assurances for digital signature schemes, further enhancing their security posture. ML-DSA, FN-DSA, and SLH-DSA also achieve SUF-CMA security.

Understanding EUF-CMA and SUF-CMA security is essential for designing or implementing cryptographic systems in order to ensure the security, reliability, and robustness of digital signature schemes. These notions allow for informed decision-making, vulnerability analysis, compliance with standards, and designing systems that provide strong protection against forgery attacks. For developers migrating to using an IETF-vetted PQC signature scheme within a given protocol or flow, a deep understanding of EUF-CMA and SUF-CMA security may not be necessary, as the schemes vetted by IETF adhere to these stringent security standards.

EUF-CMA and SUF-CMA are considered strong security benchmarks for public key signature algorithms, making them suitable for most applications. IETF specification authors should include all security concerns in the "Security Considerations" section of the relevant RFC and should not assume that implementers are experts in cryptographic theory.

11.3. Details of FN-DSA, ML-DSA, and SLH-DSA

ML-DSA [ML-DSA] is a digital signature algorithm based on the hardness of lattice problems over module lattices (i.e., the Module Learning with Errors problem (MLWE)). The design of the algorithm is based on the "Fiat-Shamir with Aborts" [Lyu09] framework introduced by Lyubashevsky, that leverages rejection sampling to render lattice-based Fiat-Shamir (FS) schemes compact and secure. ML-DSA uses uniformly-distributed random number sampling over small integers to compute coefficients in error vectors, which makes the scheme easier to implement compared with FN-DSA [FN-DSA] which uses Gaussian-distributed numbers, necessitating the need to use floating point arithmetic during signature generation.

ML-DSA offers both deterministic and randomized signing and is instantiated with 3 parameter sets providing different security levels. Security properties of ML-DSA are discussed in Section 9 of [I-D.ietf-lamps-dilithium-certificates].

FN-DSA [FN-DSA] is based on the GPV hash-and-sign lattice-based signature framework introduced by Gentry, Peikert, and Vaikuntanathan, which is a framework that requires a certain class of lattices and a trapdoor sampler technique.

The main design principle of FN-DSA is compactness, i.e., it was designed in a way that achieves minimal total memory bandwidth requirement (the sum of the signature size plus the public key size). This is possible due to the compactness of NTRU lattices. FN-DSA also offers very efficient signing and verification procedures. The main potential downsides of FN-DSA refer to the non-triviality of its algorithms and the need for floating point arithmetic support in order to support Gaussian-distributed random number sampling where the other lattice schemes use the less efficient but easier to support uniformly-distributed random number sampling.

Implementers of FN-DSA need to be aware that FN-DSA signing is highly susceptible to side-channel attacks, unless constant-time 64-bit floating-point operations are used. This requirement is extremely platform-dependent, as noted in NIST’s report.

The performance characteristics of ML-DSA and FN-DSA may differ based on the specific implementation and hardware platform. Generally, ML-DSA is known for its relatively fast signature generation, while FN-DSA can provide more efficient signature verification. The choice may depend on whether the application requires more frequent signature generation or signature verification (See [LIBOQS]). For further clarity on the sizes and security levels, please refer to the tables in sections Section 12 and Section 13.

SLH-DSA [SLH-DSA] utilizes the concept of stateless hash-based signatures, where each signature is unique and unrelated to any previous signature (as discussed in Section 9.2). This property eliminates the need for maintaining state information during the signing process. SLH-DSA was designed to sign up to 2^64 messages under a given key pair, and it offers three security levels. The parameters for each of the security levels were chosen to provide 128 bits of security, 192 bits of security, and 256 bits of security. SLH-DSA offers smaller public key sizes, larger signature sizes, slower signature generation, and slower verification when compared to ML-DSA and FN-DSA. SLH-DSA does not introduce a new hardness assumption beyond those inherent to the underlying hash functions. It builds upon established foundations in cryptography, making it a reliable and robust digital signature scheme for a post-quantum world.

All of these algorithms, ML-DSA, FN-DSA, and SLH-DSA include two signature modes: pure mode, where the entire content is signed directly, and pre-hash mode, where a digest of the content is signed.

11.4. Details of XMSS and LMS

The eXtended Merkle Signature Scheme (XMSS) [RFC8391] and Hierarchical Signature Scheme (HSS) / Leighton-Micali Signature (LMS) [RFC8554] are stateful hash-based signature schemes, where the secret key state changes over time. In both schemes, reusing a secret key state compromises cryptographic security guarantees.

XMSS and LMS can be used for signing a potentially large but fixed number of messages and the number of signing operations depends upon the size of the tree. XMSS and LMS provide cryptographic digital signatures without relying on the conjectured hardness of mathematical problems, instead leveraging the properties of cryptographic hash functions. Multi-tree XMSS and LMS (i.e., XMSS-MT and HSS respectively) use a hyper-tree based hierarchical approach with a Merkle tree at each level of the hierarchy. [RFC8391] describes both single-tree and multi-tree variants of XMSS, while [RFC8554] describes the Leighton-Micali One-Time Signature (LM-OTS) system as well as the LMS and HSS N-time signature systems. Comparison of XMSS and LMS is discussed in Section 10 of [RFC8554].

The number of tree layers in multi-tree XMSS and HSS provides a trade-off between signature size on the one side and key generation and signing speed on the other side. Increasing the number of layers reduces key generation time exponentially and signing time linearly at the cost of increasing the signature size linearly. HSS allows for customization of each subtree whereas XMSS-MT does not, electing instead to use the same structure for each subtree.

Due to the complexities described above, the XMSS and LMS are not a suitable replacement for traditional signature schemes like RSA or ECDSA. Applications that expect a long lifetime of a signature, like firmware update or secure boot, are typical use cases where those schemes can be successfully applied.

11.4.1. LMS Key and Signature Sizes

The LMS scheme is characterized by four distinct parameter sets: the underlying hash function (SHA2-256 or SHAKE-256), the length of the digest (24 or 32 bytes), the LMS tree height parameter that controls a maximal number of signatures that the private key can produce, and the width of the Winternitz coefficients (see [RFC8554], section 4.1) that can be used to trade-off signing time for signature size. Parameters can be mixed, providing 80 possible parameterizations of

the scheme.

The public (PK) and private (SK) key size depends on the length of the digest (M). The signature size depends on the digest, the Winternitz parameter (W), the LMS tree height (H), and the length of the digest. The table below provides key and signature sizes for parameterization with the digest size M=32 of the scheme.

PK	SK	W	H=5	H=10	H=15	H=20	H=25
56	52	1	8684	8844	9004	9164	9324
56	52	2	4460	4620	4780	4940	5100
56	52	4	2348	2508	2668	2828	2988
56	52	8	1292	1452	1612	1772	1932

Table 1

11.5. Hash-then-Sign

Within the hash-then-sign paradigm, the message is hashed before signing it. By pre-hashing, the onus of resistance to existential forgeries becomes heavily reliant on the collision-resistance of the hash function in use. The hash-then-sign paradigm has the ability to improve application performance by reducing the size of signed messages that need to be transmitted between application and cryptographic module, and making the signature size predictable and manageable. As a corollary, hashing remains mandatory even for short messages and assigns a further computational requirement onto the verifier. This makes the performance of hash-then-sign schemes more consistent, but not necessarily more efficient.

Using a hash function to produce a fixed-size digest of a message ensures that the signature is compatible with a wide range of systems and protocols, regardless of the specific message size or format. Crucially for hardware security modules, Hash-then-Sign also significantly reduces the amount of data that needs to be transmitted and processed by a Hardware Security Module (HSM). Consider scenarios such as a networked HSM located in a different data center from the calling application or a smart card connected over a USB interface. In these cases, streaming a message that is megabytes or gigabytes long can result in notable network latency, on-device signing delays, or even depletion of available on-device memory.

Note that the vast majority of Internet protocols that sign large messages already perform some form of content hashing at the protocol level, so this tends to be more of a concern with proprietary cryptographic protocols, and protocols from non-IETF standards bodies. Protocols like TLS 1.3 and DNSSEC use the Hash-then-Sign paradigm. In TLS 1.3 [RFC8446] CertificateVerify messages, the content that is covered under the signature includes the transcript hash output (Section 4.4.1 of [RFC8446]), while DNSSEC [RFC4034] uses it to provide origin authentication and integrity assurance services for DNS data. Similarly, the Cryptographic Message Syntax (CMS) [RFC5652] includes a mandatory message digest step before invoking the signature algorithm.

In the case of ML-DSA, it internally incorporates the necessary hash operations as part of its signing algorithm. ML-DSA directly takes the original message, applies a hash function internally, and then uses the resulting hash value for the signature generation process. In the case of SLH-DSA, it internally performs randomized message compression using a keyed hash function that can process arbitrary length messages. In the case of FN-DSA, the SHAKE-256 hash function is used as part of the signature process to derive a digest of the message being signed.

Therefore, ML-DSA, FN-DSA, and SLH-DSA offer enhanced security over the traditional Hash-then-Sign paradigm because by incorporating dynamic key material into the message digest, a pre-computed hash collision on the message to be signed no longer yields a signature forgery. Applications requiring the performance and bandwidth benefits of Hash-then-Sign may still pre-hash at the protocol level prior to invoking ML-DSA, FN-DSA, or SLH-DSA, but protocol designers should be aware that doing so re-introduces the weakness that hash collisions directly yield signature forgeries. Signing the full un-digested message is recommended where applications can tolerate it.

12. Recommendations for Security / Performance Tradeoffs

The table below denotes the five security levels provided by NIST for PQC algorithms. Neither NIST nor the IETF make any specific recommendations about which security level to use. In general, protocols will include algorithm choices at multiple levels so that users can choose the level appropriate to their policies and data classification, similar to how organizations today choose which size of RSA key to use. The security levels are defined as requiring computational resources comparable to or greater than an attack on AES (128, 192 and 256) and SHA2/SHA3 algorithms, i.e., exhaustive key recovery for AES and optimal collision search for SHA2/SHA3. This information is a re-print of information provided in the NIST PQC project [NIST] as of time this document is published.

PQ Security Level	AES/SHA(2/3) hardness	PQC Algorithm
1	AES-128 (exhaustive key recovery)	ML-KEM-512, FN-DSA-512, SLH-DSA-SHA2/SHAKE-128f/s
2	SHA-256/SHA3-256 (collision search)	ML-DSA-44
3	AES-192 (exhaustive key recovery)	ML-KEM-768, ML-DSA-65, SLH-DSA-SHA2/SHAKE-192f/s
4	SHA-384/SHA3-384 (collision search)	No algorithm tested at this level
5	AES-256 (exhaustive key recovery)	ML-KEM-1024, FN-DSA-1024, ML-DSA-87, SLH-DSA-SHA2/ SHAKE-256f/s

Table 2

The SLH-DSA-x-yf/s "f/s" in the above table denotes whether SLH-DSA is using SHAKE or SHA-2 as an underlying hash function "x" and whether it is the fast (f) or small (s) version for "y" bit AES security level. Refer to [I-D.ietf-lamps-cms-sphincs-plus] for further details on SLH-DSA algorithms.

The following table compares the signature sizes for different SLH-DSA algorithm categories at equivalent security levels, using the "simple" version. The categories include "(f)" for fast signature generation, and "(s)" for smaller signature size and faster verification, although with slower signature generation. Both SHA-256 and SHAKE-256 parameterizations produce the same signature sizes and are therefore included together in the table.

PQ Security Level	Algorithm	Public key size (in bytes)	Private key size (in bytes)	Signature size (in bytes)
1	SLH-DSA-{SHA2,SHAKE}-128f	32	64	17088
1	SLH-DSA-{SHA2,SHAKE}-128s	32	64	7856
3	SLH-DSA-{SHA2,SHAKE}-192f	48	96	35664
3	SLH-DSA-{SHA2,SHAKE}-192s	48	96	16224
5	SLH-DSA-{SHA2,SHAKE}-256f	64	128	49856
5	SLH-DSA-{SHA2,SHAKE}-256s	64	128	29792

Table 3

The following table illustrates the impact of performance on different security levels in terms of private key sizes, public key sizes, and ciphertext/signature sizes.

PQ Security Level	Algorithm	Public key size (in bytes)	Private key size (in bytes)	Ciphertext/ signature size (in bytes)
1	ML-KEM-512	800	1632	768
1	FN-DSA-512	897	1281	666
2	ML-DSA-44	1312	2560	2420
3	ML-KEM-768	1184	2400	1088
3	ML-DSA-65	1952	4032	3309
5	FN-DSA-1024	1793	2305	1280
5	ML-KEM-1024	1568	3168	1588
5	ML-DSA-87	2592	4896	4627

Table 4

13. Comparing PQC KEMs/Signatures vs Traditional KEMs (KEXs)/Signatures

This section provides two tables for comparison of different KEMs and signatures respectively, in the traditional and post-quantum scenarios. These tables focus on the secret key sizes, public key sizes, and ciphertext/signature sizes for the PQC algorithms and their traditional counterparts of similar security levels.

The first table compares traditional vs. PQC KEMs in terms of security, public and private key sizes, and ciphertext sizes.

PQ Security Level	Algorithm	Public key size (in bytes)	Private key size (in bytes)	Ciphertext size (in bytes)
Traditional	P256_HKDF_SHA-256	65	32	65
Traditional	P521_HKDF_SHA-512	133	66	133
Traditional	X25519_HKDF_SHA-256	32	32	32
1	ML-KEM-512	800	1632	768
3	ML-KEM-768	1184	2400	1088
5	ML-KEM-1024	1568	3168	1568

Table 5

The next table compares traditional vs. PQC signature schemes in terms of security, public, private key sizes, and signature sizes.

PQ Security Level	Algorithm	Public key size (in bytes)	Private key size (in bytes)	Signature size (in bytes)
Traditional	RSA2048	256	256	256
Traditional	ECDSA-P256	64	32	64
1	FN-DSA-512	897	1281	666
2	ML-DSA-44	1312	2560	2420
3	ML-DSA-65	1952	4032	3309
5	FN-DSA-1024	1793	2305	1280
5	ML-DSA-87	2592	4896	4627

Table 6

As is clear from the above table, PQC KEMs and signature schemes typically have significantly larger keys and ciphertexts/signatures than their traditional counterparts. These increased key and signatures sizes could introduce problems in protocols. As an example, IKEv2 uses UDP as the transport for its messages. One challenge with integrating a PQC KEM into IKEv2 is that IKE fragmentation cannot be utilized in the initial IKE_SA_INIT exchange. To address this issue, [RFC9242] introduces a solution by defining a new exchange called the "Intermediate Exchange" which can be fragmented using the IKE fragmentation mechanism. [RFC9370] then uses this Intermediate Exchange to carry out the PQC key exchange after the initial IKEv2 exchange and before the IKE_AUTH exchange. Another example from [SP-1800-38C] section 6.3.3 shows that increased key and signature sizes cause protocol key exchange messages to span more network packets, therefore it results in a higher total loss probability per packet. In lossy network conditions, this may increase the latency of the key exchange.

14. Post-Quantum and Traditional Hybrid Schemes

The migration to PQC is unique in the history of modern digital cryptography in that neither the traditional algorithms nor the post-quantum algorithms are fully trusted to protect data for the required lifetimes. The traditional algorithms, such as RSA and ECDH, will fall to quantum cryptanalysis, while the post-quantum algorithms face uncertainty about the underlying mathematics, compliance issues, unknown vulnerabilities, and hardware and software implementations that have not had sufficient maturing time to rule out traditional cryptanalytic attacks and implementation bugs.

During the transition from traditional to post-quantum algorithms, there may be a desire or a requirement for protocols that use both algorithm types. [I-D.ietf-pquip-pqt-hybrid-terminology] defines the terminology for the post-quantum and traditional (PQ/T) hybrid schemes.

14.1. PQ/T Hybrid Confidentiality

The PQ/T Hybrid Confidentiality property can be used to protect from a "harvest now, decrypt later" attack described in Section 8, which refers to an attacker collecting encrypted data now and waiting for quantum computers to become powerful enough to break the encryption later. Two types of hybrid key agreement schemes are discussed below.

* Concatenated hybrid key agreement scheme: The final shared secret that will be used as an input of the key derivation function is the result of the concatenation of the secrets established with

each key agreement scheme. For example, in [I-D.ietf-tls-hybrid-design], the client uses the TLS supported groups extension to advertise support for a PQ/T hybrid scheme, and the server can select this group if it supports the scheme. The hybrid-aware client and server establish a hybrid secret by concatenating the two shared secrets, which is used as the shared secret in the existing TLS 1.3 key schedule.

* Cascaded hybrid key agreement scheme: The final shared secret is computed by applying as many iterations of the key derivation function as the number of key agreement schemes composing the hybrid key agreement scheme. For example, [RFC9370] extends the Internet Key Exchange Protocol Version 2 (IKEv2) to allow one or more PQC algorithms in addition to the traditional algorithm to derive the final IKE SA keys using the cascade method as explained in Section 2.2.2 of [RFC9370].

Various instantiations of these two types of hybrid key agreement schemes have been explored. One must be careful when selecting which hybrid scheme to use. The chosen scheme for protocols like TLS 1.3 [I-D.ietf-tls-hybrid-design] has IND-CCA2 robustness, that is IND-CCA2 security is guaranteed for the scheme as long as at least one of the component algorithms is IND-CCA2 secure.

14.2. PQ/T Hybrid Authentication

The PQ/T hybrid authentication property can be utilized in scenarios where an on-path attacker possesses network devices equipped with CRQCs, capable of breaking traditional authentication protocols, or where an attacker can attack long-lived authenticated data such as CA certificates or signed software images. This property ensures authentication through a PQ/T hybrid scheme or a PQ/T hybrid protocol, as long as at least one component algorithm remains secure to provide the intended security level. For example, a PQ/T hybrid certificate [I-D.ietf-lamps-pq-composite-sigs]can be employed to facilitate a PQ/T hybrid authentication protocol. However, a PQ/T hybrid authentication protocol does not need to use a PQ/T hybrid certificate; separate certificates could be used for individual component algorithms [I-D.ietf-lamps-cert-binding-for-multi-auth]. When separate certificates are used, it may be possible for attackers to take them apart or put them together in unexpected ways, including enabling cross-protocol attacks. The exact risks this presents are highly dependent on the protocol and use case, so a full security analysis is needed. Best practices for ensuring that pairs of certificates are only used as intended are discussed in more detail in Sections 12.3.2 and 12.3.3 of this document.

The frequency and duration of system upgrades and the time when CRQCs will become widely available need to be weighed to determine whether and when to support the PQ/T Hybrid Authentication property.

14.3. Hybrid Cryptographic Algorithm Combinations: Considerations and Approaches

14.3.1. Hybrid Cryptographic Combinations

It is also possible to use more than two algorithms together in a hybrid scheme, with various methods for combining them. For post-quantum transition purposes, the combination of a post-quantum algorithm with a traditional algorithm is the most straightforward and recommended. The use of multiple post-quantum algorithms with different mathematical bases has also been considered. Combining algorithms in a way that requires both to be used together ensures stronger security, while combinations that do not require both will sacrifice security but offer other benefits like backwards compatibility and crypto agility. Including a traditional key alongside a post-quantum key often has minimal bandwidth impact.

14.3.2. Composite Keys in Hybrid Schemes

When combining keys in an "and" mode, it may make more sense to consider them to be a single composite key, instead of two keys. This generally requires fewer changes to various components of PKI ecosystems, many of which are not prepared to deal with two keys or dual signatures. To those protocol- or application-layer parsers, a "composite" algorithm composed of two "component" algorithms is simply a new algorithm, and support for adding new algorithms generally already exists. Treating multiple "component" keys as a single "composite" key also has security advantages such as preventing cross-protocol reuse of the individual component keys and guarantees about revoking or retiring all component keys together at the same time, especially if the composite is treated as a single object all the way down into the cryptographic module.

All that needs to be done is to standardize the formats of how the two keys from the two algorithms are combined into a single data structure, and how the two resulting signatures or KEMs are combined into a single signature or KEM. The answer can be as simple as concatenation, if the lengths are fixed or easily determined. At the time this document is published, security research is ongoing as to the security properties of concatenation-based composite signatures and KEMs vs more sophisticated signature and KEM combiners, and in which protocol contexts those simpler combiners are sufficient.

One last consideration is the specific pairs of algorithms that can be combined. A recent trend in protocols is to only allow a small number of "known good" configurations that make sense, often referred to in cryptography as a "ciphersuite", instead of allowing arbitrary combinations of individual configuration choices that may interact in dangerous ways. The current consensus is that the same approach should be followed for combining cryptographic algorithms, and that "known good" pairs should be explicitly listed ("explicit composite"), instead of just allowing arbitrary combinations of any two cryptographic algorithms ("generic composite").

The same considerations apply when using multiple certificates to transport a pair of related keys for the same subject. Exactly how two certificates should be managed in order to avoid some of the pitfalls mentioned above is still an active area of investigation. Using two certificates keeps the certificate tooling simple and straightforward, but in the end simply moves the problems with requiring that both certs are intended to be used as a pair, must produce two signatures which must be carried separately, and both must validate, to the certificate management layer, where addressing these concerns in a robust way can be difficult.

At least one scheme has been proposed that allows the pair of certificates to exist as a single certificate when being issued and managed, but dynamically split into individual certificates when needed ([I-D.draft-bonnell-lamps-chameleon-certs].

14.3.3. Key Reuse in Hybrid Schemes

An important security note, particularly when using hybrid signature keys, but also to a lesser extent hybrid KEM keys, is key reuse. In traditional cryptography, problems can occur with so-called "cross-protocol attacks" when the same key can be used for multiple protocols; for example signing TLS handshakes and signing S/MIME emails. While it is not best-practice to reuse keys within the same protocol, for example using the same key for multiple S/MIME certificates for the same user, it is not generally catastrophic for security. However, key reuse becomes a large security problem within hybrids.

Consider an {RSA, ML-DSA} hybrid key where the RSA key also appears within a single-algorithm certificate. In this case, an attacker could perform a "stripping attack" where they take some piece of data signed with the {RSA, ML-DSA} key, remove the ML-DSA signature and present the data as if it was intended for the RSA only certificate. This leads to a set of security definitions called "non-separability properties", which refers to how well the signature scheme resists various complexities of downgrade / stripping attacks

[I-D.draft-ietf-pquip-hybrid-signature-spectrums]. Therefore, it is recommended that implementers either reuse the entire hybrid key as a whole, or perform fresh key generation of all component keys per usage, and must not take an existing key and reuse it as a component of a hybrid.

14.3.4. Jurisdictional Fragmentation

Another potential application of hybrids bears mentioning, even though it is not directly PQC-related. That is using hybrids to navigate inter-jurisdictional cryptographic connections. Traditional cryptography is already fragmented by jurisdiction: consider that while most jurisdictions support Elliptic Curve Diffie-Hellman, those in the United States will prefer the NIST curves while those in Germany will prefer the Brainpool curves. China, Russia, and other jurisdictions have their own national cryptography standards. This situation of fragmented global cryptography standards is unlikely to improve with PQC. If "and" mode hybrids become standardized for the reasons mentioned above, then one could imagine leveraging them to create "ciphersuites" in which a single cryptographic operation simultaneously satisfies the cryptographic requirements of both endpoints.

14.3.5. Future Directions and Ongoing Research

Many aspects of hybrid cryptography are still under investigation. LAMPS WG at IETF is actively exploring the security properties of these combinations, and future standards will reflect the evolving consensus on these issues.

15. Security Considerations

15.1. Cryptanalysis

Traditional cryptanalysis exploits weaknesses in algorithm design, mathematical vulnerabilities, or implementation flaws, that are exploitable with classical (i.e. non-quantum) hardware, whereas quantum cryptanalysis harnesses the power of CRQCs to solve specific mathematical problems more efficiently. Another form of quantum cryptanalysis is "quantum side-channel" attacks. In such attacks, a device under threat is directly connected to a quantum computer, which then injects entangled or superimposed data streams to exploit hardware that lacks protection against quantum side-channels. Both pose threats to the security of cryptographic algorithms, including those used in PQC. Developing and adopting new cryptographic algorithms resilient against these threats is crucial for ensuring long-term security in the face of advancing cryptanalysis techniques.

Recent attacks on the side-channel implementations using deep learning based power analysis have also shown that one needs to be cautious while implementing the required PQC algorithms in hardware. Two of the most recent works include one attack on ML-KEM [KyberSide] and one attack on Saber [SaberSide]. An evolving threat landscape points to the fact that lattice based cryptography is indeed more vulnerable to side-channel attacks as in [SideCh], [LatticeSide]. Consequently, there were some mitigation techniques for side channel attacks that have been proposed as in [Mitigate1], [Mitigate2], and [Mitigate3].

15.2. Cryptographic Agility

Cryptographic agility is recommended for both traditional and quantum cryptanalysis as it enables organizations to adapt to emerging threats, adopt stronger algorithms, comply with standards, and plan for long-term security in the face of evolving cryptanalytic techniques and the advent of CRQCs.

Several PQC schemes are available that need to be tested; cryptography experts around the world are pushing for the best possible solutions, and the first standards that will ease the introduction of PQC are being prepared. It is of paramount importance and a call for imminent action for organizations, bodies, and enterprises to start evaluating their cryptographic agility, assess the complexity of implementing PQC into their products, processes, and systems, and develop a migration plan that achieves their security goals to the best possible extent.

An important and often overlooked step in achieving cryptographic agility is maintaining a cryptographic inventory. Modern software stacks incorporate cryptography in numerous places, making it challenging to identify all instances. Therefore, cryptographic agility and inventory management take two major forms: First, application developers responsible for software maintenance should actively search for instances of hard-coded cryptographic algorithms within applications. When possible, they should design the choice of algorithm to be dynamic, based on application configuration. Second, administrators, policy officers, and compliance teams should take note of any instances where an application exposes cryptographic configurations. These instances should be managed either through organization-wide written cryptographic policies or automated cryptographic policy systems.

Numerous commercial solutions are available for both detecting hard-coded cryptographic algorithms in source code and compiled binaries, as well as providing cryptographic policy management control planes for enterprise and production environments.

15.3. Hybrid Key Exchange and Signatures: Bridging the Gap Between Post-Quantum and Traditional Cryptography

Post-quantum algorithms selected for standardization are relatively new and they have not been subject to the same depth of study as traditional algorithms. PQC implementations will also be new and therefore more likely to contain implementation bugs than the battle-tested crypto implementations that are relied on today. In addition, certain deployments may need to retain traditional algorithms due to regulatory constraints, for example FIPS [SP-800-56C] or PCI compliance [PCI]. Hybrid key exchange is recommended to enhance security against the "harvest now, decrypt later" attack. Additionally, hybrid signatures provide for time to react in the case of the announcement of a devastating attack against any one algorithm, while not fully abandoning traditional cryptosystems.

Hybrid key exchange performs both a classical and a post-quantum key exchange in parallel. It provides security redundancy against potential weaknesses in PQ algorithms, allows for a gradual transition of trust in PQC algorithms, and, in backward-compatible designs, enables gradual adoption without breaking compatibility with existing systems. For instance, in TLS 1.3, a hybrid key exchange can combine a widely supported classical algorithm, such as X25519, with a post-quantum algorithm like ML-KEM. This allows legacy clients to continue using the classical algorithm while enabling upgraded clients to proceed with hybrid key exchange. In contrast, overhead-spreading hybrid designs focus on reducing the PQ overhead. For example, approaches like those described in [I-D.hale-mls-combiner] amortize PQ costs by selectively applying PQ updates in key exchange processes, allowing systems to balance security and efficiency. This strategy ensures a post-quantum secure channel while keeping the overhead manageable, making it particularly suitable for constrained environments.

While some hybrid key exchange options introduce additional computational and bandwidth overhead, the impact of traditional key exchange algorithms (e.g., key size) is typically small, helping to keep the overall increase in resource usage manageable for most systems. In highly constrained environments, however, those hybrid key exchange protocols may be impractical due to their higher resource requirements compared to pure post-quantum or traditional key exchange approaches. However, some hybrid key exchange designs distribute the PQC overhead, making them more suitable for constrained environments. The choice of hybrid key exchange design depends on the specific system requirements and use case, so the appropriate approach may vary.

15.4. Caution: Ciphertext commitment in KEM vs DH

The ciphertext generated by a KEM is not necessarily directly linked to the shared secret it produces. KEMs allow for multiple ciphertexts to encapsulate the same shared secret, which enables flexibility in key management without enforcing a strict one-to-one correspondence between ciphertexts and shared secrets. This allows for secret reuse across different recipients, sessions, or operational contexts without the need for new secrets for each use, simplifying key distribution and reducing computational overhead. In contrast, cryptographic schemes like Diffie-Hellman inherently link the public key to the derived shared secret, meaning any change in the public key results in a different shared secret.

16. IANA Considerations

This document has no IANA considerations.

17. Further Reading & Resources

A good book on modern cryptography is Serious Cryptography, 2nd Edition, by Jean-Philippe Aumasson, ISBN 9781718503847.

The Open Quantum Safe (OQS) Project [OQS] is an open-source project that aims to support the transition to quantum-resistant cryptography.

The IETF’s PQUIP Working Group [PQUIP-WG] maintains a list of PQC-related protocol work within the IETF.

18. Informative References

[AddSig] "AddSig", n.d., <https://csrc.nist.gov/Projects/pqc-dig-sig/standardization>.

[ANSSI] "ANSSI views on the Post-Quantum Cryptography transition", n.d., <https://cyber.gouv.fr/sites/default/files/document/follow_up_position_paper_on_post_quantum_cryptography.pdf>.

[BHK09] "Subtleties in the Definition of IND-CCA: When and How Should Challenge-Decryption be Disallowed?", <https://eprint.iacr.org/2009/418>.

[BIKE] "BIKE", n.d., <http://pqc-hqc.org/>.

[BPQS] "BPQS", n.d., <https://eprint.iacr.org/2018/658.pdf>.

[BSI-PQC] "Quantum-safe cryptography fundamentals, current developments and recommendations", May 2022, <https://www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/Publications/Brochure/quantum-safe-cryptography.html?nn=916626>.

[ClassicMcEliece] "Classic McEliece", n.d., <https://classic.mceliece.org/>.

[Cloudflare] "NIST's pleasant post-quantum surprise", <https://blog.cloudflare.com/nist-post-quantum-surprise/>.

[CNSA2-0] "Announcing the Commercial National Security Algorithm Suite 2.0", <https://media.defense.gov/2022/Sep/07/2003071834/-1/-1/0/CSA_CNSA_2.0_ALGORITHMS_.PDF>.

[CRQCThreat] "CRQCThreat", n.d., <https://sam-jaques.appspot.com/quantum_landscape_2024>.

[CS01] "Design and Analysis of Practical Public-Key Encryption Schemes Secure against Adaptive Chosen Ciphertext Attack", <https://eprint.iacr.org/2001/108>.

[FN-DSA] "Fast Fourier lattice-based compact signatures over NTRU", <https://falcon-sign.info/>.

[FrodoKEM] "FrodoKEM", n.d., <https://frodokem.org/>.

[GMR88] "A digital signature scheme secure against adaptive chosen-message attacks.", <https://people.csail.mit.edu/silvio/Selected%20Scientific%20Papers/Digital%20Signatures/A_Digital_Signature_Scheme_Secure_Against_Adaptive_Chosen-Message_Attack.pdf>.

[Grover-search] "C. Zalka, "Grover's quantum searching algorithm is optimal," Physical Review A, vol. 60, pp. 2746-2751, 1999.".

[HQC] "HQC", n.d., <http://pqc-hqc.org/>.

[I-D.draft-bonnell-lamps-chameleon-certs] Bonnell, C., Gray, J., Hook, D., Okubo, T., and M. Ounsworth, "A Mechanism for Encoding Differences in Paired Certificates", Work in Progress, Internet-Draft, draft-

bonnell-lamps-chameleon-certs-06, 16 April 2025, <https://datatracker.ietf.org/doc/html/draft-bonnell-lamps-chameleon-certs-06>.

[I-D.draft-connolly-cfrg-xwing-kem]
Connolly, D., Schwabe, P., and B. Westerbaan, "X-Wing: general-purpose hybrid post-quantum KEM", Work in Progress, Internet-Draft, draft-connolly-cfrg-xwing-kem-07, 3 May 2025, <https://datatracker.ietf.org/doc/html/draft-connolly-cfrg-xwing-kem-07>.

[I-D.draft-ietf-lake-edhoc]
Selander, G., Mattsson, J. P., and F. Palombini, "Ephemeral Diffie-Hellman Over COSE (EDHOC)", Work in Progress, Internet-Draft, draft-ietf-lake-edhoc-23, 22 January 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-lake-edhoc-23>.

[I-D.draft-ietf-pquip-hybrid-signature-spectrums]
Bindel, N., Hale, B., Connolly, D., and F. D, "Hybrid signature spectrums", Work in Progress, Internet-Draft, draft-ietf-pquip-hybrid-signature-spectrums-07, 20 June 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-pquip-hybrid-signature-spectrums-07>.

[I-D.draft-ounsworth-cfrg-kem-combiners]
Ounsworth, M., Wussler, A., and S. Kousidis, "Combiner function for hybrid key encapsulation mechanisms (Hybrid KEMs)", Work in Progress, Internet-Draft, draft-ounsworth-cfrg-kem-combiners-05, 31 January 2024, <https://datatracker.ietf.org/doc/html/draft-ounsworth-cfrg-kem-combiners-05>.

[I-D.hale-mls-combiner]
Jol, Hale, B., Mularczyk, M., and X. Tian, "Flexible Hybrid PQ MLS Combiner", Work in Progress, Internet-Draft, draft-hale-mls-combiner-01, 26 September 2024, <https://datatracker.ietf.org/doc/html/draft-hale-mls-combiner-01>.

[I-D.ietf-lamps-cert-binding-for-multi-auth]
Becker, A., Guthrie, R., and M. J. Jenkins, "Related Certificates for Use in Multiple Authentications within a Protocol", Work in Progress, Internet-Draft, draft-ietf-lamps-cert-binding-for-multi-auth-06, 10 December 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-lamps-cert-binding-for-multi-auth-06>.

[I-D.ietf-lamps-cms-sphincs-plus]
Housley, R., Fluhrer, S., Kampanakis, P., and B. Westerbaan, "Use of the SLH-DSA Signature Algorithm in the Cryptographic Message Syntax (CMS)", Work in Progress, Internet-Draft, draft-ietf-lamps-cms-sphincs-plus-19, 13 January 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-lamps-cms-sphincs-plus-19>.

[I-D.ietf-lamps-dilithium-certificates]
Massimo, J., Kampanakis, P., Turner, S., and B. Westerbaan, "Internet X.509 Public Key Infrastructure - Algorithm Identifiers for the Module-Lattice-Based Digital Signature Algorithm (ML-DSA)", Work in Progress, Internet-Draft, draft-ietf-lamps-dilithium-certificates-12, 26 June 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-lamps-dilithium-certificates-12>.

[I-D.ietf-lamps-pq-composite-sigs]
Ounsworth, M., Gray, J., Pala, M., Klauner, J., and S. Fluhrer, "Composite ML-DSA for use in X.509 Public Key Infrastructure", Work in Progress, Internet-Draft, draft-ietf-lamps-pq-composite-sigs-06, 18 June 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-lamps-pq-composite-sigs-06>.

[I-D.ietf-pquip-pqt-hybrid-terminology]
D, F., P, M., and B. Hale, "Terminology for Post-Quantum Traditional Hybrid Schemes", Work in Progress, Internet-Draft, draft-ietf-pquip-pqt-hybrid-terminology-06, 10 January 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-pquip-pqt-hybrid-terminology-06>.

[I-D.ietf-tls-hybrid-design]
Stebila, D., Fluhrer, S., and S. Gueron, "Hybrid key exchange in TLS 1.3", Work in Progress, Internet-Draft, draft-ietf-tls-hybrid-design-13, 17 June 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-tls-hybrid-design-13>.

[I-D.irtf-cfrg-bbs-signatures]
Looker, T., Kalos, V., Whitehead, A., and M. Lodder, "The BBS Signature Scheme", Work in Progress, Internet-Draft, draft-irtf-cfrg-bbs-signatures-08, 3 March 2025, <https://datatracker.ietf.org/doc/html/draft-irtf-cfrg-bbs-signatures-08>.

[KEEPINGUP]
"Keeping Up with the KEMs: Stronger Security Notions for KEMs and automated analysis of KEM-based protocols", n.d., <https://eprint.iacr.org/2023/1933>.

[KyberSide]
"A Side-Channel Attack on a Hardware Implementation of CRYSTALS-Kyber", <https://eprint.iacr.org/2022/1452>.

[LattFail1]
"Decryption Failure Attacks on IND-CCA Secure Lattice-Based Schemes", <https://link.springer.com/chapter/10.1007/978-3-030-17259-6_19#chapter-info>.

[LattFail2]
"(One) Failure Is Not an Option: Bootstrapping the Search for Failures in Lattice-Based Encryption Schemes.", <https://link.springer.com/chapter/10.1007/978-3-030-45727-3_1>.

[LatticeSide]
"Generic Side-channel attacks on CCA-secure lattice-based PKE and KEM schemes", <https://eprint.iacr.org/2019/948>.

[LIBOQS] "LibOQS - Open Quantum Safe", <https://github.com/open-quantum-safe/liboqs>.

[Lyu09] "V. Lyubashevsky, “Fiat-Shamir With Aborts: Applications to Lattice and Factoring-Based Signatures “, ASIACRYPT 2009", <https://www.iacr.org/archive/asiacrypt2009/59120596/59120596.pdf>.

[Mitigate1]
"POLKA: Towards Leakage-Resistant Post-Quantum CCA-Secure Public Key Encryption", <https://eprint.iacr.org/2022/873>.

[Mitigate2]
"Leakage-Resilient Certificate-Based Authenticated Key Exchange Protocol", <https://ieeexplore.ieee.org/document/9855226>.

[Mitigate3]
"Post-Quantum Authenticated Encryption against Chosen-Ciphertext Side-Channel Attacks", <https://eprint.iacr.org/2022/916>.

[ML-DSA] "FIPS-204: Module-Lattice-Based Digital Signature Standard", <https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.204.pdf>.

[ML-KEM] "FIPS-203: Module-Lattice-based Key-Encapsulation Mechanism Standard", <https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.203.pdf>.

[NIST] "Post-Quantum Cryptography Standardization", <https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization>.

[NISTFINAL] "NIST Releases First 3 Finalized Post-Quantum Encryption Standards", n.d., <https://www.nist.gov/news-events/news/2024/08/nist-releases-first-3-finalized-post-quantum-encryption-standards>.

[NTRU] "NTRU", n.d., <https://ntru.org/index.shtml>.

[OQS] "Open Quantum Safe Project", n.d., <https://openquantumsafe.org/>.

[PCI] "Payment Card Industry Data Security Standard", n.d., <https://docs-prv.pcisecuritystandards.org/PCI%20DSS/Standard/PCI-DSS-v4_0_1.pdf>.

[PQCAPI] "PQC - API notes", <https://csrc.nist.gov/CSRC/media/Projects/Post-Quantum-Cryptography/documents/example-files/api-notes.pdf>.

[PQRSA] "Post-quantum RSA", April 2017, <https://cr.yp.to/papers/pqrsa-20170419.pdf>.

[PQUIP-WG] "Post-Quantum Use In Protocols (pquip) Working Group", n.d., <https://datatracker.ietf.org/group/pquip/documents/>.

[QC-DNS] "Quantum Computing and the DNS", <https://www.icann.org/octo-031-en.pdf>.

[QuantSide] "QuantSide", n.d., <https://arxiv.org/pdf/2304.03315>.

[RFC4034] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "Resource Records for the DNS Security Extensions", RFC 4034, DOI 10.17487/RFC4034, March 2005, <https://www.rfc-editor.org/rfc/rfc4034>.

[RFC5652] Housley, R., "Cryptographic Message Syntax (CMS)", STD 70, RFC 5652, DOI 10.17487/RFC5652, September 2009, <https://www.rfc-editor.org/rfc/rfc5652>.

[RFC6090] McGrew, D., Igoe, K., and M. Salter, "Fundamental Elliptic Curve Cryptography Algorithms", RFC 6090, DOI 10.17487/RFC6090, February 2011, <https://www.rfc-editor.org/rfc/rfc6090>.

[RFC8391] Huelsing, A., Butin, D., Gazdag, S., Rijneveld, J., and A. Mohaisen, "XMSS: eXtended Merkle Signature Scheme", RFC 8391, DOI 10.17487/RFC8391, May 2018, <https://www.rfc-editor.org/rfc/rfc8391>.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/rfc/rfc8446>.

[RFC8554] McGrew, D., Curcio, M., and S. Fluhrer, "Leighton-Micali Hash-Based Signatures", RFC 8554, DOI 10.17487/RFC8554, April 2019, <https://www.rfc-editor.org/rfc/rfc8554>.

[RFC9180] Barnes, R., Bhargavan, K., Lipp, B., and C. Wood, "Hybrid Public Key Encryption", RFC 9180, DOI 10.17487/RFC9180, February 2022, <https://www.rfc-editor.org/rfc/rfc9180>.

[RFC9242] Smyslov, V., "Intermediate Exchange in the Internet Key Exchange Protocol Version 2 (IKEv2)", RFC 9242, DOI 10.17487/RFC9242, May 2022, <https://www.rfc-editor.org/rfc/rfc9242>.

[RFC9370] Tjhai, CJ., Tomlinson, M., Bartlett, G., Fluhrer, S., Van Geest, D., Garcia-Morchon, O., and V. Smyslov, "Multiple Key Exchanges in the Internet Key Exchange Protocol Version 2 (IKEv2)", RFC 9370, DOI 10.17487/RFC9370, May 2023, <https://www.rfc-editor.org/rfc/rfc9370>.

[RSA] "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems+", <https://dl.acm.org/doi/pdf/10.1145/359340.359342>.

[RSA10SC] "Breaking RSA Encryption - an Update on the State-of-the-Art", <https://www.quintessencelabs.com/blog/breaking-rsa-encryption-update-state-art>.

[RSA8HRS] "How to factor 2048 bit RSA integers in 8 hours using 20 million noisy qubits", <https://arxiv.org/abs/1905.09749>.

[RSAShor] "Circuit for Shor' s algorithm using 2n+3 qubits", <https://arxiv.org/pdf/quant-ph/0205095.pdf>.

[SaberSide] "A side-channel attack on a masked and shuffled software implementation of Saber", <https://link.springer.com/article/10.1007/s13389-023-00315-3>.

[SideCh] "Side-Channel Attacks on Lattice-Based KEMs Are Not Prevented by Higher-Order Masking", <https://eprint.iacr.org/2022/919>.

[SIDH-Attack] "An efficient key recovery attack on SIDH", n.d., <https://eprint.iacr.org/2022/975.pdf>.

[SIKE] "SIKE Supersingular Isogeny Key Encapsulation", n.d., <https://sike.org/>.

[SLH-DSA] "FIPS-205: Stateless Hash-Based Digital Signature Standard", <https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.205.pdf>.

[SP-1800-38C] "Migration to Post-Quantum Cryptography Quantum Readiness: Quantum-Resistant Cryptography Technology Interoperability and Performance Report", <https://www.nccoe.nist.gov/sites/default/files/2023-12/pqc-migration-nist-sp-1800-38c-preliminary-draft.pdf>.

[SP-800-56C] "Recommendation for Key-Derivation Methods in Key-Establishment Schemes", <https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-56Cr2.pdf>.

[Threat-Report] "Quantum Threat Timeline Report 2020", <https://globalriskinstitute.org/publications/quantum-threat-timeline-report-2020/>.

Acknowledgements

This document leverages text from an earlier draft by Paul Hoffman. Thanks to Dan Wing, Florence D, Thom Wiggers, Sophia Grundner-Culemann, Panos Kampanakis, Ben S, Sofia Celi, Melchior Aelmans, Falko Strenzke, Deirdre Connolly, Hani Ezzadeen, Britta Hale, Scott Rose, Hilarie Orman, Thomas Fossati, and Daniel Van Geest for the discussion, review and comments.

In particular, the authors would like to acknowledge the contributions to this document by Kris Kwiatkowski.

Authors' Addresses

Aritra Banerjee
Nokia
London
United Kingdom
Email: aritra.banerjee@nokia.com

Tirumaleswar Reddy
Nokia
Bangalore
Karnataka
India
Email: k.tirumaleswar_reddy@nokia.com

Dimitrios Schoinianakis
Nokia
Athens
Greece
Email: dimitrios.schoinianakis@nokia-bell-labs.com

Timothy Hollebeek
DigiCert
Pittsburgh,
United States of America
Email: tim.hollebeek@digicert.com

Mike Ounsworth
Entrust Limited
2500 Solandt Road Suite 100
Ottawa, Ontario K2K 3G5
Canada

Email: mike.ounsworth@entrust.com