Workgroup: PQUIP
Internet-Draft:
draft-ietf-pquip-pqc-engineers-03
Published: 22 February 2024
Intended Status: Informational
Expires: 25 August 2024
Authors: A. Banerjee T. Reddy D. Schoinianakis
Nokia Nokia Nokia
T. Hollebeek
DigiCert

Post-Quantum Cryptography for Engineers

Abstract

The presence of a Cryptographically Relevant Quantum Computer (CRQC) would render state-of-the-art, traditional public-key algorithms deployed today obsolete, since the assumptions about the intractability of the mathematical problems for these algorithms that offer confident levels of security today no longer apply in the presence of a CRQC. This means there is a requirement to update protocols and infrastructure to use post-quantum algorithms, which are public-key algorithms designed to be secure against CRQCs as well as classical computers. These new public-key algorithms behave similarly to previous public key algorithms, however the intractable mathematical problems have been carefully chosen so they are hard for CRQCs as well as classical computers. This document explains why engineers need to be aware of and understand post-quantum cryptography. It emphasizes the potential impact of CRQCs on current cryptographic systems and the need to transition to post-quantum algorithms to ensure long-term security. The most important thing to understand is that this transition is not like previous transitions from DES to AES or from SHA-1 to SHA-2. While drop-in replacement may be possible in some cases, others will require protocol re-design to accommodate significant differences in behavior between the new post-quantum algorithms and the classical algorithms that they are replacing.

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

Copyright Notice

Table of Contents

1. Introduction
2. Conventions and Definitions
3. Contributing to This Document
4. Traditional Cryptographic Primitives that Could Be Replaced by PQC
5. Invariants of Post-Quantum Cryptography
6. NIST PQC Algorithms
   - 6.1. NIST candidates selected for standardization
     - 6.1.1. PQC Key Encapsulation Mechanisms (KEMs)
     - 6.1.2. PQC Signatures
   - 6.2. Candidates advancing to the fourth-round for standardization at NIST
7. Threat of CRQCs on Cryptography
   - 7.1. Symmetric cryptography
   - 7.2. Asymmetric cryptography
8. Timeline for transition

9. Post-quantum cryptography categories
9.1. Lattice-Based Public-Key Cryptography
9.2. Hash-Based Public-Key Cryptography
9.3. Code-Based Public-Key Cryptography
10. KEMs
10.1. What is a KEM
10.1.1. Authenticated Key Exchange (AKE)
10.2. Security property
10.3. HPKE
11. PQC Signatures
11.1. What is a Post-quantum Signature
11.2. Security property
11.3. Details of FALCON, Dilithium, and SPHINCS+
11.4. Details of XMSS and LMS
11.4.1. LMS scheme - key and signature sizes
11.5. Hash-then-Sign
12. Recommendations for Security / Performance Tradeoffs
13. Comparing PQC KEMs/Signatures vs Traditional KEMs (KEXs)/ Signatures
14. Post-Quantum and Traditional Hybrid Schemes
14.1. PQ/T Hybrid Confidentiality
14.2. PQ/T Hybrid Authentication
14.3. Additional Considerations
15. Security Considerations
15.1. Cryptanalysis
15.2. Cryptographic Agility
15.3. Hybrid Key Exchange and Signatures: Bridging the Gap Between Post-Quantum and Traditional Cryptography
15.4. Caution: Ciphertext commitment in KEM vs DH
16. Further Reading & Resources
16.1. Reading List
16.2. Developer Resources
17. Contributors
Acknowledgements
References
Normative References
Informative References
Authors' Addresses

1. Introduction

Quantum computing is no longer perceived as a conjecture of computational sciences and theoretical physics. Considerable research efforts and enormous corporate and government funding for the development of practical quantum computing systems are being invested currently. At the time of writing the document, Cryptographically Relevant Quantum Computers (CRQCs) that can break widely used public-key cryptographic algorithms are not yet available. However, it is worth noting that 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 all of modern cryptography, happen to fall within the niche that we expect quantum computers to excel at. As such, as quantum technology advances, there is the potential for future quantum computers to have a significant impact on current cryptographic systems. Predicting the emergence of CRQC is a challenging task, and there is ongoing uncertainty regarding when they will become practically feasible.

Extensive research has produced several "post-quantum cryptographic (PQC) algorithms" (sometimes referred to as "quantum-safe" 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. This document aims to provide general guidance to engineers who utilize public-key cryptography in their software. It covers topics such as selecting appropriate PQC algorithms, understanding the differences between PQC Key Encapsulation Mechanisms (KEMs) and traditional Diffie-Hellman style key exchange, and provides insights into expected key sizes and processing time differences between PQC algorithms and traditional ones. Additionally, it discusses the potential threat to symmetric cryptography from Cryptographically Relevant Quantum Computers (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. It might be worth mentioning that recently NSA released an article on Future Quantum-Resistant (QR) Algorithm Requirements for National Security Systems [CNSA2-0] based on the need to protect against deployments of CRQCs in the future. Germany's BSI has also released a PQC migration and recommendations document [BSI-PQC] which largely aligns with United States NIST and NSA guidance, but does differ on some of the guidance.

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 algorithms from the Symmetric side based on stream, block ciphers, hash functions, MACs, etc. 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 or Quantum Key Generation, which use quantum hardware to exploit quantum effects to protect communications and generate keys, respectively. Post-quantum cryptography is based on conventional (i.e., non-quantum) math and software and can be run on any general purpose computer.

Please note: 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 Cryptographic Forum Research Group (CFRG).

While there is ongoing discussion about whether to use the term 'Post-Quantum' or 'Quantum Ready/Resistant' to describe algorithms that resist CRQCs, a consensus has not yet been reached. It's 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. The term "quantum resistant" or "quantum ready" are used for algorithms which are synonymous with Post-Quantum termed algorithms but a final decision has not yet been reached as to the ambiguity of these terms.

2. Conventions and Definitions

The key words "**MUST**", "**MUST NOT**", "**REQUIRED**", "**SHALL**", "**SHALL NOT**", "**SHOULD**", "**SHOULD NOT**", "**RECOMMENDED**", "**NOT RECOMMENDED**", "**MAY**", and "**OPTIONAL**" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

3. Contributing to This Document

The guide was inspired by a thread in September 2022 on the pqc@ietf.org mailing list. The document is being collaborated on through a GitHub repository.

The editors actively encourage contributions to this document. Please consider writing a section on a topic that you think is missing. Short of that, writing a paragraph or two on an issue you found when writing code that uses PQC would make this document more useful to other coders. Opening issues that suggest new material is fine too, but relying on others to write the first draft of such material is much less likely to happen than if you take a stab at it yourself.

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 sufficiently large general-purpose quantum computer, known as 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 this is based on public key cryptography and is therefore vulnerable to the Shor's algorithm. A CRQC can employ Shor's algorithm to efficiently find the prime factors of a large public key (in 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 exponent or discrete logarithm of the (short or long-term) Diffie-Hellman public key. This, in turn, would reveal the precise secret required to derive the session 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 public key cryptography will also affect traditional digital signatures, hence the importance of developing post-quantum digital signatures.

5. Invariants of Post-Quantum Cryptography

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-SHA2), rely on secret keys shared between the sender and

receiver. 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 (such as SHA-2) and a secret key shared between the sender and receiver to produce a message authentication code. CRQCs, in theory, do not offer substantial advantages in breaking symmetric-key algorithms compared to classical computers (see Section 7.1 for more details).

6. NIST PQC Algorithms

In 2016, the National Institute of Standards and Technology (NIST) started a process to solicit, evaluate, and standardize one or more quantum-resistant public-key cryptographic algorithms, as seen here. The first set of algorithms for standardization (https://csrc.nist.gov/publications/detail/nistir/8413/final) were selected in July 2022.

NIST announced as well that they will be opening a fourth round to standardize an alternative KEM, and a call for new candidates for a post-quantum signature algorithm.

These algorithms are not a drop-in replacement for classical 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 classical algorithms with either a PQC KEM or a PQC signature method, depending on how the classical 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)

* CRYSTALS-Kyber: Kyber is a module learning with errors (MLWE)-based key encapsulation mechanism (Section 9.1).

6.1.2. PQC Signatures

* CRYSTALS-Dilithium: CRYSTALS-Dilithium is a lattice signature scheme (Section 9.1 and Section 11.3).

* Falcon: Falcon is a lattice signature scheme (Section 9.1 and Section 11.3).

*SPHINCS+: SPHINCS+ is a stateless hash-based signature scheme (Section 9.2 and Section 11.3).

6.2. Candidates advancing to the fourth-round for standardization at NIST

The fourth-round of the NIST process focuses only on KEMs. The goal of that round is to select an alternative algorithm that is based on different hard problem than Kyber. The candidates still advancing for standardization are:

*Classic McEliece: 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.

*BIKE: Based on the the hardness of syndrome decoding of QC-MDPC codes. Quasi-Cyclic Moderate Density Parity Check (QC-MDPC) code are a class of error correcting codes that leverages bit flipping technique to efficiently correct errors.

*HQC : Based on the hardness of syndrome decoding of Quasi-cyclic concatenated Reed Muller Reed Solomon (RMRS) codes in the Hamming metric. Reed Muller (RM) codes are a class of block error correcting codes used especially in wireless and deep space communications. Reed Solomon (RS) are a class of block error correcting codes that are used to detect and correct multiple bit errors.

*SIKE (Broken): Supersingular Isogeny Key Encapsulation (SIKE) is a specific realization of the SIDH (Supersingular Isogeny Diffie-Hellman) protocol. Recently, a mathematical attack based on the "glue-and-split" theorem from 1997 from Ernst Kani was found against the underlying chosen starting curve and torsion information. In practical terms, this attack allows for the efficient recovery of the private key. NIST announced that SIKE was no longer under consideration, but the authors of SIKE had asked for it to remain in the list so that people are aware that it is broken. While SIKE is broken, Isogenies in general remain an active area of cryptographic research due to their very attractive bandwidth usage, and we may yet see more cryptographic primitives in the future from this research area.

7. Threat of CRQCs on Cryptography

Post-quantum cryptography or quantum-safe cryptography refers to cryptographic algorithms that are secure against cryptographic attacks from both CRQCs and classic computers.

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.

7.1. Symmetric cryptography

Grover's algorithm is a quantum search algorithm that provides a theoretical quadratic speedup for searching an unstructured database, compared to classical algorithms. If we 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 achieve quantum resistance. 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, 2^{64} operations performed in parallel are feasible for modern classical computers, but 2^{64} quantum operations performed serially in a quantum computer are not. 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]). Therefore, while Grover's attack suggests that we should double the sizes of symmetric keys, the current consensus among experts is that the current key sizes remain secure in practice.

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.

How can someone be sure that an improved algorithm won’t outperform Grover's algorithm at some point in time? Christof Zalka has shown that Grover's algorithm (and in particular its non-parallel nature) achieves the best possible complexity for unstructured search [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 classical 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 classical and quantum attacks. As a result, 128-bit algorithms can be considered quantum-safe for the foreseeable future.

7.2. Asymmetric cryptography

“Shor’s algorithm” on the other side, efficiently solves the integer factorization problem (and the related discrete logarithm problem), which offer 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 classic computer, it needs a CRQC.

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

For structured data such as public-key and signatures, instead, CRQCs can fully solve the underlying hard problems used in classic cryptography (see Shor's Algorithm). Because an increase of 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 classic public-key cryptography (i.e., the integer factorization problem, the finite-field discrete logarithm problem, and the elliptic-curve discrete logarithm problem).

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, we are 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 can be decrypted 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 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.

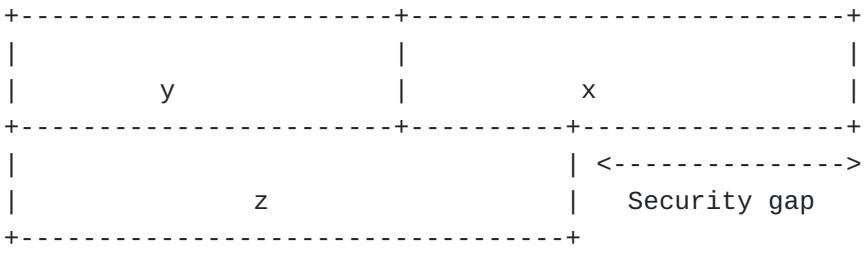

Figure 1: Mosca model

These challenges are illustrated nicely by the so-called Mosca model discussed in [Threat-Report]. In the Figure 1, "x" denotes the time that our 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 are we preparing our IT systems during those "y" years (or in other words, how can one 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 we track 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.

9. Post-quantum cryptography categories

The current set of problems used in post-quantum cryptography can be currently grouped into three different categories: lattice-based, hash-based and code-based.

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, Shortest 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.

The possibility to implement public-key schemes on lattices is tied to the characteristics of the vector basis used for the lattice. In particular, solving any of the mentioned problems can be easy when using "reduced" or "good" bases (i.e., as short as possible and as orthogonal as possible), while it becomes computationally infeasible when using "bad" bases (i.e., long vectors not orthogonal). Although the problem might seem trivial, it is computationally hard when considering many dimensions, or when the underlying field is not simple numbers, but high-order polynomials. Therefore, a typical approach is to use "bad" basis for public keys and "good" basis for private keys. The public keys ("bad" basis) let you easily verify signatures by checking, for example, that a vector is the closest or smallest, but do not let you solve the problem (i.e., finding the vector) that would yield the private key. Conversely, private keys (i.e., the "good" basis) can be used for generating the signatures (e.g., finding the specific vector).

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 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 such class of algorithms include Kyber, Falcon and Dilithium.

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 Post-Quantum 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 this point in time (July 2023), the PQC candidates by NIST are considered secure under these attacks and we suggest 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 mathematically based on the security of the selected 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 schemes. Unlike digital signature techniques, most hash-based signature schemes are stateful, which means that signing necessitates the update and careful tracking of the secret key. Producing multiple signatures using the same secret key state results in loss of security and ultimately signature forgery attacks against that key.

The SPHINCS algorithm on the other hand leverages the HORST (Hash to Obtain Random Subset with Trees) technique and remains the only hash based signature scheme that is stateless.

SPHINCS+ is an advancement on SPHINCS which reduces the signature sizes in SPHINCS and makes it more compact. SPHINCS+ was recently standardized by NIST.

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 the 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 NIST Round 4 (unbroken) finalists: Classic McEliece, HQC, BIKE.

10. KEMs

10.1. What is a KEM

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 in one of two ways: (1) Derive a Data Encryption Key (DEK) to encrypt the data (2) Derive a Key Encryption Key (KEK) used to wrap the DEK. The term "encapsulation" is chosen intentionally to indicate that KEM algorithms behave differently at the API level than the Key Agreement or Key Encipherment / Key Transport mechanisms that we are accustomed to using today. Key Agreement schemes imply that both parties contribute a public / private keypair 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:

KEM relies on the following primitives [PQCAPI]:

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

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

*def kemDecaps(ct, sk) -> ss

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

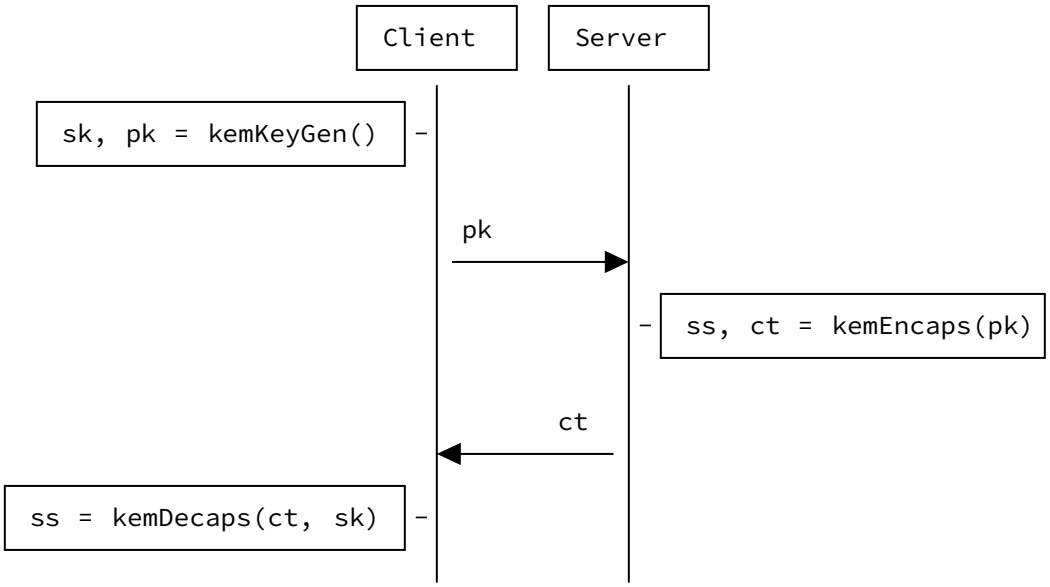

Figure 2: KEM based Key Exchange

10.1.1. Authenticated Key Exchange (AKE)

Authenticated Key Exchange 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-22]. 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], explained in [hpke]. The following figure illustrates the Diffie-Hellman (DH) Key exchange:

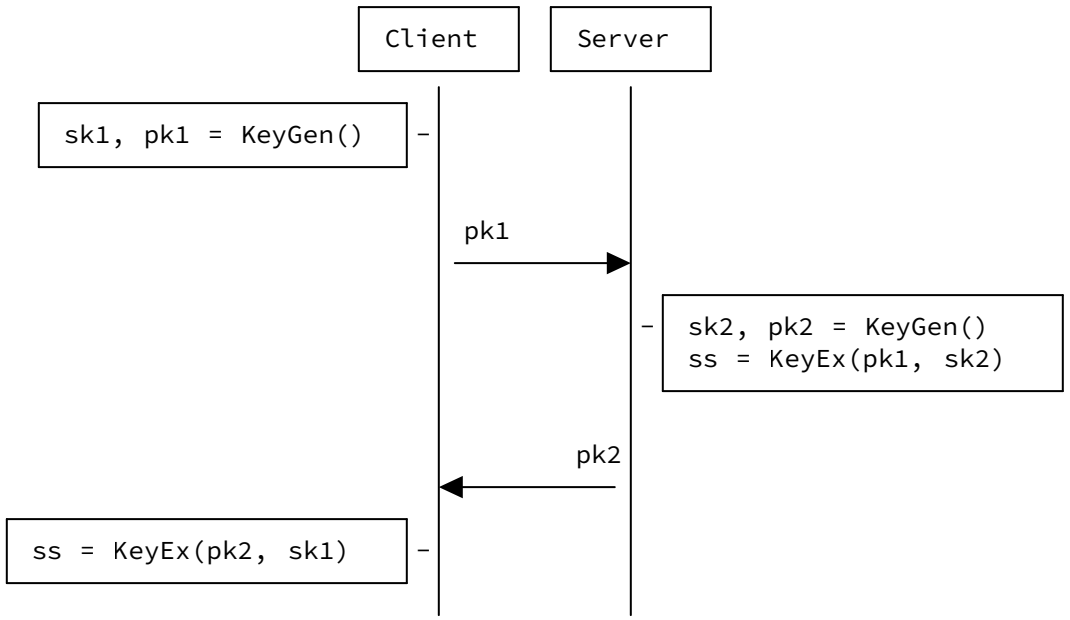

Figure 3: Diffie-Hellman based Authenticated Key Exchange

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 Authenticated Key Exchange (AKE). It's also worth noting that in an Ephemeral-Static Diffie-Hellman (DH) scenario, where sk2 and pk2 represent long-term keys. such as those contained in an email encryption certificate, the client can compute ss = KeyEx(pk2, sk1) without waiting for a response from the Server. This characteristic transforms it into a non-interactive and authenticated key exchange method. Many Internet protocols rely on this aspect of DH. When using Key Encapsulation Mechanisms (KEMs) as the underlying primitive, a flow may be non-interactive or authenticated, but not both. Consequently, certain Internet protocols will necessitate redesign to accommodate this distinction, either by introducing extra network round-trips or by making trade-offs in security properties.

Post-Quantum KEMs are inherently interactive Key Exchange (KE) protocols because they involve back-and-forth communication to negotiate and establish a shared secret key. This is unlike Diffie-Hellman (DH) Key Exchange (KEX) or RSA Key Transport, which provide the non-interactive key exchange (NIKE) property. NIKE is a cryptographic primitive that enables two parties who know each other's public keys to agree on a symmetric shared key without requiring any real-time interaction. Consider encrypted email, where the content needs to be encrypted and sent even if the receiving device containing the decryption keys (e.g., a phone or laptop) is currently offline.

Another important property of Diffie-Hellman is that in addition to being a NIKE, it is also an Authenticated Key Exchange (AKE), meaning that since both parties needed to involve their asymmetric keypair, both parties have proof-of-identity of the other party. In order to achieve an AKE with KEM primitives, two full KEM exchanges need to be performed, and their results combined to form a single shared secret.

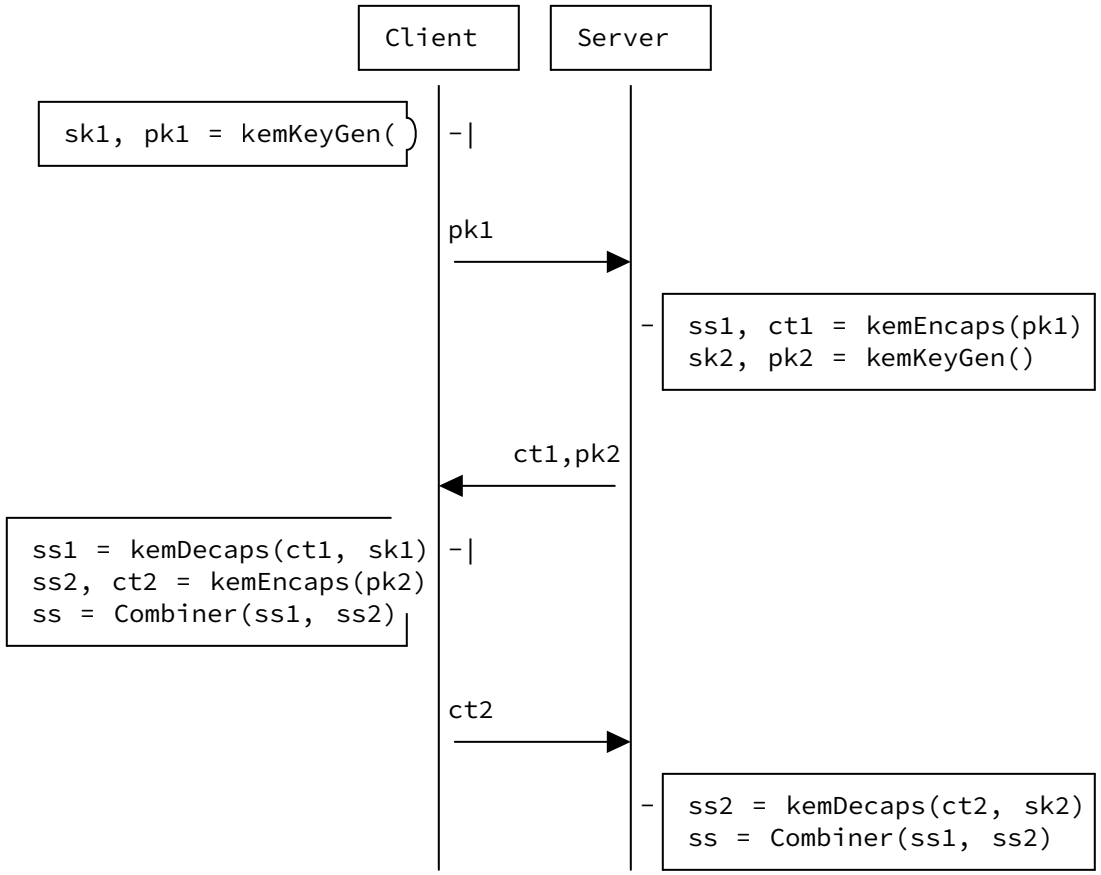

Figure 4: KEM based Authenticated Key Exchange

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. Sometimes combiners require both the shared secrets and ciphertexts as input and can act on more than two KEMs; Combiner(ss1, ct1, ss2, ct2, ..). For a more thorough discussion of KEM combiners, see [I-D.draft-ounsworth-cfrg-kem-combiners-04].

10.2. Security property

* 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, resistance against chosen-ciphertext attacks, and prevents the adversary from forging valid ciphertexts (given access to the public key). An appropriate definition of IND-CCA2 security for KEMs can be found in [CS01] and [BHK09]. Kyber and Classic McEliece provides IND-CCA2 security.

Understanding IND-CCA2 security is essential for individuals involved in designing or implementing cryptographic systems and protocols 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. IETF specification authors should include all security concerns in the 'Security Considerations' section of the relevant RFC and not rely on implementers being deep experts in cryptographic theory.

10.3. HPKE

HPKE (Hybrid Public Key Encryption) [RFC9180] deals with a variant of KEM which is essentially a PKE of arbitrary sized plaintexts for a recipient public key. It works with a combination of KEMs, KDFs and AEAD schemes (Authenticated Encryption with Additional Data). 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.westerbaan-cfrg-hpke-xyber768d00-02]. Kyber, which is a 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 1.2 of [I-D.westerbaan-cfrg-hpke-xyber768d00-02].

11. PQC Signatures

11.1. What is a Post-quantum Signature

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

11.2. Security property

* EUF-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. Dilithium, Falcon and SPHINCS+ provide EUF-CMA security.

Understanding EUF-CMA security is essential for individual involved in designing or implementing cryptographic systems to ensure the security, reliability, and trustworthiness of digital signature schemes. It allows for informed decision-making, vulnerability analysis, compliance with standards, and designing systems that provide strong protection against forgery attacks. Understanding EUF-CMA security is generally not necessary for developers migrating to using an IETF-vetted post-quantum cryptography (PQC) signature scheme within a given protocol or flow. IETF specification authors should include all security concerns in the 'Security Considerations' section of the relevant RFC and should not assume that implementers are deep experts in cryptographic theory

11.3. Details of FALCON, Dilithium, and SPHINCS+

Dilithium [Dilithium] is a digital signature algorithm (part of the CRYSTALS suite) based on the hardness 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 FS schemes compact and secure. Dilithium uses uniform distribution over small integers for computing coefficients in error vectors, which makes the scheme easier to implement.

Dilithium offers both deterministic and randomized signing and is instantiated with 3 parameter sets providing different security

levels. Security properties of Dilithium are discussed in Section 9 of [I-D.ietf-lamps-dilithium-certificates].

Falcon [Falcon] 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 class of lattices and a trapdoor sampler technique.

The main design principle of Falcon 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. Falcon also offers very efficient signing and verification procedures. The main potential downsides of Falcon 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 Falcon need to be aware that Falcon 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 Dilithium and Falcon may differ based on the specific implementation and hardware platform. Generally, Dilithium is known for its relatively fast signature generation, while Falcon 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.

SPHINCS+ [SPHINCS] 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. SPHINCS+ was designed to sign up to 2^64 messages 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. SPHINCS+ offers smaller key sizes, larger signature sizes, slower signature generation, and slower verification when compared to Dilithium and Falcon. SPHINCS+ 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. The advantages and

disadvantages of SPHINCS+ over other signature algorithms is discussed in Section 3.1 of [I-D.draft-ietf-cose-sphincs-plus].

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 changes over time. In both schemes, reusing a secret key state compromises cryptographic security guarantees.

Multi-Tree 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. XMSS and Hierarchical Signature System (HSS) use a 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 XMSS^MT 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.

Due to the complexities described above, the XMSS and LMS are not a suitable replacement for classical 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 succesfully applied.

11.4.1. LMS scheme - key and signature sizes

The LMS scheme is characterized by four distinct parameter sets - underlying hash function (SHA2-256 or SHAKE-256), the length of the digest (24 or 32 bytes), LMS tree height - parameter that controls a maximal number of signatures that the private key can produce (possible values are 5,10,15,20,25) and the width of the Winternitz coefficients (see [RFC8554], section 4.1) that can be used to trade-off signing time for signature size (possible values are 1,2,4,8). Parameters can be mixed, providing 80 possible parametrizations 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 Winternitz

parameter (W), the LMS tree height (H), and the length of the digest. The tables 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 performance by reducing the size of signed messages, 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. 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.

Protocols like TLS 1.3 and DNSSEC use the Hash-then-Sign paradigm. In TLS 1.3 [RFC8446] CertificateVerify message, the content that is covered under the signature includes the transcript hash output (Section 4.4.1 of [RFC8446]), while DNSSEC [RFC4033] uses it to provide origin authentication and integrity assurance services for DNS data.

In the case of Dilithium, it internally incorporates the necessary hash operations as part of its signing algorithm. Dilithium directly takes the original message, applies a hash function internally, and then uses the resulting hash value for the signature generation process. In case of SPHINCS+, it internally performs randomized message compression using a keyed hash function that can process arbitrary length messages. In case of Falcon, a hash function is

used as part of the signature process, it uses the SHAKE-256 hash function to derive a digest of the message being signed. Therefore, Dilithium, Falcon, and SPHINCS+ 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 Dilithium, Falcon, or SPHINCS+, 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 strongly preferred where applications can tolerate it.

12. Recommendations for Security / Performance Tradeoffs

The table below denotes the 5 security levels provided by NIST required for PQC algorithms. Users can leverage the appropriate algorithm based on the security level required for their use case. 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 of writing (July 2023).

PQ Security Level	AES/SHA(2/3) hardness	PQC Algorithm
1	AES-128 (exhaustive key recovery)	Kyber512, Falcon512, Sphincs+SHA-256 128f/s
2	SHA-256/SHA3-256 (collision search)	Dilithium2
3	AES-192 (exhaustive key recovery)	Kyber768, Dilithium3, Sphincs+SHA-256 192f/s
4	SHA-384/SHA3-384 (collision search)	No algorithm tested at this level
5	AES-256 (exhaustive key recovery)	Kyber1024, Falcon1024, Dilithium5, Sphincs+SHA-256 256f/s

Table 2

Please note the Sphincs+SHA-256 x"f/s" in the above table denotes whether its the Sphincs+ fast (f) version or small (s) version for "x" bit AES security level. Refer to [I-D.ietf-lamps-cms-sphincs-plus-02] for further details on Sphincs+ algorithms.

The following table discusses the signature size differences for similar SPHINCS+ algorithm security levels with the "simple" version

but for different categories i.e., (f) for fast verification and (s) for compactness/smaller. Both SHA-256 and SHAKE-256 parametrisation output the same signature sizes, so both have been included.

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

Table 3

The following table discusses 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	Kyber512	800	1632	768
1	Falcon512	897	1281	666
2	Dilithium2	1312	2528	2420
3	Kyber768	1184	2400	1088
5	Falcon1024	1793	2305	1280
5	Kyber1024	1568	3168	1588

Table 4

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

In this section, we provide two tables for comparison of different KEMs and Signatures respectively, in the traditional and post-quantum scenarios. These tables will 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, 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	Kyber512	800	1632	768
3	Kyber768	1184	2400	1088
5	Kyber1024	1568	3168	1588

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	P256	64	32	64
1	Falcon512	897	1281	666
2	Dilithium2	1312	2528	768
3	Dilithium3	1952	4000	3293
5	Falcon1024	1793	2305	1280

Table 6

As one can clearly observe from the above tables, leveraging a PQC KEM/Signature significantly increases the key sizes and the ciphertext/signature sizes compared to traditional KEM(KEX)/ Signatures. But the PQC algorithms do provide the additional security level in case there is an attack from a CRQC, whereas schemes based on prime factorization or discrete logarithm problems (finite field or elliptic curves) would provide no level of security at all against such attacks.

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 PQC key exchange into the initial IKEv2 exchange is that IKE fragmentation cannot be utilized. 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 elliptic curve, 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 classical 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 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:

1. Concatenate 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.

2. Cascade 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 and will be discussed further. One must be careful when selecting which hybrid scheme to use. The chosen schemes at IETF are IND-CCA2 robust, 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. 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 instance, a PQ/T hybrid certificate 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 [I-D.ounsworth-pq-composite-keys]; separate certificates could be used for individual component algorithms [I-D.ietf-lamps-cert-binding-for-multi-auth].

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

14.3. Additional Considerations

It is also possible to use more than two algorithms together in a hybrid scheme, and there are multiple possible ways those algorithms can be combined. For the purposes of a post-quantum transition, the simple combination of a post-quantum algorithm with a single classical algorithm is the most straightforward, but the use of multiple post-quantum algorithms with different hard math problems has also been considered. When combining algorithms, it is possible to require that both algorithms be used together (the so-called "and" mode) or that only one does (the "or" mode), or even some more complicated scheme. Schemes that do not require both algorithms to validate only have the strength of the weakest algorithm, and therefore offer little or no security benefit but may offer backwards compatibility, crypto agility, or ease-of-migration benefits. Care should be taken when designing "or" mode hybrids to ensure that the larger PQ keys are not required to be transmitted to and processed by legacy clients that will not use them; this was the major drawback of the failed proposal

[I-D.draft-truskovsky-lamps-pq-hybrid-x509]. This combination of properties makes optionally including post-quantum keys without requiring their use to be generally unattractive in most use cases. On the other hand, including a classical key -- particularly an elliptic curve key -- alongside a lattice key is generally considered to be negligible in terms of the extra bandwidth usage.

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 time of writing (August 2023) 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 pairs of algorithms that can be combined. A recent trends in protocols is to only allow a small number of "known good" configurations that make sense, 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 crypto 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 (https://datatracker.ietf.org/doc/draft-bonnell-lamps-chameleon-certs/).

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 standardised 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.

Many of these points are still being actively explored and discussed, and the consensus may change over time.

15. Security Considerations

15.1. Cryptanalysis

Classical 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 Kyber [KyberSide] and one attack on Saber [SaberSide]. 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 relevant for both classical 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 hardcoded 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 hardcoded 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 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 we rely 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. Hybrid key exchange enables potential security against "Harvest Now, Decrypt Later" attack and 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.

15.4. Caution: Ciphertext commitment in KEM vs DH

The ciphertext generated by a KEM is not necessarily inherently linked to the shared secret it produces. In contrast, in some other cryptographic schemes like Diffie-Hellman, a change in the public key results in a change in the derived shared secret.

16. Further Reading & Resources

16.1. Reading List

(A reading list. Serious Cryptography. Pointers to PQC sites with good explanations. List of reasonable Wikipedia pages.)

16.2. Developer Resources

*Open Quantum Safe and corresponding github

*PQUIP WG list of PQC-related protocol work within the IETF

17. Contributors

The authors would like to acknowledge that this content is assembled from countless hours of discussion and countless megabytes of email discussions. We have tried to reference as much source material as possible, and apologize to anyone whose work was inadvertently missed.

In particular, the authors would like to acknowledge the contributions to this document by the following individuals:

Kris Kwiatkowski

PQShield, LTD

United Kingdom.

kris@amongbytes.com

Mike Ounsworth

Entrust

Canada

mike.ounsworth@entrust.com

Acknowledgements

This document leverages text from https://github.com/paulehoffman/post-quantum-for-engineers/blob/main/pqc-for-engineers.md. Thanks to Dan Wing, Florence D, Thom Wiggers, Sophia Grundner-Culemann, Panos Kampanakis, Ben S3, Sofia Celi, Melchior Aelmans, Falko Strenzke, Deirdre Connolly, and Daniel Van Geest for the discussion, review and comments.

References

Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/rfc/rfc2119>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/rfc/rfc8174>.

[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>.

[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>.

[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>.

Informative References

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

[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>.

[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>.

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

[Dilithium] "Cryptographic Suite for Algebraic Lattices (CRYSTALS) - Dilithium", <https://pq-crystals.org/dilithium/index.shtml>.

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

[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.".

[I-D.draft-ietf-cose-sphincs-plus] Prorock, M., Steele, O., Misoczki, R., Osborne, M., and C. Cloostermans, "SLH-DSA for JOSE and COSE", Work in Progress, Internet-Draft, draft-ietf-cose-sphincs-plus-02, 12 January 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-cose-sphincs-plus-02>.

[I-D.draft-ietf-lake-edhoc-22] Selander, G., Mattsson, J. P., and F. Palombini, "Ephemeral Diffie-Hellman Over COSE (EDHOC)", Work in Progress, Internet-Draft, draft-ietf-lake-

edhoc-22, 25 August 2023, <https://datatracker.ietf.org/doc/html/draft-ietf-lake-edhoc-22>.

[I-D.draft-ounsworth-cfrg-kem-combiners-04] 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-04, 8 July 2023, <https://datatracker.ietf.org/doc/html/draft-ounsworth-cfrg-kem-combiners-04>.

[I-D.draft-truskovsky-lamps-pq-hybrid-x509] Truskovsky, A., Van Geest, D., Fluhrer, S., Kampanakis, P., Ounsworth, M., and S. Mister, "Multiple Public-Key Algorithm X.509 Certificates", Work in Progress, Internet-Draft, draft-truskovsky-lamps-pq-hybrid-x509-02, 24 August 2023, <https://datatracker.ietf.org/doc/html/draft-truskovsky-lamps-pq-hybrid-x509-02>.

[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-03, 29 November 2023, <https://datatracker.ietf.org/doc/html/draft-ietf-lamps-cert-binding-for-multi-auth-03>.

[I-D.ietf-lamps-cms-sphincs-plus-02] Housley, R., Fluhrer, S., Kampanakis, P., and B. Westerbaan, "Use of the SPHINCS+ Signature Algorithm in the Cryptographic Message Syntax (CMS)", Work in Progress, Internet-Draft, draft-ietf-lamps-cms-sphincs-plus-02, 17 May 2023, <https://datatracker.ietf.org/doc/html/draft-ietf-lamps-cms-sphincs-plus-02>.

[I-D.ietf-lamps-dilithium-certificates] Massimo, J., Kampanakis, P., Turner, S., and B. Westerbaan, "Internet X.509 Public Key Infrastructure: Algorithm Identifiers for ML-DSA", Work in Progress, Internet-Draft, draft-ietf-lamps-dilithium-certificates-03, 5 February 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-lamps-dilithium-certificates-03>.

[I-D.ietf-pquip-pqt-hybrid-terminology] D, F., "Terminology for Post-Quantum Traditional Hybrid Schemes", Work in Progress, Internet-Draft, draft-ietf-pquip-pqt-hybrid-terminology-02, 2 February 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-pquip-pqt-hybrid-terminology-02>.

[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-09, 7 September 2023, <https://datatracker.ietf.org/doc/html/draft-ietf-tls-hybrid-design-09>.

[I-D.ounsworth-pq-composite-keys] Ounsworth, M., Gray, J., Pala, M., and J. Klaußner, "Composite Public and Private Keys For Use In Internet PKI", Work in Progress, Internet-Draft, draft-ounsworth-pq-composite-keys-05, 29 May 2023, <https://datatracker.ietf.org/doc/html/draft-ounsworth-pq-composite-keys-05>.

[I-D.westerbaan-cfrg-hpke-xyber768d00-02] Westerbaan, B. and C. A. Wood, "X25519Kyber768Draft00 hybrid post-quantum KEM for HPKE", Work in Progress, Internet-Draft, draft-westerbaan-cfrg-hpke-xyber768d00-02, 4 May 2023, <https://datatracker.ietf.org/doc/html/draft-westerbaan-cfrg-hpke-xyber768d00-02>.

[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>.

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

[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>.

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

[RFC4033] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "DNS Security Introduction and Requirements", RFC 4033, DOI 10.17487/RFC4033, March 2005, <https://www.rfc-editor.org/rfc/rfc4033>.

[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>.

[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>.

[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>.

[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>.

[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>.

[SPHINCS] "SPHINCS+", <https://sphincs.org/index.html>.

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

Authors' Addresses

Aritra Banerjee
Nokia
Munich
Germany

Email: aritra.banerjee@nokia.com

Tirumaleswar Reddy
Nokia
Bangalore
Karnataka
India

Email: kondtir@gmail.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