uta
Internet-Draft
Intended status: Standards Track
Expires: 22 March 2026

T. Reddy
Nokia
H. Tschofenig
H-BRS

18 September 2025

Post-Quantum Cryptography Recommendations for TLS-based Applications
draft-ietf-uta-pqc-app-00

Abstract

Post-quantum cryptography presents new challenges for device manufacturers, application developers, and service providers. This document highlights the unique characteristics of applications and offers best practices for implementing quantum-ready usage profiles in applications that use TLS and key supporting protocols such as DNS.

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-reddy-uta-pqc-app/.

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

Status of This Memo

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 22 March 2026.

Copyright Notice

Copyright (c) 2025 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (https://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

Table of Contents

1. Introduction

The visible face of the Internet predominantly comprises services operating on a client-server architecture, where a client communicates with an application service. When using protocols such as TLS 1.3 [RFC8446], DTLS 1.3 [RFC9147], or protocols built on these foundations (e.g., QUIC [RFC9001]), clients and servers perform ephemeral public-key exchanges, such as Elliptic Curve Diffie-Hellman (ECDH), to derive a shared secret that ensures forward secrecy. Additionally, they validate each other's identities through X.509 certificates, establishing secure communication.

The emergence of a Cryptographically Relevant Quantum Computer (CRQC) would render current public-key algorithms insecure and obsolete. This is because the mathematical assumptions underpinning these algorithms, which currently offer high levels of security, would no longer hold in the presence of a CRQC. Consequently, there is an urgent need to update protocols and infrastructure with post-quantum cryptographic (PQC) algorithms. These algorithms are designed to remain secure against both CRQCs and classical computers. The traditional cryptographic primitives requiring replacement are discussed in [I-D.ietf-pquip-pqc-engineers], and the NIST PQC Standardization process has selected algorithms such as ML-KEM, SLH-DSA, and ML-DSA as candidates for future deployment in protocols.

Historically, the industry has successfully transitioned between cryptographic protocols, such as upgrading TLS versions and deprecating older ones (e.g., SSLv2), and shifting from RSA to Elliptic Curve Cryptography (ECC), which improved security and reduced key sizes. However, the transition to PQC presents unique challenges, primarily due to the following:

1. Algorithm Maturity: While NIST has finalized a set of PQC algorithms, ensuring the correctness and security of implementations remains critical. Even the most secure algorithm is vulnerable if implementation flaws introduce security risks.

2. Key and Signature Sizes: Many PQC algorithms require significantly larger key and signature sizes, which can inflate handshake packet sizes and impact network performance. For example, ML-KEM public keys are substantially larger than ECDH keys (see Table 5 in [I-D.ietf-pquip-pqc-engineers]). Similarly, public keys for SLH-DSA and ML-DSA are much larger than those for P256 (see Table 6 in [I-D.ietf-pquip-pqc-engineers]). Signature sizes for algorithms like SLH-DSA and ML-DSA are also considerably larger compared to traditional options like Ed25519 or ECDSA-P256, posing challenges for constrained environments (e.g., IoT) and increasing handshake times in high-latency or lossy networks.

3. Performance Trade-Offs: While some PQC algorithms exhibit slower operations compared to traditional algorithms, others provide specific advantages. For instance, ML-KEM requires less CPU than X25519, and ML-DSA offers faster signature verification times compared to Ed25519, although its signature generation process is slower.

Any application transmitting messages over untrusted networks is potentially vulnerable to active or passive attacks by adversaries, including those equipped with CRQCs. The degree of vulnerability varies depending on the application, the underlying systems, the value of the data being transmitted, and the attractiveness of attacking a particular individual, device, or flow. This document outlines quantum-ready usage profiles for applications designed to protect against passive and on-path attacks leveraging CRQCs. It also discusses how TLS client and server implementations, together with essential supporting protocols (e.g., DNS), can address these challenges using various techniques detailed in subsequent sections.

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.

This document adopts terminology defined in [I-D.ietf-pquip-pqt-hybrid-terminology]. For the purposes of this document, it is useful to categorize cryptographic algorithms into three distinct classes:

* Traditional Algorithm: An asymmetric cryptographic algorithm based on integer factorization, finite field discrete logarithms, or elliptic curve discrete logarithms. In the context of TLS, an

example of a traditional key exchange algorithm is Elliptic Curve Diffie-Hellman (ECDH), which is almost exclusively used in its ephemeral mode, referred to as Elliptic Curve Diffie-Hellman Ephemeral (ECDHE).

* Post-Quantum Algorithm: An asymmetric cryptographic algorithm designed to be secure against attacks from both quantum and classical computers. An example of a post-quantum key exchange algorithm is the Module-Lattice Key Encapsulation Mechanism (ML-KEM). Such algorithms rely on mathematical problems (e.g., lattices) that are believed to be hard for both classical and CRQCs to solve efficiently.

* Hybrid Algorithm: We distinguish between key exchanges and signature algorithms:

  - Hybrid Key Exchange: A key exchange mechanism that combines two component algorithms - one traditional algorithm and one post-quantum algorithm. The resulting shared secret remains secure as long as at least one of the component key exchange algorithms remains unbroken.

  - PQ/T Hybrid Digital Signature: A multi-algorithm digital signature scheme composed of two or more component signature algorithms, where at least one is a post-quantum algorithm and at least one is a traditional algorithm.

Digital signature algorithms play a critical role in X.509 certificates, Certificate Transparency Signed Certificate Timestamps, Online Certificate Status Protocol (OCSP) statements, remote attestation evidence, and any other mechanism that contributes signatures during a TLS handshake or in context of a secure communication establishment.

3. Timeline for Transition

The timeline and driving motivations for transitioning to quantum-ready cryptography differ between data confidentiality and data authentication (e.g., signatures). The risk of "Harvest Now, Decrypt Later" (HNDL) attacks demands immediate action to protect data confidentiality (see Section 7 of [I-D.ietf-pquip-pqc-engineers]), while the threat to authentication systems, although less urgent, requires forward-thinking planning to mitigate future risks.

Encrypted payloads transmitted using Transport Layer Security (TLS) are vulnerable to decryption if an attacker equipped with a CRQC gains access to the traditional asymmetric public keys used in the TLS key exchange along with the transmitted ciphertext. TLS

implementations typically use Diffie-Hellman-based key exchange schemes. If an attacker obtains a complete set of encrypted payloads, including the TLS setup, they could theoretically use a CRQC to derive the private key and decrypt the data.

The primary concern for data confidentiality is the "Harvest Now, Decrypt Later" scenario, where a malicious actor with sufficient resources stores encrypted data today to decrypt it in the future, once a CRQC becomes available. This means that even data encrypted today is at risk unless quantum-safe strategies are implemented. The window of vulnerability—the effective security lifetime of the encrypted data—can range from seconds to decades, depending on the sensitivity of the data and how long it remains valuable. This highlights the immediate need to adopt quantum-resistant cryptographic measures to ensure long-term confidentiality.

For data authentication, the concern shifts to potential on-path attackers equipped with CRQCs capable of breaking certificate-based authentication mechanisms that rely on traditional algorithms. Such attackers could impersonate legitimate entities, tricking victims into connecting to the attacker’s device instead of the intended target, resulting in impersonation attacks. While this is not as immediate a threat as "Harvest Now, Decrypt Later" attacks, it remains a significant risk that must be addressed proactively.

In client/server certificate-based authentication, the security window between the generation of the signature in the CertificateVerify message and its verification by the peer during the TLS handshake is typically short. However, the security lifetime of digital signatures on X.509 certificates, including those issued by root Certification Authorities (CAs), warrants closer scrutiny. Root CA certificates can have validity periods of 20 years or more, while root Certificate Revocation Lists (CRLs) often remain valid for a year or longer. Delegated credentials, such as CRL Signing Certificates or OCSP response signing certificates, generally have shorter lifetimes but still present a potential vulnerability window.

While data confidentiality faces the immediate and pressing threat of "Harvest Now, Decrypt Later" attacks, requiring urgent quantum-safe adoption, data authentication poses a longer-term risk that still necessitates careful planning. Both scenarios underscore the importance of transitioning to quantum-resistant cryptographic systems to safeguard data and authentication mechanisms in a post-quantum era.

4. Data Confidentiality

As explained in the previous section, data that is only temporarily in transit may nevertheless require protection for many years. However, uncertainties regarding the security of PQC algorithm implementations, evolving regulatory requirements, and the ongoing development of cryptanalysis justify a transitional approach where well-established traditional algorithms are used alongside new PQC primitives.

Applications utilizing (D)TLS that are vulnerable to "Harvest Now, Decrypt Later" attacks MUST transition to (D)TLS 1.3 and adopt one of the following strategies:

* Hybrid Key Exchange: Hybrid key exchange combines traditional and PQC key exchange algorithms, offering resilience even if one algorithm is compromised. As defined in [I-D.ietf-tls-hybrid-design], this approach ensures robust security during the migration to PQC. For TLS 1.3, hybrid Post-Quantum key exchange groups are introduced in [I-D.ietf-tls-ecdhe-mlkem]:

    1. X25519MLKEM768: Combines the classical X25519 key exchange with the ML-KEM-768 Post-Quantum Key Encapsulation Mechanism.

    2. SecP256r1MLKEM768: Combines the classical SecP256r1 key exchange with the ML-KEM-768 Post-Quantum Key Encapsulation Mechanism.

    3. SecP384r1MLKEM1024: Combines the classical SecP384r1 key exchange with the ML-KEM-1024 Post-Quantum Key Encapsulation Mechanism.

* Pure Post-Quantum Key Exchange: For deployments that require exclusively Post-Quantum key exchange, [I-D.ietf-tls-mlkem-key-agreement] defines the following standalone NamedGroups for Post-Quantum key agreement in TLS 1.3: ML-KEM-512, ML-KEM-768, and ML-KEM-1024.

Hybrid Key Exchange is generally preferred over pure PQC key exchange because it provides defense-in-depth by combining the strengths of both classical and PQC algorithms. This ensures continued security, even if one algorithm is compromised during the transitional period.

However, Pure PQC Key Exchange may be required for specific deployments with regulatory or compliance mandates that necessitate the exclusive use of post-quantum cryptography. Examples include sectors governed by stringent cryptographic standards.

In practice, applications that rely on TLS typically depend on the underlying TLS library. Upgrading to a library version that supports TLS 1.3 and PQC key exchange extensions is a necessary first step, but it may not be sufficient, as it is not known whether PQC groups are enabled by default across different implementations. Applications that configure protocol versions or cipher suites explicitly MUST update these settings to ensure that hybrid or pure PQC key exchange groups are enabled. Applications that rely on library defaults SHOULD review the library documentation or perform interoperability testing to confirm that PQC groups are negotiated as intended. Operators should also consider potential interoperability issues with legacy peers that do not yet support TLS 1.3 and PQC key exchange extensions.

4.1. Optimizing ClientHello for Hybrid Key Exchange in TLS Handshake

The client initiates the TLS handshake by sending a list of supported key agreement methods in the key_share extension. One of the important challenges during the migration to PQC is that the client may not know whether the server supports hybrid key exchange. To address this uncertainty, the client can adopt one of the following three strategies:

1. Send Both Traditional and Hybrid Key Exchange Algorithms: In the initial ClientHello message, the client can include both traditional and hybrid key exchange algorithm key shares. This eliminates the need for multiple round trips but comes with its own trade-offs.

* Advantage: Reduces latency since the server can immediately select an appropriate key exchange method.

* Challenges:

  - The size of the hybrid key exchange algorithm key share may exceed the Maximum Transmission Unit (MTU), potentially causing the ClientHello message to be fragmented across multiple packets in both TLS and DTLS. This fragmentation increases the risk of packet loss and retransmissions, leading to potential delays. During the TLS handshake, the server will respond to the ClientHello with its public key and ciphertext. If these components also exceed the MTU, the ServerHello message may be fragmented, further compounding the risk of delays due to packet loss and retransmissions.

- Middleboxes that do not handle fragmented ClientHello messages properly may drop them, as this behavior is uncommon. More generally, middleboxes may also mishandle fragmented IP/UDP packets, which makes this issue particularly significant for DTLS deployments.

- Additionally, this approach requires more computational resources on the client and increases handshake traffic.

1. Indicate Support for Hybrid Key Exchange: Alternatively, the client may initially indicate support for hybrid key exchange and send a traditional key exchange algorithm key share in the first ClientHello message. If the server supports hybrid key exchange, it will use the HelloRetryRequest to request a hybrid key exchange algorithm key share from the client. The client can then send the hybrid key exchange algorithm key share in the second ClientHello message. However, this approach has a disadvantage in that the roundtrip would introduce additional delay compared to the previous technique of sending both traditional and hybrid key exchange algorithm key shares to the server in the initial ClientHello message.

2. Use Server Key Share Preferences Communicated via DNS: [I-D.ietf-tls-key-share-prediction] defines a mechanism where servers communicate their key share preferences through DNS responses. TLS clients can use this information to tailor their initial ClientHello message, reducing the need for additional round trips. By leveraging these DNS-based hints, the client can optimize the handshake process and avoid unnecessary delays.

Clients MAY also use information from completed handshakes to cache the server's key exchange algorithm preferences, as described in Section 4.2.7 of [RFC8446]. To minimize the risk of the ClientHello message being split across multiple packets, clients should avoid duplicating PQC KEM public key shares. Strategies for preventing duplication are outlined in Section 4 of [I-D.ietf-tls-hybrid-design]. By carefully managing key shares, the client can reduce the size of the ClientHello message and improve compatibility with network infrastructure.

5. Use of External PSK with Traditional Key Exchange for Data Confidentiality

[RFC8772] provides an alternative approach for ensuring data confidentiality by combining an external pre-shared key (PSK) with a traditional key exchange mechanism, such as ECDHE. The external PSK is incorporated into the TLS 1.3 key schedule, where it is mixed with the (EC)DHE-derived secret to strengthen confidentiality.

While using an external PSK in combination with (EC)DHE can enhance confidentiality, it has the following limitations:

* Key Management Complexity: Unlike ephemeral ECDHE keys, external PSKs require secure provisioning and lifecycle management.

* Limited Forward Secrecy: If an external PSK is static and reused across sessions, its compromise can retroactively expose past communications if the traditional key exchange is broken by a CRQC.

* Scalability Challenges: Establishing unique PSKs for many clients can be impractical, especially in large-scale deployments.

* Impersonation Risk: Because PSKs are symmetric, any party in possession of the PSK can authenticate as either the client or the server. This differs from certificate-based authentication, where compromise of a private key only enables impersonation of the corresponding entity.

* Quantum Resistance Dependence: While PSKs can provide additional secrecy against quantum threats, they must be generated using a secure key-management technique. If a weak PSK is used, it may not offer sufficient security against brute-force attacks.

Despite these limitations, external PSKs can serve as a complementary mechanism in PQC transition strategies, providing additional confidentiality protection when combined with traditional key exchange.

6. Authentication

Although CRQCs could potentially decrypt past TLS sessions, client/server authentication based on certificates cannot be retroactively compromised. However, the multi-year process required to establish, certify, and embed new root CAs presents a significant challenge. If CRQCs emerge earlier than anticipated, responding promptly to secure authentication systems would be difficult. While the migration to PQ X.509 certificates allows for more time compared to key exchanges, delaying these preparations should be avoided.

6.1. Quantum-Ready Authentication

The quantum-ready authentication property becomes critical in scenarios where an on-path attacker uses network devices equipped with CRQCs to break traditional authentication protocols. For example, if an attacker determines the private key of a server certificate before its expiration, they could impersonate the server, causing users to believe their connections are legitimate. This impersonation leads to serious security threats, including unauthorized data disclosure, interception of communications, and overall system compromise.

The quantum-ready authentication property ensures robust authentication through the use of either a pure Post-Quantum certificate or a PQ/T hybrid certificate:

6.2. Post-Quantum X.509 Certificates

Post-quantum certificates contain only a PQC public key and are signed using a post-quantum algorithm. They are suitable for deployments capable of fully embracing post-quantum cryptography.

* ML-DSA Certificates: Defined in [I-D.ietf-lamps-dilithium-certificates], these use the Module-Lattice Digital Signature Algorithm (ML-DSA). [I-D.tls-westerbaan-mldsa] explains how ML-DSA is applied for authentication in TLS 1.3.

* SLH-DSA Certificates: Defined in [I-D.ietf-lamps-x509-slhdsa], these use the SLH-DSA algorithm. [I-D.reddy-tls-slhdsa] details how SLH-DSA is used in TLS 1.3 and compares its advantages and disadvantages with ML-DSA in Section 2 of the document.

6.3. Hybrid (Composite) X.509 Certificates

A composite certificate contains both a traditional public key algorithm (e.g., ECDSA) and a post-quantum algorithm (e.g., ML-DSA) within a single X.509 certificate. This design enables both algorithms to be used in parallel, the traditional component ensures compatibility with existing infrastructure, while the post-quantum component introduces resistance against future quantum attacks.

Composite certificates are defined in [I-D.ietf-lamps-pq-composite-sigs]. These combine Post-Quantum algorithms like ML-DSA with traditional algorithms such as RSA-PKCS#1v1.5, RSA-PSS, ECDSA, Ed25519, or Ed448, to provide additional protection against vulnerabilities or implementation bugs in a single algorithm. [I-D.reddy-tls-composite-mldsa] specifies how composite signatures, including ML-DSA, are used for TLS 1.3 authentication.

6.4. Transition Considerations

Determining whether and when to adopt PQC certificates or PQ/T hybrid schemes depends on several factors, including:

* Frequency and duration of system upgrades
* The expected timeline for CRQC availability
* Operational flexibility to enable or disable algorithms

Deployments with limited flexibility benefit significantly from hybrid signatures, which combine traditional algorithms with PQC algorithms. This approach mitigates the risks associated with delays in transitioning to PQC and provides an immediate safeguard against zero-day vulnerabilities.

Composite certificates enhance resilience during the adoption of PQC by:

* Providing defense-in-depth: They maintain security even if one algorithm is compromised.
* Reducing exposure to unforeseen vulnerabilities: They offer immediate protection against potential weaknesses in PQC algorithms.

However, composite certificates comes with long-term implications. Once the traditional algorithm is no longer considered secure, due to CRQCs, it will have to be deprecated. To complete the transition to a fully quantum-resistant authentication model, it will be necessary to provision a new root CA certificate, that uses only a PQC signature algorithm and public key. This new root CA would issue a hierarchy of intermediate certificates, each also signed using a PQC algorithm and ultimately issue end-entity certificates that likewise contain only PQC public keys and are signed with PQC algorithms. This ensures that the entire certification path from the root of trust to the end-entity is cryptographically resistant to quantum attacks and does not depend on any traditional algorithms.

Alternatively, a deployment may choose to continue using the same hybrid certificate even after the traditional algorithm has been broken by the advent of a CRQC. While this may simplify operations by avoiding re-provisioning of trust anchors, it introduces a significant risk: the composite signature will no longer achieve Strong Unforgeability (SUF) (Section 10.2 of [I-D.ietf-pquip-pqc-engineers]), as explained in Section 11.2 of [I-D.reddy-tls-composite-mldsa].

In this scenario, a CRQC can forge the broken traditional signature component (s1_) over a message (m). That forged component can then be combined with the valid post-quantum component (s2) to produce a new composite signature (m, (s1_, s2)) that verifies successfully, thereby violating SUF. This highlights the critical need to retire hybrid certificates containing broken algorithms once CRQCs are available.

6.5. Deployment Realities

Centralized networks, which are characterized by strong administrative control, internal CAs, and close relationships with vendors, are generally well-positioned to manage the overhead of larger PQC keys and signatures. Such networks can adopt PQC signature algorithms earlier due to their ability to coordinate and deploy changes effectively. For example, telecom networks fit this model and may be able to transition more quickly than more distributed environments.

Conversely, the Web PKI ecosystem may delay adoption until more efficient and compact PQC signature algorithms, such as MAYO, UOV, HAWK, or SQISign, become available. This is due to the broader, more decentralized nature of the Web PKI ecosystem, which makes coordination and implementation more challenging.

6.6. Optimizing PQC Certificate Exchange in TLS

To address the challenge of large PQ or PQ/T hybrid certificate chains during the TLS handshake, the following mechanisms can help optimize the size of the exchanged certificate data:

* TLS Cached Information Extension ([RFC7924]): This extension enables clients to indicate that they have cached certificate information from a prior connection. The server can then signal the client to reuse the cached data instead of retransmitting the full certificate chain. While this mechanism reduces bandwidth usage, it introduces potential privacy concerns: the client includes fingerprints of cached objects in the ClientHello, which are visible to eavesdroppers. These values can be used to

correlate independent TLS sessions from the same client, potentially compromising anonymity. While this is not a concern for many industrial IoT scenarios, it may be inacceptable to smart home deployments.

* TLS Certificate Compression ([RFC8879]): This specification defines compression schemes to reduce the size of the server's certificate chain. While effective in many scenarios, its impact on PQ or PQ/T hybrid certificates is limited due to the larger sizes of public keys and signatures in PQC. These high-entropy fields, inherent to PQC algorithms, constrain the overall compression effectiveness.

* Abridged TLS Certificate ({?I-D.ietf-tls-cert-abridge}): This approach minimizes the size of the certificate chain by omitting intermediate certificates that are already known to the client. Instead, the server provides a compact representation of the certificate chain, and the client reconstructs the omitted certificates using a well-known common CA database. This mechanism significantly reduces bandwidth requirements while preserving compatibility with existing certificate validation processes. Additionally, it explores potential methods to compress the end-entity certificate itself, though this aspect remains under discussion within the TLS Working Group.

* Trust Anchor Identifiers ({?I-D.ietf-tls-trust-anchor-ids}): This extension allows a client to signal a compact list of trusted root CAs using unique trust anchor identifiers rather than Distinguished Names. This reduces the size of the "certificate_authorities" extension and helps the server select an appropriate certificate chain, especially when multiple hierarchies are used (e.g., separate traditional and PQ roots). This mechanism can help reduce handshake size and improve efficiency in hybrid or PQC deployments.

These techniques aim to optimize the exchange of certificate chains during the TLS handshake, particularly in scenarios involving large PQC-related certificates, while balancing efficiency and compatibility.

7. Informing Users of PQC Security Compatibility Issues

When the server detects that the client does not support PQC or hybrid key exchange, it may send an insufficient_security fatal alert to the client. The client, in turn, can notify service providers via device management systems or generate logs indicating that the server they are attempting to access requires a level of security that the client cannot provide due to the lack of PQC support. Additionally,

the client may log this event for diagnostic purposes, security auditing, or reporting the issue to the application developments for further analysis.

Conversely, if the client detects that the server does not support PQC or hybrid key exchange, it may present an alert or error message to the end-user or record the event in diagnostic logs. This message or record should explain that the server is incompatible with the PQC security features supported by the client.

It is important to design such alerts thoughtfully to ensure they are clear and actionable, avoiding unnecessary warnings that could overwhelm or confuse users. In some environments, such as EAP deployments, supplicants may provide little or no diagnostic feedback to end-users beyond a generic failure message. In such cases, implementers would have to ensure sufficient diagnostic logging or telemetry is available for administrators to diagnose PQC-related interoperability problems. Notifications to end-users may also not be applicable or necessary in all scenarios, particularly in the context of machine-to-machine communication.

8. PQC Transition for Critical Application Protocols

This document primarily focuses on the transition to PQC in applications that utilize TLS, while also covering other essential protocols, such as DNS, that play a critical role in supporting application functionality.

8.1. Encrypted DNS

The privacy risks associated with exchanging DNS messages in clear text are detailed in [RFC9076]. To mitigate these risks, Transport Layer Security (TLS) is employed to provide privacy for DNS communications. Encrypted DNS protocols, such as DNS-over-HTTPS (DoH) [RFC8484], DNS-over-TLS (DoT) [RFC7858], and DNS-over-QUIC (DoQ) [RFC9250], safeguard messages against eavesdropping and on-path tampering during transit.

However, encrypted DNS messages transmitted using TLS may be vulnerable to HNDL attacks if an attacker gains access to the public keys used in the TLS key exchange. If an attacker records a complete set of encrypted DNS messages, including the TLS handshake details, they could store this data today and later use a CRQC to determine the ephemeral private key used in the key exchange, thereby decrypting the content.

To address these vulnerabilities, encrypted DNS protocols MUST support the quantum-ready usage profile discussed in {#confident}.

It is important to note that the Post-Quantum security of DNSSEC [RFC9364], which provides authenticity for DNS records, is a distinct issue separate from the requirements for encrypted DNS transport protocols.

8.2. Hybrid public-key encryption (HPKE) and Encrypted Client Hello

Hybrid Public-Key Encryption (HPKE) is a cryptographic scheme designed to enable public key encryption of arbitrary-sized plaintexts using a recipient's public key. HPKE employs a non-interactive ephemeral-static Diffie-Hellman key exchange to derive a shared secret. The rationale for standardizing a public key encryption scheme is detailed in the introduction of [RFC9180].

HPKE can be extended to support both pure PQC KEMs and PQ/T hybrid KEMs, as described in [I-D.ietf-hpke-pq]. These extensions ensure compatibility with PQC, while allowing deployments to choose between pure PQC KEM or PQ/T KEM.

Client TLS libraries and applications can utilize Encrypted Client Hello (ECH) [I-D.ietf-tls-esni] to prevent passive observation of the intended server identity during the TLS handshake. However, this requires the concurrent deployment of Encrypted DNS protocols (e.g., DNS-over-TLS), as passive listeners could otherwise observe DNS queries or responses and deduce the same server identity that ECH is designed to protect. ECH employs HPKE for public key encryption.

To safeguard against "Harvest Now, Decrypt Later" attacks, ECH deployments must incorporate support for PQ/T Hybrid Post-Quantum KEMs. In this context, the public_key field in the HpkeKeyConfig structure would need to accommodate a concatenation of traditional and PQC KEM public keys to ensure robust protection against quantum-enabled adversaries.

To safeguard against HNDL attacks, ECH deployments MUST incorporate support for either pure PQC KEM or PQ/T hybrid KEM. PQ/T hybrid KEM is generally preferred, as it provides defense-in-depth by combining the strengths of both classical and PQC algorithms, ensuring continued security even if one is later found to be weak. Pure PQ KEMs may be required for deployments subject to regulatory or compliance mandates that necessitate the exclusive use of PQC. In hybrid mode, the public_key field in the HpkeKeyConfig structure accommodates a concatenation of classical and PQC KEM public keys, whereas in pure PQ mode only the PQC KEM public key is included.

9. Operational Considerations

The adoption of PQC in TLS-based applications will not be a simple binary decision but rather a gradual transition that demands a careful evaluation of trade-offs and deployment considerations. Application providers will need to assess algorithm selection, performance impact, interoperability, and security requirements tailored to their specific use cases. While the IETF defines cryptographic mechanisms for TLS and provides guidance on PQC transition strategies, it does not prescribe a one-size-fits-all approach. Instead, this document outlines key considerations to assist stakeholders in adopting PQC in a way that aligns with their operational and security requirements.

10. Security Considerations

The security considerations outlined in [I-D.ietf-pquip-pqc-engineers] must be carefully evaluated and taken into account.

Post-quantum algorithms selected for standardization are relatively new, and their implementations are still in the early stages of maturity. This makes them more susceptible to implementation bugs compared to the well-established and extensively tested cryptographic algorithms currently in use. Furthermore, certain deployments may need to continue using traditional algorithms to meet regulatory requirements, such as Federal Information Processing Standard (FIPS) [SP-800-56C] or Payment Card Industry (PCI) compliance.

Hybrid key exchange provides a practical and flexible solution, offering protection against "Harvest Now, Decrypt Later" attacks while ensuring resilience to potential catastrophic vulnerabilities in any single algorithm. This approach allows for a gradual transition to PQC, preserving the benefits of traditional cryptosystems without requiring their immediate replacement.

10.1. MITM Attacks with CRQC

A MITM attack is possible if an adversary possesses a CRQC capable of breaking traditional public-key signatures. The attacker can generate a forged certificate and create a valid signature, enabling them to impersonate a TLS peer, whether a server or a client. This completely undermines the authentication guarantees of TLS when relying on traditional certificates.

To mitigate such attacks, several steps need to be taken:

1. Revocation and Transition: Both clients and servers that use traditional certificates will have to revoke them and migrate to PQC authentication.

2. Client-Side Verification: Clients should avoid establishing TLS sessions with servers that do not support PQC authentication.

3. PKI Migration: Organizations should transition their PKI to post-quantum-safe certification authorities and discontinue issuing certificates based on traditional cryptographic methods.

Acknowledgements

Thanks to Dan Wing for suggesting a broader scope for the document, and to Mike Ounsworth, Scott Fluhrer, Russ Housley, Loganaden Velvindron, Bas Westerbaan, Richard Sohn, Andrei Popov, Alan DeKok, and Thom Wiggers for their helpful feedback and reviews.

References

Normative References

[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-07, 7 July 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-lamps-pq-composite-sigs-07>.

[I-D.ietf-lamps-x509-slhdsa]
Bashiri, K., Fluhrer, S., Gazdag, S., Van Geest, D., and S. Kousidis, "Internet X.509 Public Key Infrastructure: Algorithm Identifiers for SLH-DSA", Work in Progress, Internet-Draft, draft-ietf-lamps-x509-slhdsa-09, 30 June 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-lamps-x509-slhdsa-09>.

[I-D.ietf-tls-ecdhe-mlkem]
Kwiatkowski, K., Kampanakis, P., Westerbaan, B., and D. Stebila, "Post-quantum hybrid ECDHE-MLKEM Key Agreement for TLSv1.3", Work in Progress, Internet-Draft, draft-ietf-tls-ecdhe-mlkem-00, 23 March 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-tls-ecdhe-mlkem-00>.

[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-16, 7 September 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-tls-hybrid-design-16>.

[I-D.ietf-tls-key-share-prediction]
Benjamin, D., "TLS Key Share Prediction", Work in Progress, Internet-Draft, draft-ietf-tls-key-share-prediction-03, 29 August 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-tls-key-share-prediction-03>.

[I-D.ietf-tls-mlkem-key-agreement]
"*** BROKEN REFERENCE ***".

[I-D.reddy-tls-composite-mldsa]
Reddy.K, T., Hollebeek, T., Gray, J., and S. Fluhrer, "Use of Composite ML-DSA in TLS 1.3", Work in Progress, Internet-Draft, draft-reddy-tls-composite-mldsa-05, 4 July 2025, <https://datatracker.ietf.org/doc/html/draft-reddy-tls-composite-mldsa-05>.

[I-D.reddy-tls-slhdsa]
Reddy.K, T., Hollebeek, T., Gray, J., and S. Fluhrer, "Use of SLH-DSA in TLS 1.3", Work in Progress, Internet-Draft, draft-reddy-tls-slhdsa-01, 13 April 2025, <https://datatracker.ietf.org/doc/html/draft-reddy-tls-slhdsa-01>.

[I-D.tls-westerbaan-mldsa]
Hollebeek, T., Schmieg, S., and B. Westerbaan, "Use of ML-DSA in TLS 1.3", Work in Progress, Internet-Draft, draft-tls-westerbaan-mldsa-00, 15 November 2024, <https://datatracker.ietf.org/doc/html/draft-tls-westerbaan-mldsa-00>.

[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>.

[RFC7858] Hu, Z., Zhu, L., Heidemann, J., Mankin, A., Wessels, D., and P. Hoffman, "Specification for DNS over Transport Layer Security (TLS)", RFC 7858, DOI 10.17487/RFC7858, May 2016, <https://www.rfc-editor.org/rfc/rfc7858>.

[RFC7924] Santesson, S. and H. Tschofenig, "Transport Layer Security (TLS) Cached Information Extension", RFC 7924, DOI 10.17487/RFC7924, July 2016, <https://www.rfc-editor.org/rfc/rfc7924>.

[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>.

[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>.

[RFC8484] Hoffman, P. and P. McManus, "DNS Queries over HTTPS (DoH)", RFC 8484, DOI 10.17487/RFC8484, October 2018, <https://www.rfc-editor.org/rfc/rfc8484>.

[RFC8772] Hu, S., Eastlake, D., Qin, F., Chua, T., and D. Huang, "The China Mobile, Huawei, and ZTE Broadband Network Gateway (BNG) Simple Control and User Plane Separation Protocol (S-CUSP)", RFC 8772, DOI 10.17487/RFC8772, May 2020, <https://www.rfc-editor.org/rfc/rfc8772>.

[RFC8879] Ghedini, A. and V. Vasiliev, "TLS Certificate Compression", RFC 8879, DOI 10.17487/RFC8879, December 2020, <https://www.rfc-editor.org/rfc/rfc8879>.

[RFC9250] Huitema, C., Dickinson, S., and A. Mankin, "DNS over Dedicated QUIC Connections", RFC 9250, DOI 10.17487/RFC9250, May 2022, <https://www.rfc-editor.org/rfc/rfc9250>.

Informative References

[I-D.ietf-hpke-pq]
Barnes, R., "Post-Quantum and Post-Quantum/Traditional Hybrid Algorithms for HPKE", Work in Progress, Internet-Draft, draft-ietf-hpke-pq-01, 30 June 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-hpke-pq-01>.

[I-D.ietf-pquip-pqc-engineers]
Banerjee, A., Reddy.K, T., Schoinianakis, D., Hollebeek, T., and M. Ounsworth, "Post-Quantum Cryptography for Engineers", Work in Progress, Internet-Draft, draft-ietf-pquip-pqc-engineers-14, 25 August 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-pquip-pqc-engineers-14>.

[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-esni]
Rescorla, E., Oku, K., Sullivan, N., and C. A. Wood, "TLS Encrypted Client Hello", Work in Progress, Internet-Draft, draft-ietf-tls-esni-25, 14 June 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-tls-esni-25>.

[RFC9001]
Thomson, M., Ed. and S. Turner, Ed., "Using TLS to Secure QUIC", RFC 9001, DOI 10.17487/RFC9001, May 2021, <https://www.rfc-editor.org/rfc/rfc9001>.

[RFC9076]
Wicinski, T., Ed., "DNS Privacy Considerations", RFC 9076, DOI 10.17487/RFC9076, July 2021, <https://www.rfc-editor.org/rfc/rfc9076>.

[RFC9147]
Rescorla, E., Tschofenig, H., and N. Modadugu, "The Datagram Transport Layer Security (DTLS) Protocol Version 1.3", RFC 9147, DOI 10.17487/RFC9147, April 2022, <https://www.rfc-editor.org/rfc/rfc9147>.

[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>.

[RFC9364] Hoffman, P., "DNS Security Extensions (DNSSEC)", BCP 237, RFC 9364, DOI 10.17487/RFC9364, February 2023, <https://www.rfc-editor.org/rfc/rfc9364>.

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

Authors' Addresses

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

Hannes Tschofenig
University of Applied Sciences Bonn-Rhein-Sieg
Germany
Email: Hannes.Tschofenig@gmx.net