Transport Layer Security
Internet-Draft
Intended status: Standards Track
Expires: 22 April 2026

Y. Sheffer
Intuit
T. Reddy
Nokia

19 October 2025

PQC Continuity: Downgrade Protection for TLS Servers Migrating to PQC
draft-sheffer-tls-pqc-continuity-00

Abstract

As the Internet transitions toward post-quantum cryptography (PQC), many TLS servers will continue supporting traditional certificates to maintain compatibility with legacy clients. However, this coexistence introduces a significant vulnerability: an undetected rollback attack, where a malicious actor strips the PQC or Composite certificate and forces the use of a traditional certificate once quantum-capable adversaries exist.

To defend against this, this document defines a TLS extension that allows a client to cache a server's declared commitment to present PQC or composite certificates for a specified duration. On subsequent connections, clients enforce that cached commitment and reject traditional-only certificates that conflict with it. This mechanism, inspired by HTTP Strict Transport Security (HSTS) but operating at the TLS layer provides PQC downgrade protection without requiring changes to certificate authority (CA) infrastructure.

About This Document

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

The latest revision of this draft can be found at https://yaronf.github.io/draft-sheffer-tls-pqc-continuity/draft-sheffer-tls-pqc-continuity.html. Status information for this document may be found at https://datatracker.ietf.org/doc/draft-sheffer-tls-pqc-continuity/.

Source for this draft and an issue tracker can be found at https://github.com/yaronf/draft-sheffer-tls-pqc-continuity.

Status of This Memo

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

This Internet-Draft will expire on 22 April 2026.

Copyright Notice

Table of Contents

1. Introduction

The migration to post-quantum cryptography (PQC) will be gradual. Servers will likely host both traditional and PQC (or composite) certificates to maintain compatibility: legacy clients can still connect, while updated ones benefit from PQC authentication. The size of the legacy client base often drives the decision to keep traditional certificates. Relevant PQC work includes [I-D.ietf-lamps-dilithium-certificates] (ML-DSA), [I-D.ietf-lamps-x509-slhdsa] (SLH-DSA), and [I-D.ietf-lamps-pq-composite-sigs] (composites). Not only must legacy clients be supported by servers for years, new clients that support PQC are also incented to accept traditional certificates, to retain connectivity to legacy servers.

Once a cryptographically relevant quantum computer (CRQC) emerges publicly, traditional certificates become insecure and must be revoked, regardless of legacy disruption. However, a CRQC may remain undisclosed, allowing attackers to exploit classical algorithms secretly. In such cases, adversaries could strip PQC or composite certificates, present only traditional ones, and conduct MitM attacks. Relying parties therefore need mechanisms to detect when servers claiming PQC support revert to traditional credentials only.

To prevent such downgrade attacks, this document defines a TLS extension that enables the TLS client to cache an indication that the server is able to present a (Composite or pure) PQ certificate, for some duration of time, e.g. one year. As a result:

* Clients reconnecting to an already known server within the validity period are protected from rollback to classic certificates.

* A client begins enforcing the server’s PQC commitment only after it has successfully connected to the legitimate server at least once (i.e., a connection not intercepted by a MiTM). Earlier connections that are intercepted or downgraded do not prevent the client from gaining protection once it later observes a PQC commitment from a legitimate server.

The explicitly communicated caching time allows clients to implement a caching policy with no risk of sudden breakage, and allows servers to revert to traditional certificates if they ever see the need to do so.

This extension is modeled on HSTS [RFC6797], but whereas HSTS is at the HTTP layer, the extension is implemented at the TLS layer.

On the open Web, we expect this extension to be used mainly for caching the fact that a server is presenting a PQC or composite certificate. However, in other use cases such as service-to-service traffic, it would often make sense to use it for both clients and servers.

An alternative approach to downgrade attacks, described in [I-D.reddy-lamps-x509-pq-commit], uses specially marked certificates to denote the server's long-term commitment to use PQC algorithms. See Appendix A for a comparison between the two approaches.

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. The pq_cert_available Extension

The following section defines a TLS extension that describes a TLS peer's commitment to present PQC credentials.

3.1. Extension Definition

This is a TLS extension, as per sec. 4.2 of [RFC8446]. The extension type for pq_cert_available is TBD by IANA. It MAY appear in the ClientHello (CH) message, CertificateRequest (CR) message and in Certificate (CT) messages sent by either client or server.

A supporting client MUST include this extension in its ClientHello message, with no extension data.

If the client indicates support, the server MAY include the extension in its Certificate message. For symmetry, the server MAY also send an empty pq_cert_available extension in the CertificateRequest to indicate support for this mechanism. A client MUST NOT include pq_cert_available in its Certificate message unless the server has first included the extension in a CertificateRequest message.

The extension data when sent in the Certificate message is:

```
struct {
    SignatureScheme signature_algorithm;
    uint32 algorithm_validity_period;
}
```

This extension follows the format of TLS 1.3 Certificate message extensions as defined in Sec. 4.4.2 of [RFC8446].

Note on terminology: Since the extension may be used by either client or server, the term "sender" is used for the peer that sent the extension in its Certificate message and "recipient" for the other peer.

The signature_algorithm in this extension MUST be the signature algorithm that the sender's end-entity certificate is associated with. SignatureScheme is defined by [RFC8446].

The algorithm_validity_period field is the time duration, in seconds, that the sender commits to continue to present a certificate that enables this signature scheme. The time duration is measured starting with the TLS handshake and is unrelated to any particular certificate or its lifecycle. A value of zero indicates no post-handshake commitment.

3.2. Algorithm Selection

If one of the peers holds unexpired cached information for the other peer:

* The peer SHOULD include the cached algorithm in the signature_algorithms extension of its ClientHello (or CertificateRequest for servers), and MUST NOT include legacy (non-PQC) algorithms.

* It MAY include other PQ signature algorithms, according to local policy.

As a result, the handshake would fail if a rollback attack is attempted.

3.3. Recipient Behavior

A recipient that supports this extension MUST behave as follows:

1. If the recipient holds no cached information for the sender, and the sender includes a non-empty extension:

    * The recipient SHOULD cache the provided information after the handshake is completed successfully and after the extension's data has been validated.

    * The recipient MAY choose to cache the signature algorithm for a shorter period than specified.

2. If the recipient holds unexpired cached information for the sender, and receives a returned extension from the sender:

   * The recipient should validate the signature_algorithm relative to the certificate being presented and SHOULD extend its cache period if the received time value would expire later than its current cache expiry.

   * It SHOULD NOT accept an algorithm_validity value if it would decrease its existing value (within a few seconds' tolerance).

   * It SHOULD replace its cached signature algorithm for the sender by a different PQ algorithm if such is sent in the extension, and in this case, it SHOULD use the validity time as-is.

3. If the recipient holds unexpired cached information for the sender, and receives no returned extension from the sender, the recipient SHOULD NOT modify its cache.

// OPEN ISSUE: do we discuss how the cache is indexed? Service
// identity per RFC 9525?

3.4. Sender Behavior

1. A TLS client or server that receives an indication that its peer supports this extension SHOULD send this extension in the Certificate message, provided a PQ signature algorithm is used.

2. The sender MUST keep track of the time duration it has committed to, and use a PQ certificate to authenticate itself for that entire duration. The sender MAY change its certificates and may switch between PQ signature algorithms at will, provided the peer indicates acceptance of these algorithms.

This obligation is analogous to maintaining HSTS continuity: once a commitment is made, the sender MUST avoid reverting to classical certificates until expiry of algorithm_validity.

3.5. Operational Considerations

This extension establishes a (potentially) long-term commitment of the sender to support PQ signature algorithms. As such, we recommend that deployers first experiment with short validity periods (e.g. one day), and only when satisfied that peers populate and depopulate their cache correctly, they can move to a longer duration. In the case of HSTS, lifetimes are commonly set to one year.

4. Security Considerations

TODO Security

5. IANA Considerations

IANA is requested to assign a new value from the “TLS ExtensionType Values”

Value	Extension Name	TLS 1.3	Recommended	Reference
TBD	pq_cert_available	CH, CR,CT	Y	This document

Table 1

6. Document History

RFC Editor: please remove before publication.

6.1. draft-sheffer-tls-pqc-continuity-00

Initial version.

Acknowledgments

TODO acknowledge.

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>.

[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>.

Informative 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-13, 30 September 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-lamps-dilithium-certificates-13>.

[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-12, 10 October 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-lamps-pq-composite-sigs-12>.

[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.reddy-lamps-x509-pq-commit]
Reddy.K, T., Gray, J., and Y. Sheffer, "X.509 Extensions for PQC or Composite Certificate Hosting Continuity", Work in Progress, Internet-Draft, draft-reddy-lamps-x509-pq-commit-00, 12 October 2025, <https://datatracker.ietf.org/doc/html/draft-reddy-lamps-x509-pq-commit-00>.

[RFC6797] Hodges, J., Jackson, C., and A. Barth, "HTTP Strict Transport Security (HSTS)", RFC 6797, DOI 10.17487/RFC6797, November 2012, <https://www.rfc-editor.org/rfc/rfc6797>.

Appendix A. Comparison with the Certificate-Based Solution

This section is a comparison with an analogous solution [I-D.reddy-lamps-x509-pq-commit] for the same rollback problem, one that signals server continuity using certificates rather than the TLS connection itself.

* The certificate-based solution does not change the TLS handshake, which potentially makes adoption easier. However, changes to the Web Public Key Infrastructure would also affect adoption.

* The certificate-based solution is independent of TLS and thus can be used by other protocols.

* Operationally, it may be harder to manage the “commitment” through certificates vs. TLS configuration. For example, in the HSTS space, it is common to experiment first with very short durations, e.g. 1 day, before moving to a longer commitment. This could have a significant effect on real-life adoption.

* The revocation checking aspect of the certificate-based solution relies upon other mechanisms (e.g. CRLs, OCSP) to also be signed with PQC/Composite. Those other RFCs and implementations are likely to take even longer to materialize.

Authors’ Addresses

Yaron Sheffer
Intuit
Email: yaronf.ietf@gmail.com

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