

PQC Continuity: Downgrade Protection for TLS Servers Migrating to PQC

TLS, IETF-125



[draft-sheffer-tls-pqc-continuity-01](#)

Yaron Sheffer and Tirumaleswar Reddy

TLS Rollback During PQC Migration

- Most clients will need to accept both traditional and PQC certs for a long time
 - An emerging or secret CRQC can perform silent attacks on traditional certs
 - Active attacker strips PQC cert → forces traditional-only → MitM
 - Client cannot detect the downgrade
-
- What we propose is a mitigation, not a complete fix
 - Potentially also a detection mechanism
 - TOFU plus timed commitment
 - Also true for extended HSTS and cert-based alternatives



TOFU

TLS Extension for Downgrade Protection

- Server declares: "I support PQC for next X months"
 - Client caches this commitment
 - Future connections enforce it—fail if server reverts to traditional-only
-
- Like HSTS, but at TLS layer
 - No PKI changes needed
 - Server certs
 - Client cert support still an open issue

Protocol Details: *pq_cert_available* Extension

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

- Extension Placement:
 - ClientHello: Empty (indicates support)
 - CertificateRequest: Empty (server indicates support)
 - Certificate: With data (declares commitment)
- Semantics:
 - signature_algorithm: PQC algorithm in use (pure or composite)
 - algorithm_validity_period: Commitment duration from this handshake
 - Zero value: Clear cache / no commitment
 - Empty extension: Indicates support

Client and Server Behavior

- Sender (has PQC cert):
 - Send extension in Certificate message with algorithm + validity period
 - Track commitment duration globally
 - Must not revert to traditional-only until expiry
 - May switch between PQC algorithms (if peer accepts)

- Recipient (caching peer):
 - First connection: Cache algorithm + expiry after validation
 - Subsequent connections with cache:
 - Include cached algorithm in *signature_algorithms*
 - MUST NOT include legacy algorithms → handshake fails on rollback
 - Extend cache if new expiry is later
 - Clear cache if validity period = 0

CDNs and Middleboxes

- CDNs (Content Delivery Networks):
 - Different CDN PoPs (points of presence) may present different cert chains for same origin
 - Requirement: Server **MUST NOT** commit to PQC unless all CDN endpoints use PQC
- Middleboxes:
 - Rule: Middleboxes **MUST NOT** send PQC commitments
 - Incentivized not to break traffic
 - Edge case: MB is PQC client but traditional server, server's commitment already in client's cache
 - Would cause breakage (same issue affects HSTS-based solutions)
- Recovery:
 - Server bricking is not catastrophic, because it's a **simple fix**: install PQC cert on the server
 - Recovers all clients except those behind misbehaving middleboxes
 - In other words, this is not comparable to HPKP

Backup: Compared to Alternatives

Compared to an Extended HSTS

- Similarities:
 - Trust-on-first-use model with commitment caching
 - Configurable validity periods
 - Middlebox breakage risk (malicious/breached)
 - Attacker controlling origin can "brick" the service
- HSTS advantage: base protocol is widely implemented, amenable to extension
- PQC Continuity Advantages:
 - TLS layer → works for all TLS protocols (not just HTTP)
 - No app changes → application doesn't need to know PQ algorithms
 - Smaller attack surface → immune to XSS and cache poisoning attacks
 - Clean separation → no layer mixing between app and crypto

Compared to a Certificate-Based Solution

- TLS Extension Approach (this draft):
 - ✓ No PKI changes needed
 - ✓ Easy operational experimentation (short validity periods)
 - ✓ Fast deployment (TLS config only)
 - ✗ TLS-specific (can't be used by other protocols)
- Certificate-Based Approach ([draft-reddy-lamps-x509-pq-commit](#)):
 - ✓ Protocol-independent (works beyond TLS)
 - ✓ No need to extend TLS (but cert validation logic changes)
 - ✗ Requires PKI infrastructure changes
 - ✗ Harder to experiment operationally
 - ✗ Depends on PQC-signed revocation mechanisms (CRLs, OCSP)