dispatch
Internet-Draft
Intended status: Standards Track
Expires: 13 November 2026

J. J. Mututi
Individual

12 May 2026

QUIC Identity Protocol
draft-mututi-quip-00

Abstract

This document specifies QUIP (QUIC Identity Protocol), a federated application protocol providing portable cryptographic identity, verifiable server trust, deterministic state synchronization, and built-in accountability. QUIP runs over QUIC (RFC 9000) and uses a strict deterministic CBOR profile (QUIP-CBOR). Three conformance profiles are defined: Basic (messaging), Documents (content addressing), and Media (real-time transport). The protocol enables users to own a portable NodeId independent of service providers, servers to authenticate via DANE+DNSSEC with deployable fallback modes, and peers to detect and throttle misbehavior through signed violation receipts propagated via gossip.

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 13 November 2026.

Copyright Notice

Table of Contents

1. Introduction 3
1.1. Minimal Working Example 4
1.2. Non-Goals 4
2. Terminology 5
3. Protocol States 5
3.1. State Definitions 5
3.2. State Transitions 6
3.3. Protocol Invariants 8
4. Architecture Overview 8
5. Identity Model 8
5.1. NodeId 8
5.2. Attestations 9
5.2.1. Subject Signature Computation 9
5.2.2. Issuer Signature Computation 9
5.3. Key Rotation 9
5.4. Genesis Discovery 10
5.5. Identity Trust Modes 10
6. Trust Model 11
6.1. Server Authentication Modes 11
6.2. Independent Witness Definition 11
6.3. DNS Downgrade Protection 12
6.4. Handshake Transcript Binding 12
6.5. Verified Mode Governance 13
7. Verified Mode PKI 13
7.1. Trust Anchor Format 13
7.2. Trust Anchor Distribution 13
7.3. Attestation Provider Certificate 14
7.4. Certificate Path Validation 14
7.5. Provider Revocation Freshness 15
7.6. DNS Trust Anchor Bootstrapping 15
8. Transport 15
8.1. Connection Establishment 15
8.2. Wire Format 17
8.3. Stream Semantics 17
8.4. Bootstrap Discovery 17
9. Message Envelope 18
9.1. Envelope Structure 18
9.2. Signature Computation (Domain Separation) 18
9.3. Error Messages 19
9.4. Rate Advisory Messages 19

9.5. Sequence Validation Rules 19
9.6. Message ID Construction 20
9.7. Timestamp Validation 20
10. Gossip and Consistency 20
10.1. Vector Clocks 20
10.2. Gossip Message Format 20
10.3. Resource Authority Model 21
10.4. Anti-Entropy Protocol 21
10.5. Conflict Resolution 22
10.6. Gossip TTL 22
11. Rate Limiting and Accountability 22
11.1. Sequence Numbers 22
11.2. Violation Receipts 23
12. Resource Limits 23
13. Conformance Profiles 24
14. Experimental Features 25
15. Security Considerations 25
15.1. Threat Model Summary 25
15.2. Replay Attacks 25
15.3. Downgrade Attacks 25
15.4. Canonicalization Attacks 26
16. Privacy Considerations 26
17. IANA Considerations 26
17.1. ALPN Protocol ID Registration 26
17.2. QUIP Error Codes Registry 26
17.3. CBOR Tag 65536 Registration 27
18. QUIP-CBOR Profile 28
18.1. Encoding Rules 28
19. Implementation Guidelines 28
19.1. State Persistence 29
19.2. Sequence State Loss Recovery 29
20. References 29
20.1. Normative References 29
20.2. Informative References 30
Author’s Address 30

1. Introduction

QUIP defines a federation-native protocol that ensures:

* Portable identity independent of domain providers

* Verifiable server authenticity without reliance solely on certification authorities

* Deterministic global state convergence

* Cryptographic accountability for misbehavior

QUIP operates directly over QUIC [RFC9000] and eliminates reliance on HTTP-based federation layers.

1.1. Minimal Working Example

The following example demonstrates a complete QUIP flow:

Alice (NodeId A) → a.com → b.com → Bob (NodeId B)

1. Identity Setup
   a.com issues primary attestation binding user@a.com to NodeId A
   Requires subject signature (proof of possession)

2. Sending a Message
   Alice signs message with private key A
   a.com forwards to b.com over QUIC + DANE
   b.com verifies signature + attestation + policies
   Bob receives message

3. Accountability
   If a.com misbehaves, b.com creates signed violation receipt
   Receipt gossips to federation
   Automatic throttling after independent witness threshold

1.2. Non-Goals

QUIP does NOT guarantee:

* _Anonymity_ (use Tor or similar) — a design choice

* _Metadata confidentiality_ — connection patterns are observable

* _Resistance to global passive adversaries_ — traffic analysis remains possible

* _Byzantine consensus_ — deterministic merge is not Byzantine fault tolerant

* _Economic Sybil resistance_ — independent witness definition provides operational heuristic only

* _Witness collusion resistance_ — a federation with >=3 colluding witnesses can falsely attest violations; deployments must rely on legal/commercial incentives

Note on categorization: The items above mix design choices (anonymity, metadata confidentiality) with explicit security limitations (collusion resistance, Byzantine consensus). All are intentional non-goals of the protocol.

2. Terminology

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.

NodeId 32-byte Ed25519 public key (raw bytes)

Genesis NodeId Root identity in a rotation chain; the oldest known NodeId

Attestation Signed CBOR object binding identity claims

Primary attestation Attestation binding a specific user@domain address to a NodeId

Vector clock Map of issuer → counter, used for causality tracking

Gossip Propagation mechanism for state updates

Violation receipt Signed proof of protocol violation

Profile Conformance level (Basic, Documents, Media)

3. Protocol States

3.1. State Definitions

QUIP connections SHALL be in one of the following states:

INIT Initial state before discovery. No messages sent.

DISCOVERING Performing DNS discovery. MAY send DNS queries.

CONNECTING Establishing QUIC connection. MAY send QUIC handshake.

HANDSHAKING Exchanging QUIP handshake. Control stream active; no application streams.

SYNCING Synchronizing gossip state. Accept gossip; MAY defer message delivery.

ACTIVE Normal operation. All streams fully functional.

DEGRADED Operating with reduced capabilities. Rate limits increased; new streams discouraged.

REVOKED Identity or peer revoked. No messages accepted or sent.

CLOSED Connection terminated. No further state transitions.

3.2. State Transitions

The following transitions are normative:

From	To	Trigger	Required Action
INIT	DISCOVERING	Application requests connection	Perform DNS SRV/TXT lookup
DISCOVERING	CONNECTING	DNS resolution succeeds	Initiate QUIC handshake
CONNECTING	HANDSHAKING	QUIC connection established	Send QUIP handshake on stream 0
HANDSHAKING	SYNCING	Handshake complete, profiles compatible	Begin gossip sync on stream 1
HANDSHAKING	CLOSED	Profile mismatch or handshake error	Send error, close connection
SYNCING	ACTIVE	Initial sync complete	Normal operation begins
SYNCING	CLOSED	Sync timeout or unrecoverable error	Send error, close connection
ACTIVE	DEGRADED	Resource exhaustion or policy violation	Reduce functionality
ACTIVE	REVOKED	Receipt of valid revocation attesting this peer	Terminate all active streams
ACTIVE	CLOSED	Connection loss or explicit close	Clean up state
REVOKED	CLOSED	Any	Final cleanup
DEGRADED	ACTIVE	Resource recovery	Resume normal operation

Table 1

State transitions not listed above are not permitted and MUST cause a connection error (Section 17.2 code 17: PROTOCOL_VIOLATION).

3.3. Protocol Invariants

Implementations MUST enforce:

1. _Determinism_: All state transitions (Section 10) MUST produce identical results across peers
2. _Signature Coverage_: All externally visible objects MUST be signed with domain separation (Section 9.2)
3. _Identity Continuity_: Identity MUST be traceable to a genesis NodeId through a verifiable rotation chain
4. _Canonical Encoding_: All serialized data MUST use QUIP-CBOR (Section 18)
5. _No Duplicate Keys_: CBOR maps MUST NOT contain duplicate keys
6. _No Indefinite Lengths_: Indefinite-length byte strings and text strings are PROHIBITED

4. Architecture Overview

Layered model (bottom to top):

1. QUIC transport (TLS 1.3) — encryption, multiplexing, congestion control
2. Identity and attestation layer — NodeIds, bindings, trust models
3. Deterministic state machine — canonical CBOR, vector clocks, gossip
4. Profile layer — Basic / Documents / Media feature activation
5. Application layer — messaging, media, B2B, documents

5. Identity Model

5.1. NodeId

Every user has a NodeId — an Ed25519 public key (32 bytes). The corresponding private key is never disclosed to any domain or third party. NodeIds are encoded as raw byte strings in CBOR.

5.2. Attestations

All attestations are signed CBOR objects.

5.2.1. Subject Signature Computation

The subject signs the following QUIP-CBOR array:

```
[
  "QUIP-ATTESTATION-SUBJECT-V1",
  type,                    ; string
  subject,                 ; NodeId or address
  issuer,                  ; domain or NodeId
  [notBefore, notAfter]    ; Unix timestamps (seconds)
]
```

5.2.2. Issuer Signature Computation

The issuer signs the following QUIP-CBOR array:

```
[
  "QUIP-ATTESTATION-ISSUER-V1",
  type,
  subject,
  issuer,
  [notBefore, notAfter],
  subject_signature        ; 64 bytes
]
```

The complete attestation on the wire is:

```
tagged(65536, [
  "primary_binding",       ; discriminator
  type,
  subject,
  issuer,
  [notBefore, notAfter],
  subject_signature,
  issuer_signature
])
```

5.3. Key Rotation

```
tagged(65536, [
  "key_rotation",
  old_NodeId,              ; 32 bytes
  new_NodeId,              ; 32 bytes
  timestamp,               ; Unix milliseconds (int64)
  signature                ; 64 bytes
])
```

Signature computation:

```
signature = Ed25519_sign(
  old_private_key,
  "QUIP-ROTATION-V1" || QUIP-CBOR([
    "key_rotation",
    old_NodeId,
    new_NodeId,
    timestamp
  ])
)
```

Rotation validation rules: Receivers MUST reject:

* Cyclic chains (NodeId appears twice)

* Duplicate NodeIds in the same chain

* Chains with non-monotonic timestamps (timestamps must increase)

* Multiple successors from the same NodeId unless the previous successor is revoked

* Rotations where the old key is already revoked

* Chains exceeding maximum depth of 32 (configurable)

5.4. Genesis Discovery

The Genesis NodeId is the oldest known NodeId in a rotation chain.

First message rule: If a receiver encounters a NodeId with no prior state, and the message is signed by a key that is not clearly the genesis, the sender MUST include the full rotation chain. Failure results in error KEY_ROTATION_CHAIN_MISSING (Section 17.2 code 15).

5.5. Identity Trust Modes

The identity_trust_mode bitmask is defined as:

Bit	Name	Description
0	SIMPLE	Accept primary attestations from any domain
1	VERIFIED	Require attestations from certified providers (Section 7)
2-15	Reserved	Reserved for future use

Table 2

Negotiation: The effective trust mode is the bitwise AND of local and remote modes. If the result has no bits set, the connection proceeds with the lowest common mode (SIMPLE) and MUST log a warning.

Downgrade behavior: VERIFIED mode MUST NOT downgrade to SIMPLE without explicit operator configuration. Any downgrade MUST be logged.

6. Trust Model

6.1. Server Authentication Modes

Mode	Requirement	Use Case
COMPAT (default)	Web PKI + SHOULD use DANE if available	General deployment
STRICT (optional)	DNSSEC + DANE REQUIRED	High-security environments
LEGACY (deprecated)	Web PKI only	Legacy transition

Table 3

6.2. Independent Witness Definition

A witness domain qualifies as independent for violation receipt validation if ALL conditions hold:

1. Distinct DNSSEC-signed domain ([RFC4033], [RFC4034], [RFC4035])

2. Distinct TLSA record [RFC6698]

3. Distinct /24 IPv4 or /48 IPv6 prefix

Additional weighting (SHOULD): Implementations SHOULD also consider distinct ASN (Autonomous System Number), distinct DNSSEC KSK (Key Signing Key), historical trust scoring based on past witness accuracy, and observed federation longevity (minimum 30 days active).

6.3. DNS Downgrade Protection

After TLS handshake completion, the client MUST verify that the server's presented certificate and DANE TLSA record (if applicable) are consistent with the TXT capabilities advertisement from DNS. Inconsistencies MUST be logged and, in STRICT mode, cause connection termination.

6.4. Handshake Transcript Binding

The QUIP handshake MUST be cryptographically bound to the QUIC TLS session using the TLS exporter interface [RFC8446], Section 7.5.

Transcript canonicalization:

The client and server handshake payloads are embedded as CBOR byte strings containing the exact transmitted bytes:

```
[
  h'<raw client handshake bytes>',
  h'<raw server handshake bytes>'
]
```

Sequence of operations:

1. TLS handshake completes (both peers have established session keys)

2. QUIP handshake bytes are transmitted on stream 0 (client first, then server)

3. Both peers buffer the exact bytes they send and receive (no re-encoding)

4. After both handshake messages have been exchanged, each peer computes:

```
binding = TLS-Exporter("EXPORTER-QUIP-BINDING",
    SHA-256(QUIP-CBOR(handshake_array)), 32)
```

5. Because both peers use the same TLS session and the same canonicalized handshake array, they derive _identical binding values_ locally

6. The binding is used as a local consistency check: each peer can verify that the handshake parameters it is using are consistent with the binding it computed

7. No binding value is ever transmitted

6.5. Verified Mode Governance

Governance and policy requirements for VERIFIED mode (trust anchor authorization, provider certification criteria, dispute resolution, and antitrust/competition safeguards) are deployment-specific and outside the scope of this document. Implementations in VERIFIED mode MUST rely on local policy to define acceptable trust anchors and certification authorities.

7. Verified Mode PKI

7.1. Trust Anchor Format

Trust anchors are distributed as signed CBOR objects:

```
tagged(65536, [
  "trust_anchor",
  anchor_id,              ; string (e.g., "quip-root-v1")
  public_key,             ; 32 bytes (Ed25519)
  valid_from,             ; Unix timestamp (seconds)
  valid_until,            ; Unix timestamp (seconds)
  distribution_point,     ; string URI
  signature               ; 64 bytes (self-signed or cross-signed)
])
```

7.2. Trust Anchor Distribution

Trust anchors are distributed via:

1. _Hardcoded_ — Implementations MAY ship with a set of trusted roots

2. _DNS-based_ — _quip._trust.root.domain TXT records containing anchor hashes

3. _Transparency log_ — Optional certificate transparency-style log

7.3. Attestation Provider Certificate

```
tagged(65536, [
  "attestation_provider",
  provider_NodeId,          ; 32 bytes
  domain,                   ; string
  issuer,                   ; NodeId or anchor_id of issuer
  issued_at,                ; Unix timestamp (seconds)
  expires_at,               ; Unix timestamp (seconds, max 90 days)
  capabilities,             ; array of strings
  signature                 ; 64 bytes (by issuer)
])
```

7.4. Certificate Path Validation

When receiving a provider certificate chain, peers MUST perform the following validation:

Path building order:

1. Start from the end-entity (provider) certificate

2. Chain to issuer using issuer field (NodeId or anchor_id)

3. Continue until reaching a known trust anchor

4. Maximum chain depth: 8

Trust anchor matching: Trust anchors are identified by anchor_id string. Implementations MUST support exact string matching. Wildcard matching is NOT permitted.

Signature validation: All signatures MUST use Ed25519 [RFC8032]. Future versions may negotiate additional algorithms.

Expiry precedence: Certificates with expires_at in the past MUST be rejected. Trust anchors with valid_until in the past SHOULD be rejected (grace window of 24 hours permitted for rollover).

Rollover handling: Trust anchors SHOULD have overlapping validity windows during rollover. During overlap, either anchor is considered valid. After old anchor expires, it MUST NOT be accepted.

Cross-signing: Cross-signed certificates are permitted if they chain to a trust anchor. Maximum chain depth applies to the merged path.

Revocation check: Before accepting a certificate, check revocation status per Section 7.5. Revoked certificates MUST be rejected regardless of validity period.

7.5. Provider Revocation Freshness

Peers MUST enforce freshness limits on provider revocation lists:

Condition	Behavior
Revocation list age < 24 hours	Normal operation
Revocation list age 24-72 hours	DEGRADED mode, log warning
Revocation list age > 72 hours	VERIFIED mode MUST fail closed; reject all VERIFIED-mode attestations

Table 4

7.6. DNS Trust Anchor Bootstrapping

DNS-distributed trust anchors are advisory until chained to an existing trusted anchor or implementation trust store. Implementations MUST validate the DNSSEC chain for any DNS-discovered anchor. The first anchor in a deployment MAY be installed out-of-band (e.g., via package management, configuration file, or hardcoding). Once an anchor is trusted, it may be used to validate subsequent anchor updates.

8. Transport

8.1. Connection Establishment

QUIC connection with ALPN "quip". The first client-initiated bidirectional stream opened after QUIC connection establishment (stream ID 0) is designated the _QUIP Control Stream_. The second client-initiated bidirectional stream (stream ID 4) is designated the _QUIP Gossip Stream_.

Logical Role	Stream Type	Stream ID (Client)	Stream ID (Server)
Control stream	Client bidirectional	0	1
Gossip stream	Client bidirectional	4	5
Application streams	Client bidirectional	8, 12, ...	9, 13, ...

Table 5

All control messages (errors, rate advisories) are sent as CBOR arrays on the Control Stream.

Handshake sequencing: No application streams (streams other than the Control and Gossip streams) SHALL be processed until the QUIP handshake completes and transcript binding verification succeeds. The Control Stream and Gossip Stream MAY be used during handshake.

Phase 0: DNS Discovery

Query _quip._udp.domain.example SRV and TXT:

```
_quip._udp.domain.example. TXT
    "v=2; profiles=basic,documents; trust=compat; quic=:443"
```

Phase 1: QUIP Handshake (Control Stream)

Client sends:

```
[
  2,                    ; version
  0b111,                ; profiles bitmask
  "compat",             ; server_trust_mode
  0b01,                 ; identity_trust_mode (VERIFIED)
  {                     ; capabilities
    0: true,            ; datagram_support
    1: ["opus", "vp9"], ; media_codecs
    2: 4096             ; max_message_size
  },
  ["revocations/#"],    ; subscriptions
  {                     ; vector_clocks
    "revocation_list": {"a.com": 42}
  }
]
```

Server responds with identical structure.

Both peers compute active_profiles = local.profiles & remote.profiles. If zero, send error (Section 9.3) and close.

8.2. Wire Format

```
+------------------+-----------------------------------+
| varint length    | QUIP-CBOR object                  |
+------------------+-----------------------------------+
```

The CBOR object MUST be a single complete top-level item with no trailing bytes.

8.3. Stream Semantics

Logical Role	Direction	Reliability	Ordering
Control	Bidirectional	Reliable	Strict
Gossip	Bidirectional	Reliable	Strict
Application	Profile-dependent	Mixed	Mixed

Table 6

8.4. Bootstrap Discovery

Priority order:

1. DNS SRV + TXT (MUST)

2. QUIC Well-Known: ["WELL_KNOWN", "/peer"] on Control Stream (MUST)

3. Cached endpoints (SHOULD)

4. HTTPS fallback (OPTIONAL, DEPRECATED)

The full bootstrap handshake (GOSSIP_BOOTSTRAP exchange) is defined in Section 10.

9. Message Envelope

9.1. Envelope Structure

The signed payload is a CBOR array, not a map:

```
[
  "message",
  payload,              ; application CBOR
  NodeId,               ; 32 bytes
  sequence,             ; 64-bit uint (max 2^62-1)
  timestamp,            ; 64-bit Unix milliseconds
  message_id            ; 32 bytes (SHA-256)
]
```

The wire representation is:

```
tagged(65536, [
  "message",
  payload,
  NodeId,
  sequence,
  timestamp,
  message_id,
  signature            ; 64 bytes
])
```

9.2. Signature Computation (Domain Separation)

All signatures in QUIP use standard Ed25519 [RFC8032] without pre-hashing, with explicit domain separation strings prepended to the message.

```
signature = Ed25519_sign(
  private_key,
  domain_separator || QUIP-CBOR(signed_payload_array)
)
```

Object Type	Domain Separator (UTF-8)
Message envelope	"QUIP-MESSAGE-V1"
Gossip message	"QUIP-GOSSIP-V1"
Key rotation	"QUIP-ROTATION-V1"
Attestation (subject)	"QUIP-ATTESTATION-SUBJECT-V1"
Attestation (issuer)	"QUIP-ATTESTATION-ISSUER-V1"
Violation receipt	"QUIP-VIOLATION-V1"

Table 7

9.3. Error Messages

Errors are sent as CBOR arrays on the Control Stream:

["ERROR", error_code, message_id, error_text]

Where error_code is a uint (Section 17.2), message_id is 32 bytes (may be null for handshake errors), and error_text is a human-readable string.

9.4. Rate Advisory Messages

Rate advisories are sent as CBOR arrays on the Control Stream:

["RATE_ADVISORY", limit_type, current_rate, allowed_rate]

The limit_type field is a string matching the resource type being rate-limited (e.g., "message", "gossip", "datagram").

9.5. Sequence Validation Rules

Sequence numbers are scoped to (genesis_NodeId, discriminator) where discriminator is the string from the signed payload array (e.g., "message", "document", "gossip").

Rules:

* Sequence numbers MUST be strictly increasing per scope

* Gaps MAY be buffered up to a limit of 256 missing entries

* Out-of-order delivery is permitted, but sequence validation applies on receive

* Duplicates with the same message_id are rejected

* Retransmissions MUST reuse the identical message_id

9.6. Message ID Construction

```
message_id = SHA-256("QUIP-MESSAGE-ID-V1" || QUIP-CBOR([
  genesis_NodeId,
  NodeId,
  discriminator,
  sequence,
  timestamp
]))
```

9.7. Timestamp Validation

Timestamps are advisory only. Sequence numbers dominate ordering.

* _Soft window (MUST accept):_ ±300,000 ms (5 minutes)

* _Hard window (MAY accept):_ ±3,600,000 ms (1 hour) with warning

* _Outside hard window:_ MUST reject

Implementations MUST NOT use timestamps for causality determination.

10. Gossip and Consistency

10.1. Vector Clocks

Map from issuer → counter.

Issuer canonical forms: Domain: text string (major type 3). NodeId: byte string, 32 bytes (major type 2). Mixed representations for the same issuer are PROHIBITED.

Maximum vector clock entries: 1024. Entries beyond this limit SHOULD be pruned (oldest issuers first).

10.2. Gossip Message Format

The signed payload is a CBOR array:

```
[
  "gossip",
  resource_type,         ; "attestations" | "revocations" | "violations"
  resource_id,           ; string or bytes
  vector_clock,          ; map {issuer: counter}
  payload,               ; signed resource object
  ttl,                   ; uint (3-7)
  sender_NodeId          ; 32 bytes
]
```

The wire representation is:

```
tagged(65536, [
  "gossip",
  resource_type,
  resource_id,
  vector_clock,
  payload,
  ttl,
  sender_NodeId,
  signature
])
```

10.3. Resource Authority Model

Resource Type	Authoritative Issuer	Mutability
attestations	Attestation issuer	Immutable after issuance
revocations	Revocation issuer	Monotonic (add-only)
violations	Witness peer	Immutable

Table 8

10.4. Anti-Entropy Protocol

SYNC request:

```
["SYNC", resource_type, resource_id, their_clock]
```

SYNC response:

```
{
  "clock": merged_clock,
  "deltas": [payload1, payload2, ...],
  "next_token": continuation_token,  ; optional
  "complete": true|false
}
```

10.5. Conflict Resolution

When updates are concurrent (neither vector clock dominates the other), the deterministic merge order is:

1. _Causal check:_ If one clock has all issuers with counters >= the other clock, that clock wins (causal dominance)

2. _If concurrent:_ Lexicographic comparison of sorted (issuer, counter) tuples as QUIP-CBOR

3. _If equal:_ Higher NodeId (bytewise) of the signer of the update payload

4. _If equal:_ Smaller SHA-256 of the update payload

10.6. Gossip TTL

* Default: 5

* Range: 3-7

* Decrement before re-gossip; TTL=0 not re-gossiped

11. Rate Limiting and Accountability

11.1. Sequence Numbers

Every reliable message carries a sequence number per (sender, resource_type).

* 64-bit unsigned integer

* Maximum permitted value: 2^62 - 1 (approximately 4.6e18)

* Implementations MUST trigger a key rotation warning when the sequence number reaches 2^61 (50% of max)

* If a sender approaches 2^62 - 1, they MUST rotate their NodeId (Section 5.3) before the next increment

* Overflow beyond 2^62 - 1 is a provable violation (Section 17.2 code 16)

11.2. Violation Receipts

When a peer detects a violation (e.g., sequence gap, rate exceedance), it creates a signed violation receipt and gossips it.

Validation requirement: A violation receipt is considered valid only if corroborated by at least 3 independent witnesses as defined in Section 6.2. If fewer than 3 qualified domains exist in the federation, the threshold equals the number of domains.

Receipt signature:

```
signature = Ed25519_sign(
  private_key,
  "QUIP-VIOLATION-V1" || QUIP-CBOR(violation_data)
)
```

12. Resource Limits

Implementations MUST enforce the following limits:

Resource	Limit	Behavior When Exceeded
Maximum CBOR object size	64 KB	Reject with error
Maximum gossip message size	64 KB	Reject with error
Maximum document size	16 MB	Reject with error
Maximum attestation chain depth	8	Reject with error
Maximum rotation chain depth	32	Reject with error
Maximum vector clock entries	1024	Prune oldest
Maximum replay cache entries	100,000	LRU eviction
Maximum pending sequence gaps	256	Discard oldest

Table 9

13. Conformance Profiles

Profile	Required Features
Basic	NodeId, attestations, DANE (COMPAT), message envelope, rate limiting, gossip
Documents (Companion Document)	Adds: content addressing, schema registry, collections. Specified in a separate document.
Media (Experimental)	See Section 14

Table 10

The Documents profile features (content addressing by hash, schema registry, document collections) are specified in a companion Internet-Draft titled "QUIP Documents Profile".

14. Experimental Features

The following features are marked experimental in this version and MAY change incompatibly in future versions. Implementations SHOULD log a warning when experimental features are used.

Feature	Status
Media profile	Incomplete; separate document planned
Routing resource type	Not fully specified; use at own risk

Table 11

The Media profile uses QUIC datagrams as specified in [RFC9221].

15. Security Considerations

15.1. Threat Model Summary

QUIP defends against: domain impersonation (via subject signatures), replay attacks (via message_id cache), BGP hijacking (via DANE binding), DNS poisoning (via DNSSEC), split-brain/fork (via deterministic merge), gossip amplification (via TTL and rate limits), Sybil witnesses (via independent witness definition — raises operational cost of Sybil-style witness fabrication).

QUIP does NOT defend against: anonymity attacks, metadata confidentiality, global passive adversaries, Byzantine consensus, economic Sybil resistance, or witness collusion (relies on legal/ commercial incentives).

15.2. Replay Attacks

Domain separation (Section 9.2) prevents cross-context replay. The message_id cache provides per-message protection.

15.3. Downgrade Attacks

DNS downgrade protection (Section 6.3) and handshake transcript binding (Section 6.4) prevent silent capability stripping. STRICT mode refuses downgrades.

15.4. Canonicalization Attacks

QUIP-CBOR (Section 18) eliminates encoding ambiguity. Duplicate keys and indefinite lengths are rejected. QUIP-CBOR is defined as an application profile of [RFC8949].

16. Privacy Considerations

* NodeId reuse enables correlation

* Gossip reveals topology

* Address bindings are public

Linkability through rotation chains: Key rotation preserves identity continuity for accountability. All keys in a rotation chain are cryptographically linked to the genesis NodeId. Rotating keys does NOT provide unlinkability or anonymity.

17. IANA Considerations

17.1. ALPN Protocol ID Registration

IANA is requested to register the following ALPN protocol ID:

Protocol: QUIP

ID: "quip"

Reference: This document

17.2. QUIP Error Codes Registry

IANA is requested to create a registry titled "QUIP Error Codes" with the following initial contents:

Code	Name	Description
1	INVALID_SIGNATURE	Signature verification failed
2	ATTESTATION_EXPIRED	Attestation validity period exceeded
3	DNSSEC_FAILURE	DNSSEC validation failed
4	RATE_LIMIT_EXCEEDED	Quota exceeded
5	DUPLICATE_MESSAGE	message_id already seen
6	KEY_ROTATED	Sender’s NodeId rotated
7	UNSUPPORTED_VERSION	Schema version not supported
8	GOSSIP_RATE_LIMIT	Gossip update rate exceeded
9	PROFILE_MISMATCH	No common profile
10	HASH_MISMATCH	Document hash mismatch
11	DATAGRAM_NOT_SUPPORTED	Peer does not support datagrams
12	GOSSIP_BOOTSTRAP_FAILED	Bootstrap failed
13	KEY_COMPROMISED	Key compromise reported
14	GOSSIP_SYNC_TIMEOUT	Anti-entropy timeout

Table 12

Registration policy: IETF Review for codes 0-127. Codes 128-255 are reserved for application-specific extensions. Error code 0 is reserved and MUST NOT be used.

17.3. CBOR Tag 65536 Registration

IANA is requested to register the following CBOR tag in the CBOR Tags registry [RFC9277]:

Tag Number: 65536

Data Item: CBOR array

Semantic: Container for QUIP protocol messages. The first element of the array is a string discriminator indicating the message type (e.g., "message", "gossip", "document", "key_rotation", "primary_binding").

Point of Contact: Junior Joseph Mututi <jjmututicloud@gmail.com>

Fragment Identifier: N/A

Reference: This document

18. QUIP-CBOR Profile

QUIP defines a strict deterministic CBOR profile called the QUIP Deterministic CBOR Profile (QDP). This is an application profile, not a replacement for [RFC8949] canonical CBOR. QDP is used exclusively within QUIP protocol messages.

18.1. Encoding Rules

Class	Type	Ordering Rule
1	Unsigned integers	Numeric ascending, shortest encoding first
2	Negative integers	Numeric ascending (more negative first)
3	Byte strings	Length-ordered, then lexicographic
4	Text strings	Length-ordered, then UTF-8 bytewise lexicographic

Table 13

Prohibited constructs: duplicate map keys, indefinite-length items, floating point values, non-minimal UTF-8, CBOR simple values other than false/true/null.

19. Implementation Guidelines

19.1. State Persistence

Implementations MUST persist vector clocks for all tracked resources, sequence number state per (genesis_NodeId, discriminator), revocation state for known peers, and peer relationship state. Replay cache persistence is OPTIONAL.

19.2. Sequence State Loss Recovery

If sequence state is lost after restart, the implementation MUST log a critical error, rotate the NodeId (Section 5.3) to start a new sequence space, and NOT reuse prior sequence numbers.

20. References

20.1. 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/info/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/info/rfc8174>.

[RFC9000] Iyengar, J. and M. Thomson, "QUIC: A UDP-Based Multiplexed and Secure Transport", RFC 9000, DOI 10.17487/RFC9000, May 2021, <https://www.rfc-editor.org/info/rfc9000>.

[RFC8949] Bormann, C. and P. Hoffman, "Concise Binary Object Representation (CBOR)", RFC 8949, DOI 10.17487/RFC8949, December 2020, <https://www.rfc-editor.org/info/rfc8949>.

[RFC9277] Bormann, C., "CBOR Tags Registry Revisions", RFC 9277, DOI 10.17487/RFC9277, August 2022, <https://www.rfc-editor.org/info/rfc9277>.

[RFC6698] Hoffman, P. and J. Schlyter, "DNS-Based Authentication of Named Entities (DANE) TLSA Protocol", RFC 6698, DOI 10.17487/RFC6698, August 2012, <https://www.rfc-editor.org/info/rfc6698>.

[RFC8032] Josefsson, S. and I. Liusvaara, "Edwards-Curve Digital Signature Algorithm (EdDSA)", RFC 8032, DOI 10.17487/RFC8032, January 2017, <https://www.rfc-editor.org/info/rfc8032>.

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

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

[RFC4035] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "Protocol Modifications for the DNS Security Extensions", RFC 4035, DOI 10.17487/RFC4035, March 2005, <https://www.rfc-editor.org/info/rfc4035>.

[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/info/rfc8446>.

20.2. Informative References

[RFC9221] Pauly, T., Kinnear, E., and D. Schinazi, "An Unreliable Datagram Extension to QUIC", RFC 9221, DOI 10.17487/RFC9221, March 2022, <https://www.rfc-editor.org/info/rfc9221>.

Author’s Address

Junior Joseph Mututi
Individual
Email: jjmututicloud@gmail.com