Q-Sign

Verifiable Authority and Accountability Substrate for Autonomous Systems

Status: Draft, version 1.0.0 **Date:** 2026-06-13 **Publisher:** H33.ai, Inc. **Standard track:** Open · vendor-neutral · implementation-agnostic **Conformance:** see `/standards/conformance/`

Abstract

Q-Sign defines a cryptographic substrate for authority and accountability in autonomous systems. The substrate answers four substrate-layer questions on every protected action: who authorized this, was that authority valid at action time, can the decision be replayed deterministically, and is the complete authority chain visible end-to-end.

The substrate composes five components — Lineage DAG, Q-Key, BAAE, Replay, and NAP — bound together by triple-family post-quantum signatures (ML-DSA-87, SLH-DSA-256s, FALCON-1024). The output of the substrate on every action is a single portable receipt envelope, `.h33pqv.json`, that any third party can verify in a browser without contacting the issuer.

Q-Sign is not a governance policy engine and does not enforce business rules. It enforces a single, narrower property: no actor receives authority from anything not bound to them and valid at action time. Application-layer guarantees (no authorized proposal executes twice, no two operators issue conflicting permits) compose on top of, but do not displace, this substrate guarantee.

---

1. Scope and audience

This specification is normative for implementers building Q-Sign substrates and for verifiers consuming Q-Sign-emitted artifacts. It is informative for security architects, auditors, and platform operators evaluating Q-Sign as the foundation for authority-bound automation.

This specification does not standardize any specific governance policy, claims schema, or runtime policy language. It standardizes only the substrate semantics, the envelope format, the signature ceremony, the replay contract, and the conformance vectors.

---

2. Terminology

The key words “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “MAY”, and “OPTIONAL” in this document are to be interpreted as described in RFC 2119 and RFC 8174.

- **Principal:** an entity (human, organization, or autonomous agent) that can be addressed by an authority binding.
- **Authority:** a cryptographically attestable claim that one principal has been granted a scoped capability by another principal, valid for a bounded interval.
- **Action:** a discrete, replayable operation a principal performs in pursuit of a goal.
- **Receipt:** a signed `.h33pqv.json` artifact representing the verdict on an action, intended to be verifiable independently of the issuing platform.

---

3. Threat model

Q-Sign operates against the following adversaries:

1. **Forgery adversary:** attempts to fabricate authority not previously granted.
2. **Repudiation adversary:** attempts to deny having authorized an action that was correctly attested at action time.
3. **Replay adversary:** attempts to replay a stale authority to execute an action outside its time bounds.
4. **Scope-expansion adversary:** attempts to use a delegated authority to perform actions outside the delegated scope.
5. **Quantum-capable adversary:** is assumed to hold a cryptographically relevant quantum computer (CRQC) and to have access to any historical signature or transcript.

Q-Sign provides forgery resistance, repudiation resistance, replay resistance, scope-expansion resistance, and post-quantum security under all five adversary classes.

Q-Sign does not provide application-level integrity (e.g., ensuring that the action the principal *intended* to authorize matches the action recorded by the platform). Application correctness is the responsibility of higher layers and is verified through the Replay component.

---

4. Cryptographic basis

Q-Sign uses three NIST-selected post-quantum digital signature families simultaneously on every signature operation. An implementation MUST sign every Q-Sign envelope under all three of:

- **ML-DSA-87** (Module-Lattice Digital Signature Algorithm, NIST FIPS 204, category 5 parameter set).
- **SLH-DSA-256s** (Stateless Hash-Based Signature Algorithm, NIST FIPS 205, 256-bit hash output, small-signature variant).

- **FALCON-1024** (NIST Round 3 selection, 1024-bit lattice parameter set).

A verifier MUST verify all three signatures independently and MUST reject the envelope if any one of the three fails verification. The all-three-must-pass policy is the substrate's hedge against a future cryptanalytic break in a single family.

The following primitives are **prohibited** in any Q-Sign implementation: PLONK, Groth16, BLS pairings, KZG commitments, any pairing-based scheme, any elliptic-curve scheme (P-256, P-384, secp256k1, Curve25519, BLS12-381), and any RSA variant. Q-Sign is a fully post-quantum substrate; introduction of any non-PQ primitive renders the substrate non-conforming.

Hash functions: SHA3-384 is REQUIRED for canonical hashing of envelope contents. SHAKE-256 MAY be used internally by SLH-DSA-256s per its standard.

---

5. Component 1: Lineage DAG

5.1 Definition

The Lineage DAG is a directed acyclic graph in which each node represents a principal and each edge represents a signed delegation of authority. A delegation edge `e = (parent → child, scope, t_start, t_end, revocation_handle)` carries five fields:

- `parent`: the delegating principal's stable identifier.
- `child`: the receiving principal's stable identifier.
- `scope`: a structured set of capabilities authorized for the child.
- `t_start`, `t_end`: the validity interval, inclusive of the start, exclusive of the end.
- `revocation_handle`: a value the parent (or its delegated revoker) can publish to invalidate the edge before `t_end`.

Each edge is signed by the parent under the parent's Q-Key.

5.2 Substrate rules

The Lineage DAG MUST enforce the following invariants:

- **No widening.** A child's scope MUST be a subset of its parent's scope.
- **No extension.** A child's `(t_start, t_end)` MUST be contained within its parent's `(t_start, t_end)`.
- **No cycles.** The graph MUST remain acyclic; an edge that would close a cycle MUST be rejected.
- **No retroactive editing.** Once signed, an edge is immutable; revocation occurs by publishing the revocation handle, not by modifying the edge.

5.3 Verification

For an action by principal `child` at time `t`, `Lineage.verify(child, action.scope, t)` returns valid only if there exists a path from a root principal to `child` such that every edge on the path is unrevoked at `t`, `t_start <= t < t_end` on every edge, and `action.scope` is contained within the scope on every edge.

---

6. Component 2: Q-Key

6.1 Definition

A Q-Key is a triple of keypairs bound to a single principal. The triple consists of one keypair from each of the three signature families specified in §4. A Q-Key has a stable public identifier derived from `SHA3-384(ML-DSA-87.pk || SLH-DSA-256s.pk || FALCON-1024.pk)`.

6.2 Signing operation

A Q-Sign signature over a canonical byte string `m` is the triple:

```
Sig(m) = (
  ML-DSA-87.sign(sk_ml, m),
  SLH-DSA-256s.sign(sk_slh, m),
  FALCON-1024.sign(sk_falcon, m)
)
```

A signer MUST hold all three secret keys and MUST sign with all three on every signing operation.

6.3 Verification

A verifier MUST verify all three signatures and MUST reject the signature if any one fails. A signature with only two valid component signatures is non-conforming.

6.4 Key rotation

Key rotation is performed by issuing a delegation edge from the old Q-Key to the new Q-Key with the same scope and a time bound that overlaps the cutover window. The two keys MUST verify in parallel during the overlap window; outside it, the old key’s authority is dead.

---

7. Component 3: BAAE — Bound Authority Action Envelope

7.1 Definition

A BAAE is a structured envelope that binds an intended action to the authority that authorizes it, the instruction that describes it, and the recipient that receives it. A conforming BAAE contains, at minimum:

```
{
  "version": "q-sign/1.0",
  "action": <canonical action object>,
  "lineage_path": [<edge>, <edge>, ...],
  "instruction_tag": <tag>,
  "recipient": <recipient identifier>,
  "issued_at": <timestamp>,
  "envelope_hash": <SHA3-384 of canonical envelope minus
signature>,
  "signatures": {
    "ml_dsa_87": <signature>,
    "slh_dsa_256s": <signature>,
    "falcon_1024": <signature>
  }
}
```

7.2 Three trust bindings

The substrate evaluates three trust bindings before the action is admitted:

1. **Authority binding.** The `lineage_path` MUST be a valid Lineage DAG path from a root principal to the signing actor, with every edge unrevoked at `issued_at`, scope sufficient for `action`, and time bounds containing `issued_at`.
2. **Instruction binding.** `instruction_tag` MUST match the action shape under the binding contract (declared per action type), and `recipient` MUST be the intended target of the action.
3. **Execution binding.** A downstream HATS gate MUST emit a Permit covering this BAAE, and the resulting `.h33pqv.json` receipt MUST be linkable to this envelope by `envelope_hash`.

If any binding fails, the BAAE is rejected and the action is not admitted. A Reject receipt citing the failed binding is emitted.

---

8. Component 4: Replay

8.1 Definition

Replay is the deterministic re-execution of a recorded BAAE against a captured Lineage DAG snapshot. A replay run takes as input the BAAE and the lineage state at `issued_at`, and produces the same verdict the substrate produced at action time.

8.2 Determinism contract

A conforming implementation MUST canonicalize:

- The action object using a deterministic JSON canonicalization (RFC 8785 JCS).
- The lineage path serialization, in topological order from root to actor.
- The envelope hash computation, using SHA3-384 over the canonical envelope minus the signatures block.

A replay run on conforming implementations MUST produce byte-identical envelope hashes and identical verdicts. Differences indicate an implementation defect or a non-canonical input.

8.3 Adversarial replay

A Reject verdict on adversarial inputs is normative. The conformance suite (see §11) supplies test vectors for:

- Tampered envelope bytes.
- Expired lineage edges at the time of replay.
- Mis-scoped delegations attempting to authorize out-of-scope actions.
- Revoked edges replayed after revocation.
- Repudiation attempts where the actor denies the envelope.

All five MUST reject deterministically.

---

9. Component 5: NAP — Nested Authority Path

9.1 Definition

NAP is the end-to-end, human-readable rendering of an authority chain across an arbitrary number of intermediate principals. Where the Lineage DAG provides the cryptographic substrate, NAP provides the human-auditable artifact. A NAP record contains the ordered sequence of principals from a named human (or root principal) through every intermediate agent to the leaf action, with each hop's scope, time bound, and Q-Key fingerprint visible.

9.2 Value inflection

NAP is the substrate property that scales. When one human authorizes one agent that authorizes a fleet of sub-agents, NAP-at-scale lets a single auditor read the entire descent in one pass and verify the chain without re-running the underlying systems. The auditor verifies the NAP record by checking each Q-Sign signature on each hop; no policy interpretation is required.

9.3 NAP guarantee

A correctly constructed NAP record satisfies:

- Every leaf points back to a named root principal.
- Every hop's scope is a subset of its parent's scope.
- Every hop's time bound is contained within its parent's.
- No hop can be added to the record after the fact (the record is signed end-to-end by every principal in the chain).
- No parent can repudiate (the parent's signature on the delegation edge is in the record).

---

10. The portable receipt

Every Q-Sign-bound action emits exactly one `.h33pqv.json` receipt at the conclusion of the action, regardless of verdict. The receipt is the portable substrate artifact intended for external verification.

A Q-Sign receipt MUST contain:

- A canonical reference to the BAAE that produced it (by `envelope_hash`).
- The HATS gate's verdict (Permit or Reject) and the binding that failed for Reject.
- A NAP record covering the authority chain to the issuing actor.
- The triple-family signatures over the canonical receipt body.

A receipt is independently verifiable: the verifier needs only the receipt itself and access to the Q-Key public material referenced in the NAP. The issuing platform is not required to be reachable.

---

11. Conformance

An implementation claims Q-Sign conformance by passing the conformance vector suite published at `/standards/conformance/`. The suite covers:

- All five components individually, with positive and adversarial inputs.
- Cross-component composition (Lineage → Q-Key → BAAE → Replay → NAP).
- Receipt format conformance.
- Verifier conformance (an implementation MUST accept conforming receipts from any conforming signer).

- Adversarial validation, including all five Reject vectors from §8.3.

Example reference artifacts demonstrating each component are published at `/standards/q-sign/artifacts/`.

---

12. Security considerations

12.1 Crypto-agility within the post-quantum boundary

Q-Sign anchors on three specific PQ families to provide both performance and a hedge against single-family cryptanalytic breaks. A future version of this standard may introduce a fourth family or substitute one of the three; any such revision MUST preserve the all-must-verify property and MUST NOT introduce any non-PQ primitive.

12.2 Quantum-capable adversaries with historical access

Because all three signature families are PQ-secure, an adversary who records every signature ever emitted gains no benefit from a future CRQC. This is the substrate's response to "harvest now, decrypt later" attacks on signature material.

12.3 Revocation propagation latency

Revocation handles must propagate to every verifier promptly. Q-Sign specifies the revocation handle and the verification check; revocation propagation latency is an operational concern outside the substrate. Implementations SHOULD publish revocation feeds at well-known URIs and SHOULD support pull and push semantics.

12.4 Side channels

Implementations of ML-DSA-87, SLH-DSA-256s, and FALCON-1024 MUST be constant-time per the published implementation guidance for each scheme. Side-channel resistance is the implementer's responsibility.

---

13. References

- NIST FIPS 203, *Module-Lattice-Based Key-Encapsulation Mechanism Standard*.
- NIST FIPS 204, *Module-Lattice-Based Digital Signature Standard*.
- NIST FIPS 205, *Stateless Hash-Based Digital Signature Standard*.
- NIST PQC Round 3 Submission Documentation: *FALCON*.
- RFC 2119, *Key words for use in RFCs to Indicate Requirement Levels*.
- RFC 8174, *Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words*.
- RFC 8785, *JSON Canonicalization Scheme (JCS)*.

---

14. Acknowledgments

The Q-Sign substrate emerged from H33's H33-Root authority substrate work and from adversarial hardening conducted with the H33-Chaos validation harness across multi-agent attack scenarios. The all-three-must-verify policy was tightened in response to multi-agent swarm attacks that surfaced production bugs single-agent green tests missed: cross-agent receipt swap, proposing-agent vs receipt-signer split, and expired-tag replay.

Appendix A: Example BAAE (informative)

A minimal example BAAE is provided in `/standards/q-sign/artifacts/` for illustrative purposes. The example is non-normative; the normative envelope format is in §7.1.

Appendix B: Substrate vs application boundary (informative)

Q-Sign provides the **substrate guarantee**: no actor receives authority from anything not bound to them and valid at action time. Q-Sign does not provide application guarantees such as no-double-spend or no-conflicting-permit. Application guarantees compose on top of Q-Sign; they do not replace it. Confusing the substrate boundary with the application boundary is the most common implementation error and produces hard-to-diagnose authority leakage.

Q-Sign is published by H33.ai, Inc. as an open standard. Comments and proposed revisions can be submitted via contact@h33.ai or as a pull request against the standard's source.