Unheaded Kingdom
Internet-Draft
Intended status: Experimental
Expires: 20 September 2026

S. Bellis
Unheaded

19 March 2026

Post-Quantum Packet Authentication for the Unheaded Protocol
draft-bellis-unheaded-pqc-authentication-00

Abstract

This document specifies a post-quantum cryptographic (PQC) authentication mechanism for the Unheaded Protocol Foundation. It defines a multi-algorithm, dual-layer, tiered authentication architecture integrating three NIST PQC digital signature standards -- FIPS 205 (SLH-DSA), FIPS 204 (ML-DSA), and FIPS 206 (FN-DSA) -- plus two NIST PQC key-encapsulation mechanisms -- FIPS 203 (ML-KEM) and FIPS 207 (HQC) -- with the Monad wire format, Sophia BPF map dictionaries, and Wotan per-flow memory model.

Layer 1 (Wire-Level, REQUIRED): Full PQC signatures are stored in Sophia BPF maps via a "signature-by-reference" scheme. The Monad register carries compact 12-byte references (SigRef, KeyRef, SeqNum, HashPfx). Shield verifies signatures at the network perimeter and strips Monad Hop-by-Hop headers at ingress -- internal kingdom traffic never carries PQC wire overhead.

Layer 2 (Application-Level, OPTIONAL): User applications MAY define verification requirements in Sophia application policy dictionaries. After wire-level authentication succeeds, the application reads Wotan per-flow PQC state and matches it against its own policy.

Four PQC compliance tiers -- NONE, STANDARD, ENHANCED, and SOVEREIGN -- are signaled via Kingdom Mode bits in the Monad flags byte.

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/.

This Internet-Draft will expire on September 19, 2026.

Copyright Notice

Status of This Memo

This Internet-Draft will expire on 20 September 2026.

Copyright Notice

Table of Contents

1. Introduction

The Unheaded Protocol Foundation [MONAD] defines a 20-byte register carried in an IPv6 Hop-by-Hop extension header [RFC8200]. This register provides per-packet metadata for service mesh operations including tracing, QoS classification, circuit breaking, and deployment ring isolation.

As quantum computing advances threaten existing cryptographic assumptions, infrastructure systems MUST prepare for post-quantum cryptographic migration. NIST has finalized five PQC standards: three digital signature algorithms -- FIPS 205 (SLH-DSA) [FIPS205], FIPS 204 (ML-DSA) [FIPS204], and FIPS 206 (FN-DSA) [FIPS206] -- and two key-encapsulation mechanisms -- FIPS 203 (ML-KEM) [FIPS203] and FIPS 207 (HQC) [FIPS207].

PQC signature sizes range from 666 bytes (FN-DSA-512) to 49,856 bytes (SLH-DSA-SHAKE-256f), all far exceeding the Monad register’s 12-byte value field. This document defines a "signature-by-reference" scheme

that stores full PQC signatures and public keys in Sophia BPF maps while carrying compact references in the Monad register. Sophia [SOPHIA] dictionaries provide the BPF map structures for signature and key storage. The scheme is algorithm-agnostic -- the same 12-byte wire layout supports all three signature standards.

This approach provides:

(a) Post-quantum packet authentication at the wire level (Layer 1).

(b) Zero increase in per-packet wire overhead (Monad remains 20 bytes).

(c) Amortized verification cost across flow lifetime (first-packet verification, subsequent packets use cached result).

(d) Configurable verification policy per trust boundary.

(e) Optional application-level policy verification (Layer 2) via Sophia dictionaries, enabling enterprises to define custom authentication requirements without modifying the wire protocol.

(f) Clean perimeter isolation -- Shield strips Monad HbH headers at ingress, so internal kingdom traffic carries zero PQC wire overhead. Wire-level PQC is exclusively a perimeter concern.

(g) Algorithm agility -- the algo_id field and signature-by-reference design support all five NIST PQC standards with zero wire format changes. New algorithms require only a registry entry and a verifier implementation.

(h) Tiered compliance -- four PQC compliance tiers (NONE, STANDARD, ENHANCED, SOVEREIGN) enable graduated deployment from development environments (zero overhead) to government-grade multi-algorithm cross-verification.

2. Requirements Language

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. Terminology

SLH-DSA: Stateless Hash-Based Digital Signature Algorithm, as

defined in FIPS 205 [FIPS205]. Formerly known as SPHINCS+. Hash-based construction. eBPF-native (integer-only operations).

ML-DSA: Module-Lattice-Based Digital Signature Algorithm, as defined in FIPS 204 [FIPS204]. Formerly known as CRYSTALS-Dilithium. Lattice-based construction. eBPF-native (integer NTT modular arithmetic).

FN-DSA: FFT over NTRU-Lattice-Based Digital Signature Algorithm, as defined in FIPS 206 [FIPS206]. Formerly known as FALCON. Lattice-based construction. NOT eBPF-native: signing requires IEEE-754 binary64 floating-point arithmetic (FFT, LDL tree decomposition, discrete Gaussian sampling). Verification is integer-only and MAY be performed in eBPF.

ML-KEM: Module-Lattice-Based Key-Encapsulation Mechanism, as defined in FIPS 203 [FIPS203]. Formerly known as CRYSTALS-Kyber. Used for key establishment, not per-packet authentication.

HQC: Hamming Quasi-Cyclic Key-Encapsulation Mechanism, as defined in FIPS 207 [FIPS207]. Code-based KEM providing non-lattice diversity for key establishment.

Compliance Tier: One of four PQC enforcement levels (NONE, STANDARD, ENHANCED, SOVEREIGN) signaled via Kingdom Mode bits K1|K0 in the Monad flags byte. Tiers govern Shield verification behavior, algorithm selection, and Layer 2 enforcement policy.

Signature-by-Reference: A scheme where a compact reference (index) is carried in the packet header, pointing to a full cryptographic signature stored in a kernel-resident data structure (Sophia BPF map).

SigRef: A 24-bit unsigned integer indexing into the Sophia PQC Signature Map.

KeyRef: A 24-bit unsigned integer indexing into the Sophia PQC Public Key Map.

SeqNum: A 32-bit per-flow sequence number providing replay resistance. Monotonically increasing within a flow lifetime.

HashPfx: A 16-bit integrity tag derived from SHA-256 of the full SLH-DSA signature, truncated to 2 bytes.

Pseudo-Header: The set of immutable packet fields over which the SLH-DSA signature is computed (Section 7).

Verification Policy: A per-trust-boundary configuration determining whether packets are forwarded before (OPTIMISTIC) or after (PESSIMISTIC) signature verification completes (Section 9).

Layer 1 (Wire-Level Authentication): Infrastructure-grade PQC verification performed by Shield/XDP at the network perimeter. Operates on Monad HbH headers. REQUIRED for all PQC-enabled flows. Transparent to applications.

Layer 2 (Application-Level Policy): Optional verification where user applications read Wotan per-flow PQC state and match it against requirements defined in Sophia application policy dictionaries (Section 14). Operates AFTER Layer 1 succeeds and Monad headers have been stripped.

Application Policy Dictionary: A Sophia dictionary defining per-application PQC requirements: minimum security level, allowed algorithm set, key pinning requirements, maximum key age. Consumed by Layer 2 verification.

Header Stripping: The removal of Monad Hop-by-Hop extension headers at Shield ingress. Internal kingdom traffic operates without PQC wire overhead; Wotan per-flow state preserves the authentication result for application consumption.

4. Protocol Overview

The PQC authentication scheme operates in four phases:

Phase 1 - Key and Signature Provisioning:

The control plane generates SLH-DSA key pairs and pre-computes signatures over expected pseudo-headers. Full signatures and public keys are loaded into Sophia BPF maps. SigRef and KeyRef indices are assigned.

Phase 2 - Packet Marking:

When a packet enters the Unheaded network at Shield ingress, the Monad register’s value field is populated with the SigRef, KeyRef, SeqNum, and HashPfx corresponding to the flow’s authentication context. The S (Signed) flag in byte 0x01 is set to 1.

Phase 3 - Wire-Level Verification (Layer 1):

At the Shield ingress from an untrusted network, the eBPF program at XDP reads the SigRef and KeyRef from the Monad register, retrieves the full signature and public key from Sophia maps, and either:

(a) Checks a cached verification result (fast path), or

(b) Submits the signature to the userspace verifier via ring buffer for asynchronous SLH-DSA verification (slow path, first packet of flow only).

Upon successful verification, Shield strips the Monad HbH extension header. The verification result is persisted in Wotan per-flow PQC state (Section 12). Internal kingdom traffic proceeds without PQC wire overhead.

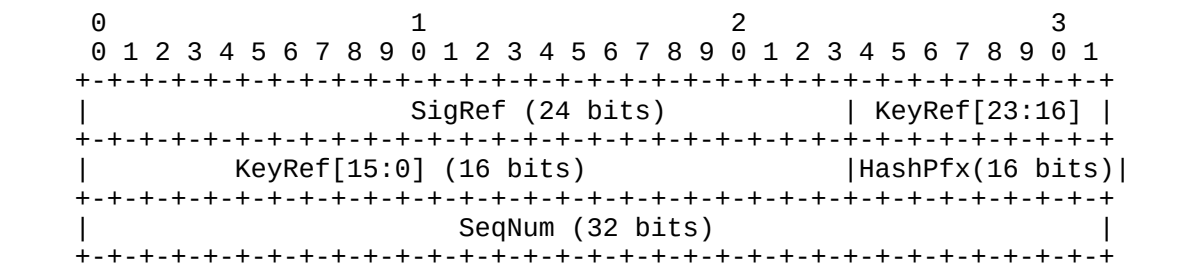
Phase 4 - Application-Level Policy (Layer 2, OPTIONAL):

After Shield ingress strips Monad headers and forwards the packet into the kingdom, user applications MAY perform a second verification pass. The application reads the flow's PQC state from Wotan memory (algo_id, key_epoch, verified status, key fingerprint) and matches it against requirements defined in a Sophia application policy dictionary.

If the flow's PQC state does not satisfy the application's policy (e.g., algorithm too weak, key too old, fingerprint not pinned), the application MAY reject the flow at the application layer. This provides defense-in-depth and enables per-application security posture without modifying the wire protocol.

5. Monad Value Layout for PQC Authentication

When the S (Signed) flag is set in the Monad flags byte, the 12-byte value region of the Monad register (bytes specific to the flow action and QoS fields as defined in [MONAD]) SHALL be interpreted as follows:



Field definitions:

SigRef (3 octets, unsigned, big-endian): Index into the Sophia PQC

Signature Map. Range: 0 to 16,777,215. Value 0 is reserved (indicates "no signature"). Implementations MUST reject packets where S flag is set but SigRef is 0.

KeyRef (3 octets, unsigned, big-endian): Index into the Sophia PQC Public Key Map. Range: 0 to 16,777,215. Value 0 is reserved. Implementations MUST reject packets where S flag is set but KeyRef is 0.

HashPfx (2 octets, unsigned, big-endian): First two bytes of SHA-256(signature_blob). Provides a fast integrity check binding the SigRef to a specific signature. Collision probability: 1/65,536 per SigRef pair (see Security Considerations).

SeqNum (4 octets, unsigned, big-endian): Per-flow sequence number. MUST be monotonically increasing within a flow lifetime. Provides replay resistance (see Section 7). The signed pseudo-header includes SeqNum; therefore replayed packets with stale sequence numbers will fail verification.

Note: When PQC compliance tiers are active (K1|K0 != 00), the S flag (bit 3 of the flags byte) is repurposed from its SAMPLED semantics defined in [MONAD] to indicate Signed status. This dual-use is signaled by the Kingdom Mode bits; implementations MUST check the K1|K0 field before interpreting the S flag. The Monad Flags Registry update for this extended semantics is specified in Section 17.2.

6. Sophia PQC Map Structures

6.1. PQC Signature Map

A new Sophia BPF hash map SHALL be created for PQC signature storage:

```
Map name:      sophia_pqc_sigs
Key type:      uint32 (SigRef, zero-extended from 24 bits)
Value type:    struct sophia_pqc_sig_entry
Max entries:   Configurable (RECOMMENDED: 1,048,576)
Flags:         BPF_F_RDONLY_PROG (data plane read-only)
Pinning:       /sys/fs/bpf/sophia/pqc_sigs
```

The value structure:

```
struct sophia_pqc_sig_entry {
    uint8_t  algo_id;           /* SLH-DSA parameter set ID    */
    uint8_t  verified;          /* 0=pending, 1=valid, 2=invalid */
    uint16_t sig_len;           /* Signature length in bytes    */
    uint32_t flow_label;        /* Owning flow label            */
    uint64_t verify_timestamp;  /* Nanosecond timestamp of verification */
    uint8_t  signature[];       /* Variable-length SLH-DSA signature */
};
```

The algo_id field SHALL use values from the PQC Algorithm Registry (Section 17):

SLH-DSA Parameter Sets (FIPS 205 -- hash-based, eBPF-native):

Value	Parameter Set	Sig Size	Security	eBPF?
0x00	Reserved	N/A	N/A	N/A
0x01	SLH-DSA-SHA2-128s	7,856 B	Level 1	YES
0x02	SLH-DSA-SHA2-128f	17,088 B	Level 1	YES
0x03	SLH-DSA-SHA2-192s	16,224 B	Level 3	YES
0x04	SLH-DSA-SHA2-192f	35,664 B	Level 3	YES
0x05	SLH-DSA-SHA2-256s	29,792 B	Level 5	YES
0x06	SLH-DSA-SHA2-256f	49,856 B	Level 5	YES
0x07	SLH-DSA-SHAKE-128s	7,856 B	Level 1	YES
0x08	SLH-DSA-SHAKE-128f	17,088 B	Level 1	YES
0x09	SLH-DSA-SHAKE-192s	16,224 B	Level 3	YES
0x0A	SLH-DSA-SHAKE-192f	35,664 B	Level 3	YES
0x0B	SLH-DSA-SHAKE-256s	29,792 B	Level 5	YES
0x0C	SLH-DSA-SHAKE-256f	49,856 B	Level 5	YES

Table 1

ML-DSA Parameter Sets (FIPS 204 -- lattice-based, eBPF-native):

Value	Parameter Set	Sig Size	Security	eBPF?
0x10	ML-DSA-44	2,420 B	Level 2	YES
0x11	ML-DSA-65	3,309 B	Level 3	YES
0x12	ML-DSA-87	4,627 B	Level 5	YES

Table 2

FN-DSA Parameter Sets (FIPS 206 -- lattice-based, userspace verify):

Value	Parameter Set	Sig Size	Security	eBPF?
0x20	FN-DSA-512	666 B	Level 1	NO*
0x21	FN-DSA-1024	1,280 B	Level 5	NO*

Table 3

*FN-DSA verification uses integer NTT and MAY be implemented in eBPF. However, FN-DSA signing requires IEEE-754 binary64 floating-point (FFT, LDL tree, Gaussian sampling) and MUST NOT execute in eBPF. See Section 15 (Multi-Algorithm Considerations).

KEM Algorithm Identifiers (key establishment only, not per-packet):

Value	Algorithm	CT Size	Security	Use
0x80	ML-KEM-512	768 B	Level 1	Tunnel key
0x81	ML-KEM-768	1,088 B	Level 3	Tunnel key
0x82	ML-KEM-1024	1,568 B	Level 5	Tunnel key
0x90	HQC-128	4,497 B	Level 1	Tunnel key
0x91	HQC-192	9,042 B	Level 3	Tunnel key

Table 4

Reserved Ranges:

Value	Description
0x0D-0x0F	Reserved (SLH-DSA future)
0x13-0x1F	Reserved (ML-DSA future)
0x22-0x7F	Reserved (signature algos)
0x83-0x8F	Reserved (ML-KEM future)
0x92-0xFE	Unassigned
0xFF	Reserved

Table 5

The sophia_pqc_sigs map MUST be created with the BPF_F_RDONLY_PROG flag. Data plane eBPF programs (XDP, TC) MUST NOT have write access. Only the control plane (userspace) SHALL write to this map.

6.2. PQC Public Key Map

A new Sophia BPF hash map SHALL be created for PQC public key storage:

```
Map name:      sophia_pqc_keys
Key type:      uint32 (KeyRef, zero-extended from 24 bits)
Value type:    struct sophia_pqc_key_entry
Max entries:   Configurable (RECOMMENDED: 65,536)
Flags:         BPF_F_RDONLY_PROG
Pinning:       /sys/fs/bpf/sophia/pqc_keys
```

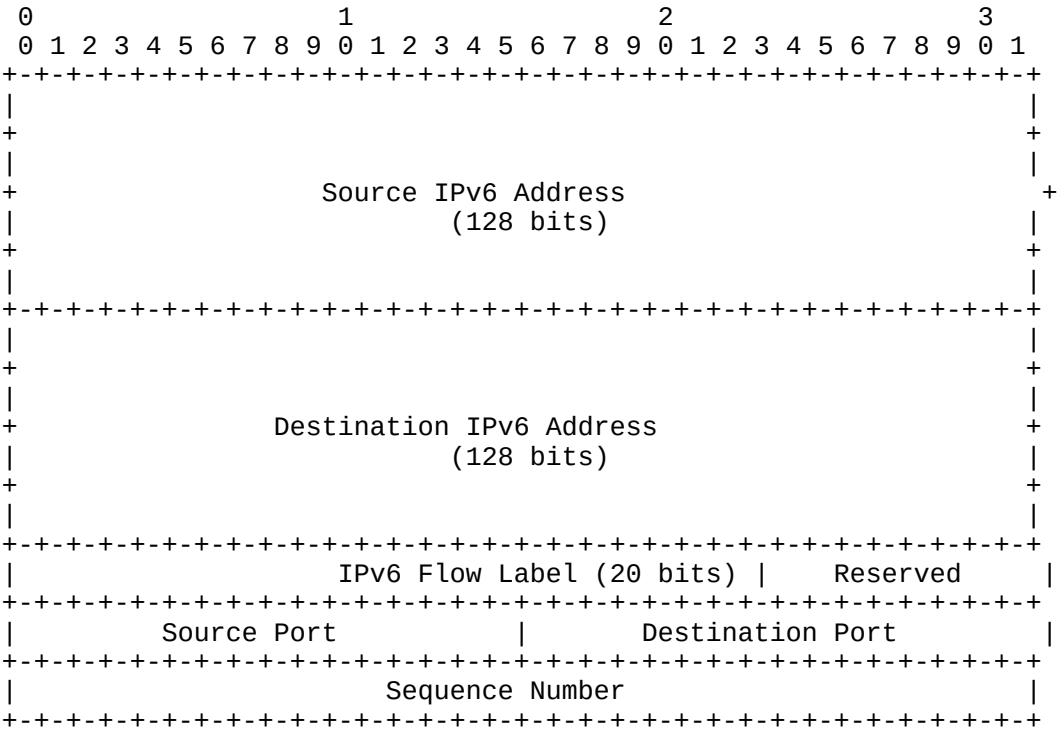
The value structure:

```
struct sophia_pqc_key_entry {
    uint8_t  algo_id;           /* Must match sig entry algo_id */
    uint8_t  key_epoch;         /* Key rotation epoch counter   */
    uint16_t key_len;           /* Public key length in bytes   */
    uint8_t  fingerprint[32];   /* SHA-256 of full public key   */
    uint8_t  public_key[];      /* Variable-length SLH-DSA pubkey */
};
```

7. Signed Pseudo-Header

The SLH-DSA signature MUST be computed over a pseudo-header constructed from immutable packet fields. The pseudo-header ensures that the signature binds to a specific flow, destination, and sequence position.

The pseudo-header is constructed as follows (52 bytes):



Fields:

Source IPv6 Address (16 octets): Copied from the IPv6 header. MUST NOT change in transit.

Destination IPv6 Address (16 octets): Copied from the IPv6 header. MUST NOT change in transit.

IPv6 Flow Label (20 bits): Copied from the IPv6 header. The remaining 12 bits of this 4-byte field MUST be set to zero.

Source Port (2 octets): Transport layer source port. Set to zero if the transport protocol does not use ports.

Destination Port (2 octets): Transport layer destination port. Set to zero if the transport protocol does not use ports.

Sequence Number (4 octets): Copied from the SeqNum field of the Monad PQC value layout (Section 5). This field provides replay resistance.

Implementations MUST construct the pseudo-header identically at both the signing and verification endpoints. Any difference in construction will cause verification to fail.

Note: The pseudo-header intentionally excludes mutable fields such as hop_count, the Monad flags byte, and the CRC-16 checksum. Including mutable fields would invalidate the signature at each hop.

8. Verification Pipeline

8.1. Fast Path (Cached Verification)

When a packet arrives at an XDP verification point and the S flag is set:

1. Parse SigRef from the Monad value field.

2. Look up SigRef in sophia_pqc_sigs map.

3. If entry exists and verified == 1 (valid):

   (a) Compare entry.flow_label with packet flow label. If mismatch, proceed to slow path (possible SigRef reuse attack).

   (b) Read HashPfx from Monad register. Compare with stored SHA-256(signature)[0:2]. If mismatch, DROP and emit Anamnesis ERROR event.

   (c) Check SeqNum > last_seen_seq for this flow in Wotan memory. If not, DROP (replay detected) and emit Anamnesis ERROR event.

   (d) Update last_seen_seq in Wotan memory.

   (e) Forward packet (XDP_PASS).

4. If entry exists and verified == 2 (invalid):

   (a) DROP packet (XDP_DROP). Emit Anamnesis ERROR event.

5. If entry does not exist or verified == 0 (pending):

    (a) Proceed to slow path (Section 8.2).

8.2. Slow Path (Asynchronous Verification)

When cached verification is not available:

1. Write packet metadata to the Anamnesis ring buffer: flow key, SigRef, KeyRef, SeqNum, timestamp.

2. Apply verification policy (Section 9) to determine whether to forward or hold the packet.

3. The userspace PQC verifier daemon reads from the ring buffer and performs:

    (a) Retrieve full signature from sophia_pqc_sigs[SigRef].

    (b) Retrieve full public key from sophia_pqc_keys[KeyRef].

    (c) Construct pseudo-header from packet metadata.

    (d) Execute SLH-DSA.Verify(public_key, pseudo_header, signature).

    (e) Update sophia_pqc_sigs[SigRef].verified to 1 (valid) or 2 (invalid) via control plane map update.

4. If verification fails, the control plane MUST:

    (a) Set verified = 2 in the signature map entry.

    (b) Emit an Anamnesis ERROR event with event details.

    (c) Optionally tear down the flow via Wotan state update.

8.3. Verifier Daemon Health

The PQC verifier daemon MUST implement a health check mechanism. If the daemon becomes unresponsive:

1. The control plane MUST detect failure within 10 seconds (RECOMMENDED: 5-second health check interval with 2 missed checks triggering failure).

2. All flows with verified == 0 (pending) entries that are older than the configured timeout (default: 10 seconds) MUST be treated as verification failures.

3. The daemon MUST be automatically restarted.

4. An Anamnesis event of type ERROR MUST be emitted.

9. Verification Policies

The verification policy determines packet handling during the asynchronous verification window (typically 1-5 milliseconds for first packet of a new flow).

9.1. PESSIMISTIC Policy

Packets are held until verification completes.

* First packet of flow: buffered in BPF ring buffer.

* Released only after verified == 1.

* If verification fails or times out: DROP.

* Added latency: 1-5ms on flow establishment.

* RECOMMENDED for: ingress from untrusted networks.

9.2. OPTIMISTIC Policy

Packets are forwarded immediately; verification occurs asynchronously.

* First packet of flow: forwarded with UNVERIFIED status.

* If verification subsequently fails: tear down flow, DROP all subsequent packets.

* Risk window: 1-5ms of potentially unauthenticated traffic.

* RECOMMENDED for: internal east-west traffic where mTLS provides additional authentication.

9.3. Policy Selection

The verification policy MUST be configurable per trust boundary. Policy is configured out-of-band via the control plane (see Section 10). The compliance tier determines the default policy: SOVEREIGN defaults to PESSIMISTIC; STANDARD and ENHANCED default to OPTIMISTIC.

Implementations MUST default to PESSIMISTIC when no explicit policy is configured.

10. Compliance Tiers

Four PQC compliance tiers govern Shield processing behavior. Tiers are signaled via the Kingdom Mode bits (K1|K0) in the Monad flags byte:

K1	K0	Tier	Algorithms	Layer 2	Multi-Algo
0	0	NONE	--	--	--
0	1	STANDARD	SLH-DSA only	OFF	NO
1	0	ENHANCED	SLH-DSA+ML-DSA+FN-DSA	OPT	NO
1	1	SOVEREIGN	All three, 2-of-3	MANDATORY	YES

Table 6

10.1. Tier NONE (K1=0, K0=0)

Shield performs no PQC processing. Monad S flag is ignored. No Sophia PQC maps are loaded. No Wotan PQC state is allocated. Wire overhead: zero.

Use: development, staging, internal microservices behind a trusted perimeter where PQC is not required.

10.2. Tier STANDARD (K1=0, K0=1)

Shield verifies Layer 1 using SLH-DSA only. The primary signature standard (FIPS 205) provides the most conservative security posture: hash-based construction with no lattice assumptions, fully eBPF-native verification.

Sophia maps loaded: sophia_pqc_sigs, sophia_pqc_keys. Wotan state: basic PQC fields (0x0000FF00-0x0000FF0F). Layer 2: disabled. Default policy: OPTIMISTIC.

RECOMMENDED for: standard SaaS deployments requiring baseline post-quantum compliance.

10.3. Tier ENHANCED (K1=1, K0=0)

Shield verifies Layer 1 using any of the three signature algorithms (SLH-DSA, ML-DSA, FN-DSA). Applications MAY enable Layer 2 Sophia policy dictionaries for per-application verification requirements.

Sophia maps loaded: all PQC maps including sophia_pqc_app_policy. Wotan state: full PQC fields (0x0000FF00-0x0000FF27). Layer 2: optional (applications opt in via policy dictionary). Default policy: OPTIMISTIC.

RECOMMENDED for: enterprise deployments requiring algorithm flexibility and application-level security policy (SOC2, HIPAA, FedRAMP).

10.4. Tier SOVEREIGN (K1=1, K0=1)

Shield verifies Layer 1 using ALL THREE signature algorithms in a 2-of-3 multi-algorithm cross-verification scheme. Layer 2 is MANDATORY -- every application MUST have a Sophia policy dictionary. Pinned key enforcement is REQUIRED.

For each incoming packet, Shield:

1. Verifies the primary signature (algo indicated by algo_id).

2. Retrieves a secondary signature (different algorithm family) from the Sophia map using an extended SigRef (Section 15.4).

3. If any 2 of 3 algorithms confirm the signature, verification succeeds. If fewer than 2 verify, DROP.

Sophia maps loaded: all PQC maps, plus sophia_pqc_sovereign_sigs for secondary/tertiary signature entries. Wotan state: full PQC fields plus sovereign audit fields. Layer 2: mandatory. Default policy: PESSIMISTIC. Anamnesis: every verification result emits an audit event.

RECOMMENDED for: government, defense, classified environments requiring FIPS 140-3 Level 4 compliance and crypto-agility against single-algorithm compromise.

10.5. Tier Transitions

Tier changes are effected via the control plane API:

1. Control plane updates the Kingdom Mode bits in the Monad configuration.

2. Shield hot-reloads Sophia maps appropriate for the new tier.

3. Wotan PQC state addresses are allocated or deallocated as required.

4. Transition is effective on the next packet -- zero downtime.

Tier downgrades (e.g., SOVEREIGN to STANDARD) MUST emit an Anamnesis WARNING event. Tier upgrades are silent.

11. Key Lifecycle Management

11.1. Key Generation

SLH-DSA key pairs MUST be generated using a CSPRNG conforming to NIST SP 800-90A [SP80090A]. For deployments requiring FIPS 140-3 validation, key generation SHOULD be performed within a validated Hardware Security Module (HSM).

11.2. Key Rotation

Key rotation is signaled via the key_epoch field in the Sophia PQC Public Key Map entry. When a key is rotated:

1. New key pair is generated and loaded into sophia_pqc_keys under a new KeyRef with incremented key_epoch.

2. New signatures are computed for active flows and loaded into sophia_pqc_sigs under new SigRef values.

3. A grace period (RECOMMENDED: 60 seconds) allows in-flight packets signed with the old key to complete verification.

4. After the grace period, old key and signature entries MAY be evicted from Sophia maps.

11.3. Key Revocation

Immediate key revocation is supported via the key_revoke flow action (action ID 0x12). When a key_revoke action is triggered:

1. All signature map entries referencing the revoked KeyRef MUST have their verified field set to 2 (invalid).

2. All subsequent packets referencing the revoked KeyRef MUST be dropped.

3. An Anamnesis event MUST be emitted.

12. Wotan Integration

The Wotan per-flow memory model [WOTAN] provides storage for PQC authentication state:

12.1. Per-Flow PQC State

Within each flow's 64KB Wotan address space, the following addresses are reserved for PQC state:

Address	Size	Field
0x0000FF00	4	last_seen_seq (last verified sequence number)
0x0000FF04	1	pqc_verified (0=no, 1=yes, 2=failed)
0x0000FF05	1	pqc_algo_id (SLH-DSA parameter set)
0x0000FF06	1	pqc_key_epoch (current key epoch)
0x0000FF07	1	reserved
0x0000FF08	4	pqc_verify_count (number of verifications)
0x0000FF0C	4	pqc_fail_count (number of failures)
0x0000FF10	8	pqc_key_fp (truncated SHA-256 of public key)
0x0000FF18	8	pqc_verify_ts (verification timestamp, nanos)
0x0000FF20	4	pqc_key_created (key creation epoch seconds)
0x0000FF24	4	pqc_app_policy_id (Layer 2 policy ref, 0=none)

The addresses 0x0000FF10-0x0000FF27 support Layer 2 application policy verification (Section 14). pqc_key_fp and pqc_verify_ts are written by Shield at ingress after successful Layer 1 verification. pqc_app_policy_id is written by the application to record which policy was applied (audit trail).

These fields persist after Monad header stripping, providing the authoritative PQC state for internal kingdom operations.

12.2. Sequence Number Management

The SeqNum field provides replay resistance. Wotan stores the last verified sequence number per flow at address 0x0000FF00.

For each packet with S flag set:

1. Read last_seen_seq via bpf_wotan_read(flow_label, 0x0000FF00, &last_seq, 4).

2. If packet.SeqNum <= last_seen_seq: DROP (replay).

3. If packet.SeqNum > last_seen_seq: update via bpf_wotan_cas(flow_label, 0x0000FF00, last_seen_seq, packet.SeqNum).

4. If CAS fails (concurrent update): re-read and retry (maximum 3 attempts before DROP).

13. Shield Processing Rules (Header Stripping)

Shield operates as the perimeter gateway between untrusted external networks and the internal kingdom. PQC wire-level authentication is EXCLUSIVELY a perimeter concern. Monad Hop-by-Hop extension headers are stripped at ingress and re-stamped at egress. Internal kingdom traffic carries zero PQC wire overhead.

13.1. Ingress (Untrusted to Kingdom)

When Shield receives a packet from an untrusted source with the S flag set:

1. Validate Monad CRC-16 per [MONAD].

2. Extract SigRef, KeyRef, SeqNum, HashPfx from value field.

3. Execute verification pipeline (Section 8).

4. If verification succeeds or policy is OPTIMISTIC:

   (a) Persist verification result to Wotan per-flow PQC state (Section 12): pqc_verified, pqc_algo_id, pqc_key_epoch, last_seen_seq.

   (b) Pin the key fingerprint from sophia_pqc_keys[KeyRef] into Wotan address 0x0000FF10 (8 bytes, truncated SHA-256).

   (c) Strip the Monad Hop-by-Hop extension header from the packet. The internal kingdom network does not process or forward PQC wire metadata.

   (d) Forward the stripped packet into the kingdom.

5. If verification fails: DROP, emit DEATH event.

13.2. Egress (Kingdom to External)

When Shield transmits a packet to an external network:

1. Read Wotan per-flow PQC state. If pqc_verified == 1:

   (a) Re-stamp a fresh Monad Hop-by-Hop extension header with current SigRef, KeyRef, incremented SeqNum, and recomputed HashPfx.

   (b) Set the S flag in the Monad flags byte.

2. Kingdom Mode bits MUST be zeroed (per [MONAD]).

3. External receivers without Unheaded support will ignore the S flag and Monad value field.

13.3. Internal Transit (No PQC Wire Overhead)

Within the kingdom, after Shield ingress header stripping:

1. Packets do NOT carry Monad HbH extension headers.

2. PQC verification is NOT repeated at internal hops.

3. The authoritative PQC authentication state resides in Wotan per-flow memory. Applications and internal services read Wotan state -- not wire headers -- to determine authentication status.

4. This design ensures:

   (a) Zero per-packet PQC overhead on internal links.

   (b) Threat surface for PQC-specific attacks (SigRef exhaustion, cache poisoning, timing oracles) is confined to the Shield perimeter.

   (c) Internal lateral movement cannot exploit PQC wire attack vectors -- the headers do not exist.

14. Application-Level Policy Verification (Layer 2)

Layer 2 verification is OPTIONAL. It enables user applications to define and enforce their own PQC authentication requirements independently of the wire-level infrastructure.

14.1. Sophia Application Policy Dictionary

Applications MAY define a Sophia dictionary containing PQC policy fields. The dictionary is loaded into a Sophia BPF map and consumed by the application at runtime.

```
Map name:      sophia_pqc_app_policy
Key type:      uint32 (application_id)
Value type:    struct sophia_pqc_policy
Max entries:   Configurable (RECOMMENDED: 4,096)
Flags:         BPF_F_RDONLY_PROG
Pinning:       /sys/fs/bpf/sophia/pqc_app_policy
```

The policy structure:

```
struct sophia_pqc_policy {
    uint8_t  min_security_level;  /* NIST level: 1, 3, or 5        */
    uint8_t  require_pinned_key;  /* 0=no, 1=yes                   */
    uint8_t  num_allowed_algos;   /* Count of allowed algo_ids     */
    uint8_t  reserved;
    uint32_t max_key_age_sec;     /* Maximum key epoch age         */
    uint8_t  allowed_algos[12];   /* Up to 12 allowed algo_id vals */
    uint8_t  pinned_fp[32];       /* Expected key fingerprint      */
                                  /* (if require_pinned_key == 1)  */
};
```

Field definitions:

min_security_level (1 octet): Minimum NIST post-quantum security level the application requires. Flows authenticated with a weaker parameter set MUST be rejected by the application. Mapping: algo_id 0x01-0x02,0x07-0x08 to Level 1; 0x03-0x04,0x09-0x0A to Level 3; 0x05-0x06,0x0B-0x0C to Level 5.

require_pinned_key (1 octet): If set to 1, the application MUST compare the flow's key fingerprint (from Wotan address 0x0000FF10) against pinned_fp. Mismatch leads to rejection.

num_allowed_algos (1 octet): Number of valid entries in the allowed_algos array. If 0, all algorithms are accepted (policy only checks min_security_level).

max_key_age_sec (4 octets): Maximum age of the key epoch in seconds. The application reads pqc_key_epoch from Wotan and compares against the key's creation timestamp in sophia_pqc_keys. Keys older than this value lead to rejection.

allowed_algos (12 octets): Array of acceptable algo_id values. If

the flow's pqc_algo_id (Wotan 0x0000FF05) is not in this set, the application rejects the flow.

pinned_fp (32 octets): Expected SHA-256 fingerprint of the public key. Only checked when require_pinned_key == 1.

14.2. Layer 2 Verification Procedure

When an application performs Layer 2 verification:

1. Read pqc_verified from Wotan address 0x0000FF04. If not 1 (valid), reject. Layer 1 MUST succeed first.

2. Read pqc_algo_id from Wotan address 0x0000FF05.

3. Look up the application's policy from sophia_pqc_app_policy using the application_id as key.

4. If policy.num_allowed_algos > 0: verify pqc_algo_id is in policy.allowed_algos[]. If not found, reject.

5. Map pqc_algo_id to NIST security level. If level < policy.min_security_level, reject.

6. If policy.require_pinned_key == 1: read key fingerprint from Wotan address 0x0000FF10 (8 bytes). Compare against policy.pinned_fp[0:8]. If mismatch, reject.

7. If policy.max_key_age_sec > 0: read pqc_key_epoch from Wotan address 0x0000FF06. Look up key creation timestamp from sophia_pqc_keys. If (now - creation) > max_key_age_sec, reject.

8. All checks pass: accept flow at application layer.

14.3. Layer 2 Independence

Layer 2 operates entirely on Wotan per-flow state and Sophia policy dictionaries. It has NO dependency on Monad wire headers (which have been stripped at Shield ingress). This means:

(a) Layer 2 can be added or modified without any wire protocol changes.

(b) Different applications on the same host MAY enforce different policies for the same flow.

(c) Layer 2 is a pure application-space concern. The infrastructure (Shield, XDP, Monad) is unaware of it.

(d) Enterprise customers can define arbitrarily strict policies without affecting the performance of Layer 1 wire-level authentication.

14.4. Sophia Dictionary Definition Format

Applications define their policy using standard Sophia dictionary syntax. Example enterprise policy:

```
dictionary "enterprise-auth-policy" {
    field pqc_min_security_level  uint8   = 3;
    field pqc_require_pinned_key  uint8   = 1;
    field pqc_max_key_age_sec     uint32  = 86400;
    field pqc_allowed_algos       uint8[] = [0x03, 0x04, 0x09, 0x0A];
    field pqc_pinned_fp           bytes32 = 0xa1b2c3...;
}
```

This policy requires: NIST Level 3 minimum, specific SHA2/SHAKE-192 algorithms only, key rotation within 24 hours, and a pinned key fingerprint. Any flow not meeting ALL requirements is rejected at the application layer, even if Layer 1 wire authentication passed.

15. Multi-Algorithm Considerations

The inclusion of three distinct signature algorithm families introduces architectural constraints that implementations MUST address.

15.1. eBPF Compatibility Matrix

Not all PQC algorithms can execute entirely within the eBPF/XDP verification pipeline. The following matrix governs where verification occurs:

Algorithm	Verify in eBPF?	Sign in eBPF?	Constraint
SLH-DSA	YES	NO	Hash-only, integer
ML-DSA	YES	NO	Integer NTT mod q
FN-DSA	PARTIAL*	NO	Float in signing

Table 7

*FN-DSA verification performs integer NTT modular arithmetic (mod q=12289) plus L2-norm and infinity-norm checks. These operations are integer-only and fit within the BPF verifier's [RFC9669] 1,000,000 instruction budget (~21,000 instructions estimated). However, implementations MAY choose to route FN-DSA verification to userspace via bpf_kfunc upcall for simplicity.

FN-DSA signing requires IEEE-754 binary64 floating-point for: FFT expansion of private basis, LDL tree decomposition, and discrete Gaussian sampling. eBPF does not support floating-point operations. FN-DSA signing MUST occur in a dedicated userspace daemon.

15.2. FN-DSA Signing Daemon

When FN-DSA is enabled (Tier ENHANCED or SOVEREIGN), a dedicated signing daemon MUST be deployed:

(a) The daemon MUST run with process isolation (separate cgroup, network namespace, dedicated CPU cores via isolcpus).

(b) The daemon MUST use a constant-time FN-DSA implementation to mitigate timing side channels (see Security Considerations Section 18.11).

(c) FN-DSA mandates randomized signing only (NIST FIPS 206). Deterministic signing is PROHIBITED due to floating-point reproducibility concerns that could leak private key information.

(d) The daemon MUST validate entropy source quality at startup. If system entropy is insufficient (< 256 bits available), the daemon MUST refuse to start and emit an Anamnesis CRITICAL event.

(e) Communication between Shield and the signing daemon MUST use an authenticated channel (Unix domain socket with SO_PEERCRED verification).

15.3. Algorithm Negotiation

When multiple algorithms are available (Tier ENHANCED or SOVEREIGN), the control plane selects the algorithm per flow based on:

1. Peer capability advertisement (if Unheaded-to-Unheaded).

2. Administrative policy (Sophia app policy dictionary).

3. Performance preference (FN-DSA for minimum bandwidth, ML-DSA for fastest verification, SLH-DSA for maximum conservatism).

The selected algo_id is written to the Sophia signature map entry and persisted in Wotan per-flow state. Algorithm selection MUST NOT change mid-flow.

15.4. Sovereign Multi-Signature Layout

In Tier SOVEREIGN, each flow carries signatures from at least two distinct algorithm families. The Sophia signature map entry is extended:

```
struct sophia_pqc_sovereign_entry {
    uint8_t  primary_algo_id;     /* Primary algorithm            */
    uint8_t  secondary_algo_id;   /* Secondary (different family) */
    uint8_t  tertiary_algo_id;    /* Tertiary (0x00 if 2-of-3)    */
    uint8_t  consensus;           /* Bitfield: b0=pri b1=sec b2=ter */
    uint32_t primary_sigref;      /* SigRef for primary sig       */
    uint32_t secondary_sigref;    /* SigRef for secondary sig     */
    uint32_t tertiary_sigref;     /* SigRef for tertiary sig      */
};
```

The consensus field tracks which algorithms have verified successfully. When popcount(consensus) >= 2, the packet is authenticated. This ensures that compromise of any single algorithm family does not break authentication.

16. KEM Integration (Key Establishment)

The KEM algorithms (ML-KEM, HQC) are used for key establishment between Shield instances, NOT for per-packet authentication.

16.1. Shield-to-Shield Tunnel Keys

When two Shield instances establish a PQC-authenticated tunnel:

1. Initiator generates an ML-KEM or HQC encapsulation using the responder’s public KEM key.

2. Ciphertext is transmitted via the Sophia control channel (not in Monad HbH headers -- ciphertexts are too large).

3. Responder decapsulates to derive a shared secret.

4. Shared secret is used to derive per-flow signing keys via HKDF-SHA256.

16.2. KEM Algorithm Selection

ML-KEM (FIPS 203) is RECOMMENDED as the primary KEM due to its small ciphertext (768-1,568 bytes) and fast performance.

HQC (FIPS 207) SHOULD be available as a non-lattice backup. If lattice-based assumptions are compromised (affecting both ML-KEM and ML-DSA), HQC provides code-based diversity.

KEM algorithm identifiers use the 0x80-0x9F range in the algo_id registry (Section 17). KEM entries appear in Sophia key maps but do NOT appear in per-packet Monad headers.

17. IANA Considerations

17.1. PQC Algorithm Registry

This document requests IANA to create a new registry: "Unheaded PQC Algorithm Identifiers"

Registration Policy: Specification Required [RFC8126]

Value	Description	FIPS	Reference
0x00	Reserved	--	This document
0x01	SLH-DSA-SHA2-128s	205	This document
0x02	SLH-DSA-SHA2-128f	205	This document
0x03	SLH-DSA-SHA2-192s	205	This document
0x04	SLH-DSA-SHA2-192f	205	This document
0x05	SLH-DSA-SHA2-256s	205	This document
0x06	SLH-DSA-SHA2-256f	205	This document
0x07	SLH-DSA-SHAKE-128s	205	This document
0x08	SLH-DSA-SHAKE-128f	205	This document
0x09	SLH-DSA-SHAKE-192s	205	This document
0x0A	SLH-DSA-SHAKE-192f	205	This document
0x0B	SLH-DSA-SHAKE-256s	205	This document

0x0C	SLH-DSA-SHAKE-256f	205	This document
0x0D-0x0F	Reserved (SLH-DSA)	205	This document
0x10	ML-DSA-44	204	This document
0x11	ML-DSA-65	204	This document
0x12	ML-DSA-87	204	This document
0x13-0x1F	Reserved (ML-DSA)	204	This document
0x20	FN-DSA-512	206	This document
0x21	FN-DSA-1024	206	This document
0x22-0x7F	Reserved (sigs)	--	This document
0x80	ML-KEM-512	203	This document
0x81	ML-KEM-768	203	This document
0x82	ML-KEM-1024	203	This document
0x83-0x8F	Reserved (ML-KEM)	203	This document
0x90	HQC-128	207	This document
0x91	HQC-192	207	This document
0x92-0xFE	Unassigned	--	
0xFF	Reserved	--	This document

Table 8

17.2. Monad Flags Registry Update

This document updates the Monad Flags Registry to formally define the semantics of the S (Signed) flag (bit 3) when used with PQC authentication:

S = 1: The Monad value field contains PQC authentication references as defined in Section 5 of this document.

17.3. Sophia Map Type Registry Update

This document registers three new Sophia map types:

Map Type	Name	Reference
0x10	PQC Signatures	This document
0x11	PQC Public Keys	This document
0x12	PQC App Policies	This document
0x13	PQC Sovereign Sigs	This document
0x14	PQC KEM Keys	This document

Table 9

18. Security Considerations

18.1. Signature-by-Reference Trust Model

The signature-by-reference scheme relocates the full signature from the wire to kernel-resident BPF maps. The security of this scheme depends on:

(a) The Sophia PQC maps being write-protected from the data plane (BPF_F_RDONLY_PROG). If an attacker gains write access to the maps, they can substitute arbitrary signatures.

(b) The SigRef-to-signature binding being immutable for the lifetime of the flow. SigRef values MUST NOT be reused within the same key epoch.

(c) The control plane being trusted. The control plane is the sole writer to Sophia maps. Compromise of the control plane compromises all authentication.

18.2. Replay Resistance

The SeqNum field in the Monad register provides replay resistance. Without SeqNum, an attacker who captures a legitimate packet could replay it indefinitely, as the SigRef would still reference a valid cached verification result.

The SeqNum MUST be included in the signed pseudo-header. The Wotan per-flow last_seen_seq counter MUST be checked at each verification point. Packets with SeqNum less than or equal to last_seen_seq MUST be dropped.

Implementations SHOULD use a sliding window (RECOMMENDED: 64 packets) rather than strict monotonic ordering to tolerate minor packet reordering.

18.3. HashPfx Collision Probability

The 16-bit HashPfx provides a fast integrity check with collision probability of 1/65,536. This is NOT a security mechanism -- it is an optimization to detect accidental SigRef corruption without performing a full map lookup.

Deliberate attacks against HashPfx (preimage or collision) are trivial given its 16-bit size. Security depends entirely on SLH-DSA verification, not on HashPfx.

18.4. Asynchronous Verification Window

The OPTIMISTIC verification policy creates a window of 1-5 milliseconds during which unverified packets may be forwarded. Deployments with strict authentication requirements MUST use the PESSIMISTIC policy.

The OPTIMISTIC policy is acceptable when additional authentication mechanisms (e.g., mTLS) are in place and the PQC authentication serves as a defense-in-depth layer.

18.5. Memory Exhaustion

An attacker may attempt to exhaust Sophia map capacity by generating flows with unique SigRef values. Implementations MUST:

(a) Enforce a maximum map size.

(b) Implement LRU eviction for verified entries.

(c) Rate-limit new SigRef allocations from untrusted sources.

(d) Monitor map utilization and alert when approaching capacity (RECOMMENDED: alert at 80%).

18.6. CRC-16 Is Not Cryptographic

As noted in [MONAD], the CRC-16/CCITT checksum detects accidental corruption, not deliberate tampering. PQC authentication via SLH-DSA provides tamper detection. The two mechanisms are complementary: CRC-16 catches transmission errors; SLH-DSA catches deliberate modification.

18.7. Side-Channel Considerations

SLH-DSA verification timing may vary based on signature content. Implementations MUST use constant-time comparison for HashPfx and fingerprint checks. The asynchronous verification architecture inherently mitigates timing side channels in the data plane, as verification occurs off the packet forwarding path.

18.8. Quantum Security Level Selection

The choice of SLH-DSA parameter set determines the quantum security level. NIST Level 1 (SLH-DSA-128s) is RECOMMENDED for most deployments as it provides 128-bit security against quantum attacks with the smallest signature size (7,856 bytes).

Deployments handling classified or long-term sensitive data SHOULD use Level 3 (SLH-DSA-192s) or Level 5 (SLH-DSA-256s).

18.9. Header Stripping and Perimeter Isolation

The header stripping design confines PQC wire-level attack vectors to the Shield perimeter. This fundamentally limits the blast radius of several attack classes:

(a) SigRef exhaustion (PQC-001): Only external traffic carries SigRef values. Internal lateral movement cannot exploit SigRef-based amplification -- the headers are stripped.

(b) Cache poisoning (PQC-002): SigRef prediction attacks are only possible from external sources. Internal services never reference the signature cache via wire headers.

(c) Timing oracles (PQC-003): Verification timing differences are only observable from outside the perimeter. Internal paths see constant-time Wotan reads.

However, header stripping does NOT mitigate:

(d) Control plane compromise (PQC-008): A compromised control plane can poison Wotan state directly, bypassing wire-level authentication entirely. Defense-in-depth measures (key pinning, audit trails, HSM integration) remain critical.

18.10. Dual-Layer Verification Security Properties

The dual-layer model provides defense-in-depth:

(a) Layer 1 alone is sufficient for infrastructure-grade authentication. Layer 2 is additive security.

(b) Layer 2 cannot weaken Layer 1. A flow rejected by Layer 1 never reaches the application. Layer 2 can only further restrict what Layer 1 has already accepted.

(c) Layer 2 policy dictionaries are stored in BPF_F_RDONLY_PROG maps. Applications cannot modify their own policies at runtime via the data plane. Policy changes require control plane intervention.

(d) Different applications MAY enforce different Layer 2 policies on the same flow. Application A may accept a Level 1 authenticated flow while Application B on the same host rejects it for not meeting Level 3 requirements. This is by design -- applications own their security posture.

(e) Layer 2 verification adds negligible overhead: Wotan reads (~50ns each) plus Sophia policy map lookup (~50-100ns). Total Layer 2 cost: ~200-300ns per flow establishment.

18.11. FN-DSA Floating-Point Side Channels (PQC-009)

FN-DSA signing requires IEEE-754 binary64 floating-point operations for FFT expansion, LDL tree decomposition, and discrete Gaussian sampling. The "Do Not Disturb a Sleeping Falcon" paper (Eurocrypt 2025) demonstrates that timing variations in these operations can leak private key bits after approximately 2^26 observed signatures.

Mitigations:

(a) The FN-DSA signing daemon MUST use a constant-time implementation as specified in FIPS 206.

(b) The signing daemon MUST run on isolated CPU cores (isolcpus) with FPU state pinned (no context-switch FPU save/restore observable by other processes).

(c) Memory used by the signing daemon MUST be mlocked to prevent swapping.

(d) NIST requires randomized signing for FN-DSA. This prevents the deterministic signature patterns that enable the Eurocrypt 2025 attack.

Header stripping does NOT mitigate this finding -- the signing daemon is internal to the kingdom.

18.12. Userspace-Kernel Boundary Attacks (PQC-010)

When FN-DSA verification is routed to userspace via bpf_kfunc upcall, a time-of-check-to-time-of-use (TOCTOU) window exists between eBPF verification and application consumption.

Mitigations:

(a) Signature verification results MUST be pinned to the flow via a SHA-256 hash stored in Wotan per-flow state. Applications compare the pinned hash, not the raw result.

(b) The bpf_kfunc interface MUST authenticate the calling BPF program via BTF type verification.

(c) Ring buffer communication between kernel and userspace MUST use monotonic sequence numbers to prevent replay.

Header stripping partially mitigates this -- the TOCTOU window exists only at the Shield perimeter.

18.13. Algorithm Confusion and Downgrade (PQC-011)

With three signature algorithm families, an attacker may attempt to force use of a weaker or compromised algorithm by manipulating the algo_id field in the Monad register.

Mitigations:

(a) Shield MUST validate that algo_id matches the algo_id stored in the Sophia signature map entry. Mismatch leads to DROP.

(b) The compliance tier MUST restrict which algo_id values are accepted. Tier STANDARD accepts only SLH-DSA (0x01-0x0C). Packets with ML-DSA or FN-DSA algo_id values at Tier STANDARD MUST be dropped.

(c) Algorithm selection MUST NOT change mid-flow. If a packet arrives with a different algo_id than the flow's established algorithm, DROP and emit Anamnesis ERROR.

(d) Tier SOVEREIGN requires 2-of-3 consensus across different algorithm families, preventing single-algorithm downgrade.

Header stripping mitigates this at the perimeter -- algo_id manipulation is only possible from external sources.

18.14. Entropy Requirements for FN-DSA (PQC-013)

FN-DSA signing requires approximately 40,448 bits of randomness per signature (79 bits x 512 coefficients). Insufficient entropy in the discrete Gaussian sampling process can cause signatures to leak private key information.

Mitigations:

(a) The signing daemon MUST verify entropy source quality at startup using RDSEED or equivalent hardware RNG.

(b) If the kernel entropy pool drops below 256 bits, the signing daemon MUST pause signing and emit an Anamnesis CRITICAL event.

(c) Deployments in virtual machines or containers MUST ensure virtio-rng or equivalent entropy passthrough is configured.

(d) SLH-DSA does not have this vulnerability -- its hash-based construction uses deterministic derivation from the message and secret key. This is one reason SLH-DSA is the RECOMMENDED default at Tier STANDARD.

18.15. Compliance Tier Security Boundaries

Each compliance tier establishes distinct security guarantees:

(a) Tier NONE: No PQC guarantees. Packets are not authenticated. Suitable only for environments where PQC is not required.

(b) Tier STANDARD: Single-algorithm (SLH-DSA) authentication at Layer 1. Vulnerable to SLH-DSA-specific compromise but provides baseline post-quantum protection.

(c) Tier ENHANCED: Multi-algorithm availability but single-algorithm enforcement per flow. Provides algorithm agility but not simultaneous cross-verification.

(d) Tier SOVEREIGN: Multi-algorithm cross-verification (2-of-3). Survives compromise of any single algorithm family. MANDATORY Layer 2 enforcement and audit trail. Maximum assurance.

Tier downgrades MUST be treated as security events. A downgrade from SOVEREIGN to STANDARD removes multi-algorithm protection.

19. References

19.1. Normative References

[FIPS203] National Institute of Standards and Technology, "Module-Lattice-Based Key-Encapsulation Mechanism Standard", FIPS 203, August 2024.

[FIPS204] National Institute of Standards and Technology, "Module-Lattice-Based Digital Signature Standard", FIPS 204, August 2024.

[FIPS205] National Institute of Standards and Technology, "Stateless Hash-Based Digital Signature Standard", FIPS 205, August 2024.

[FIPS206] National Institute of Standards and Technology, "FFT over NTRU-Lattice-Based Digital Signature Standard", FIPS 206, 2025.

[FIPS207] National Institute of Standards and Technology, "HQC Key-Encapsulation Mechanism Standard", FIPS 207, 2025.

[MONAD] Bellis, S., "The Unheaded Protocol Foundation", Work in Progress, Internet-Draft, draft-bellis-unheaded-protocol-foundation-00, March 2026, <https://datatracker.ietf.org/doc/html/draft-bellis-unheaded-protocol-foundation-00>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, June 2017, <https://www.rfc-editor.org/info/rfc8126>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8200] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", STD 86, RFC 8200, July 2017, <https://www.rfc-editor.org/info/rfc8200>.

[SOPHIA] Bellis, S., "Sophia Dictionary Specification for the Unheaded Protocol", Work in Progress, Internet-Draft, draft-bellis-unheaded-sophia-dictionary-00, March 2026, <https://datatracker.ietf.org/doc/html/draft-bellis-unheaded-sophia-dictionary-00>.

[WOTAN] Bellis, S., "Wotan Memory Model for the Unheaded Protocol", Work in Progress, Internet-Draft, draft-bellis-unheaded-wotan-memory-00, March 2026, <https://datatracker.ietf.org/doc/html/draft-bellis-unheaded-wotan-memory-00>.

19.2. Informative References

[RFC9669] Thaler, D., Ed., "BPF Instruction Set Architecture (ISA)", RFC 9669, October 2024, <https://www.rfc-editor.org/info/rfc9669>.

[SP80090A] National Institute of Standards and Technology, "Recommendation for Random Number Generation Using Deterministic Random Bit Generators", NIST SP 800-90A Rev. 1, June 2015.

Appendix A. Algorithm Parameter Set Selection Guide

Cross-algorithm comparison for deployment planning:

Requirement	Recommended Algo	Sig Size	Verify Time	Tier
Most conservative	SLH-DSA-SHA2-128s	7,856 B	~2ms	STANDARD
Fast verification	ML-DSA-44	2,420 B	~0.1ms	ENHANCED
Minimum bandwidth	FN-DSA-512	666 B	~0.2ms	ENHANCED
High security (hash)	SLH-DSA-SHA2-256s	29,792 B	~5ms	STANDARD
High security (lattice)	ML-DSA-87	4,627 B	~0.3ms	ENHANCED
Government / CNSA 2.0	SLH-DSA-SHA2-192s+	16,224 B	~3ms	SOVEREIGN
Maximum assurance	2-of-3 (all algos)	varies	~5-7ms	SOVEREIGN

Table 10

Algorithm family trade-offs:

SLH-DSA (FIPS 205): Most conservative. Hash-based -- no lattice assumptions. Largest signatures but simplest trust model. Fully eBPF-native. RECOMMENDED for Tier STANDARD.

ML-DSA (FIPS 204): Best performance/size ratio. Lattice-based with integer NTT. Fully eBPF-native. If lattice assumptions hold, this is the optimal choice for high-throughput deployments.

FN-DSA (FIPS 206): Smallest signatures (666-1,280B). Lattice-based with floating-point signing. Requires userspace signing daemon. Best for bandwidth-constrained links. Carries additional side-channel risk (see Section 18.11).

For the signature-by-reference scheme, signature size impacts Sophia map memory, not wire overhead. All algorithms use the same 12-byte Monad value layout.

Appendix B. Performance Analysis

Estimated per-packet overhead at XDP:

Operation	Time	Notes
Sophia map lookup	50-100ns	Hash map, kernel memory
Cached verification read	~10ns	Single byte read
HashPfx comparison	~5ns	2-byte constant-time
SeqNum Wotan read	~50ns	bpf_wotan_read
SeqNum Wotan CAS	~100ns	bpf_wotan_cas
Total fast path	~215-265ns	Per packet, cached

Table 11

First-packet slow path (async, one-time per flow):

Operation	Time	Notes
Ring buffer write	~200ns	Zero-copy to userspace
SLH-DSA verification	0.5-5ms	Parameter set dependent
Map update with result	~100ns	Control plane write
Total slow path	~1-5ms	One-time per flow

Table 12

Memory overhead per concurrent flow (SLH-DSA-SHA2-128s):

Component	Size per flow	100K flows	1M flows
Signature entry	~8 KB	~800 MB	~8 GB
Key entry	~100 B	~10 MB	~100 MB
Wotan PQC state	16 B	~1.6 MB	~16 MB
Total	~8.1 KB	~812 MB	~8.1 GB

Table 13

Appendix C. Test Vectors

(To be provided in a future revision with complete SLH-DSA signing and verification examples using the pseudo-header format defined in Section 7.)

Author’s Address

Stevie Bellis
Unheaded
Email: stevie@bellis.tech