PQSAFE-IL5-HYBRID-v1

A crypto-agile post-quantum envelope for DoD-grade file storage and transfer (a DoD SAFE compatibility profile)

Zach Kelling

Lux Industries

Draft June 20, 2026

PQSAFE-IL5-HYBRID-v1

A crypto-agile post-quantum envelope for DoD-grade
file storage and transfer (a DoD SAFE compatibility profile)

Lux Industries Inc.
`security@lux.network`
`safe@apps.mil` (proposed)

Draft June 20, 2026

Abstract

PQSAFE-IL5-HYBRID-v1 is a crypto-agile envelope encryption profile for DoD file storage and transfer designed to sit *above* any approved back-end (DoD SAFE, S3-compatible object storage, SharePoint, MinIO, content-addressed stores, or the Lux storage chain) while ensuring those back-ends never see plaintext, sender identity, recipient identity, filenames, or classified metadata. The profile uses *hybrid* [7] post-quantum cryptography: NIST-standardised **ML-KEM-768** (FIPS 203 [1]) combined with classical X25519 via the IETF **X-Wing** construction [7]; **ML-DSA-65** (FIPS 204 [2]) combined with classical Ed25519 for sender authentication; and either AES-256-GCM (NIST CNSA 2.0 [6]) or ChaCha20-Poly1305 [8] for bulk AEAD chunk encryption with sequence-numbered nonces and per-message forward secrecy. Bulk data is never wrapped under the PQ key directly: the PQ layer protects the *file encryption key* (DEK), which protects the file. Sender identity is bound through a hybrid PQ signature over a canonical CBOR/COSE manifest [9, 10]. **All X-Wing and hybrid-identity primitives in PQSAFE come from the canonical Lux package luxfi/zwing** [13] — the same package provides the Z-Wing secure channel for TCP/IP transport, so the bytes that encrypt a file and the bytes that protect a session ride the same vetted code path (one and only one X-Wing implementation; one and only one hybrid-signature surface). Optional Reticulum Network Stack (RNS, LP-9701 [16]) routing covers disconnected, mesh, or LoRa deployments via the PQ-RNS profile in luxfi/pqrns [15], which itself rides zwing for identity + signing. The whole stack is FIPS/CNSA-2.0 aligned where validated modules exist and is explicitly *crypto-agile by design*: every algorithm, parameter set, KDF, AEAD, signature, and manifest schema is versioned, so the profile can evolve to PQ-only when DoD policy and PKI catch up. For everything the stack cannot send end-to-end (replication, backup, legacy ingest), the same envelope is re-keyed at rest using **luxfi/age** (age + ML-KEM-768 recipients [12]).

Contents

1 Threat model and design principles

The defensible architecture for DoD is not "replace DoD SAFE with blockchain/PQ magic." It is:

> *Encrypt the file before it enters any storage or transfer system. Sign the manifest. Audit every custody event. Keep keys off the storage system. Make the PQ layer hybrid until policy, FIPS modules, and PKI catch up.*

PQSAFE-IL5-HYBRID-v1 inverts the usual "trust the transport" design: DoD SAFE, object storage, Lux storage, or any other back-end is a dumb ciphertext bucket. The crypto boundary is the sender's client, not the storage system's TLS terminator.

1.1 Adversaries

- **Network adversary**: passive recorder + active store-and-decrypt-later attacker with future quantum computer. Defeated by ML-KEM-768 hybrid key wrap (X-Wing) on the envelope and ML-KEM-768 hybrid session establishment on the channel (Z-Wing).
- **Storage adversary**: anyone with read access to the back-end (DoD SAFE database admin, S3 reader, ledger reader). Sees only ciphertext and minimal metadata. Cannot reconstruct plaintext without the recipient's long-term private key.
- **Insider with key material**: a recipient may decrypt their own copy. The envelope binds recipient identity into the signed manifest so an unauthorised re-distribution is detectable; revocation is enforced through manifest expiry and a hash-only revocation registry.
- **Endpoint compromise**: out of scope for the envelope. Mitigated by HSM-bound long-term keys, hardware roots of trust, and unattended-decryption being forbidden.
- **Quantum adversary (CRQC)**: the entire stack is post-quantum on the confidentiality axis through ML-KEM-768. The authenticity axis is post-quantum through ML-DSA-65 with an SLH-DSA-192f long-term root option for high-assurance signing (LMS/XMSS available where stateful signing can be safely managed, per NIST SP 800-208 [5]).

1.2 Non-goals

- PQSAFE is *not* a cross-domain solution. Classification boundaries are enforced outside the envelope.
- PQSAFE does *not* authorise recipients. It encrypts *to* recipients chosen by an external ABAC/RBAC policy engine.
- PQSAFE does *not* store sensitive metadata on a public ledger. The Lux chain holds only opaque hash commitments.

1.3 Principles

1. **Hybrid first.** Every PQ primitive is paired with a classical primitive and combined with a KDF binding so the composition is at least as strong as the stronger of the two. This is the X-Wing construction [7] for KEMs and the "classical + ML-DSA-65" detached-signature pair for signatures.
2. **Symmetric for bulk.** Multi-GB files are encrypted with AES-256-GCM (FIPS-aligned, hardware-accelerated) or ChaCha20-Poly1305 (software-friendly fallback). The PQ primitive never wraps the bulk content; it wraps the DEK.
3. **Crypto-agile.** Every primitive carries a wire-byte identifier; manifests carry a profile-version field; the verifier routes by version. No alternate suites within a profile.
4. **Keys off the storage layer.** HSM-bound KEKs wrap per-file DEKs. Recipient public keys are certificate-bound. Recovery is M-of-N threshold, never a universal master decrypt key.
5. **Audit is tamper-evident, not secret-bearing.** Only hashes of envelope manifest, sender identity certificate, and the ordered recipient set go on the audit log.
6. **Forward secrecy at the channel layer; explicit policy for stored envelopes.** Z-Wing/PQ-RNS use ephemeral X-Wing per session and AEAD nonce sequence numbers so past sessions cannot be decrypted if a long-term identity key is later compromised. Stored envelopes target long-term recipient keys and therefore require complementary key-rotation, rewrap, revocation, and recovery policies (see Section 2.3).

7. **Domain separation everywhere.** Every HKDF and every signature carries an explicit context string identifying the protocol layer and version (e.g. `PQSAFE-MANIFEST-SIG-v1`, `PQSAFE-DEK-WRAP-v1`, `PQSAFE-CHUNK-AEAD-v1`). No two layers ever derive a key from the same transcript.

2 Cryptographic suite

Layer	Primitive	Standard / source
Bulk AEAD (FIPS profile)	AES-256-GCM	FIPS 197 + SP 800-38D [4]
Bulk AEAD (software profile)	ChaCha20-Poly1305	RFC 8439 [
Hash	SHA-384 / SHA-3-384	FIPS 180-4 / FIPS 202
Merkle hash	SHA-384 over 64-KiB chunks	this pro-file
KDF	HKDF-SHA-384	RFC 5869
Key wrap (hybrid KEM)	**X-Wing** = X25519 + ML-KEM-768	IETF draft-connolly-cfrg-xwing-kem [7]
Key wrap (high-value)	X-Wing-1024 = X448 + ML-KEM-1024	this pro-file
Sender signature (hybrid)	Ed25519 + ML-DSA-65 (detached pair)	RFC 8032 + FIPS 204 [2
Long-term root signature	SLH-DSA-SHA2-192f	FIPS 205 [3
Stateful root signature (option)	LMS-SHA-256-H10	SP 800-208 [5]
Manifest format	CBOR/COSE	RFC 8949 + RFC 8152 [10]
Channel (E2E PQ)	**Z-Wing** (X-Wing + Ed25519+ML-DSA-65 + ChaCha20-Poly1305)	luxfi/zwing
File-at-rest envelope	luxfi/age + ML-KEM-768 recipient	age-encryption 12]
Mesh / LoRa transport (optional)	RNS hybrid PQ link	LP-9701 [16]

Table 1: PQSAFE-IL5-HYBRID-v1 primitive stack.

2.1 Why X-Wing and not naive concatenation

X-Wing [7] is a hybrid KEM that combines X25519 with ML-KEM-768 through a SHA3-based KDF that binds the X25519 ephemeral public key, the ML-KEM ciphertext, and a 6-byte label

‘‘\.//^’\’’ into the derived secret. The construction is IND-CCA2 secure under *either* X25519’s DH assumption or ML-KEM’s M-LWE assumption: a break of one component does not break the hybrid. Naive concatenation $s = s_{\text{X25519}} \,\|\, s_{\text{ML-KEM}}$ is *not* a KEM in the strict KEM-API sense and is vulnerable to subtle re-encapsulation and key-reuse pitfalls [7]. X-Wing fixes the API by emitting a single ciphertext that captures both component ciphertexts and a single 32-byte shared secret that’s already KDF’d.

The IETF X25519MLKEM768 hybrid (IANA codepoint `0x11ec`) used in TLS 1.3 is the same primitive at the cipher-suite level; the X-Wing draft is the standalone construction for non-TLS protocols.

2.2 Why ML-KEM-768 (not 1024) as the default

ML-KEM-768 corresponds to NIST PQ Category 3 (security comparable to AES-192) and is the production default for both Z-Wing and the file envelope. ML-KEM-1024 (Category 5, AES-256-equivalent) is used for high-value paths (recovery keys, KEK-wrapping-KEK, federation root). Category 1 (ML-KEM-512) is not used: the marginal performance gain does not justify diverging from the production default for an operationally simple stack.

2.3 Forward secrecy — transport vs. stored envelopes

Transport forward secrecy and stored-envelope forward secrecy are *different problems with different solutions*. The paper makes the distinction explicit so the security claim does not over-reach.

Transport (Z-Wing, PQ-RNS) — full hybrid FS. For every session, both peers generate fresh X25519 and ML-KEM-768 ephemeral keypairs. After each session establishes its key, the ephemeral private keys and any intermediate HKDF material are zeroed. The session key derives from

$$K_{\text{sess}} = \mathsf{HKDF\text{-}Extract\text{-}Then\text{-}Expand}\big(\,\mathsf{X25519}_{\text{eph-eph}} \,\|\, \mathsf{ML\text{-}KEM\text{-}768}_{\text{eph-eph}}\,,\ \mathsf{context} = \text{“ZWING-SESSION-v1”} \,\|\, H($$

A future compromise of either long-term key (Ed25519 *or* ML-DSA-65) lets the adversary impersonate the identity going forward *but cannot decrypt previously recorded sessions*: the ephemeral half is gone. Retrospective decryption requires recovering *both* classical and lattice ephemerals for every recorded link, which is the hybrid FS contract. Long-term keys are used *only* to authenticate the ephemeral key-exchange transcript; they never directly wrap session traffic.

Chunk AEAD — intra-file FS via HKDF re-key. Each 64-KiB chunk is encrypted with a unique nonce $(N_0 \,\|\, \text{seq})$ where N_0 is a 12-byte random nonce-seed bound into the manifest and seq is a 4-byte big-endian chunk index. The DEK itself is rotated every 2^{32} chunks (256 TiB) via HKDF re-keying chained from the manifest’s ratchet seed; compromise of a re-keyed-DEK does not decrypt chunks emitted before the most recent re-key. For interactive transfer this is irrelevant (files fit under the rotation limit); for long-lived replication streams it prevents one DEK compromise from leaking the entire history.

Stored envelopes — DEK is wrapped to long-term recipient keys, so naive FS does not apply. Each recipient’s capsule wraps the per-file DEK under the recipient’s long-term X-Wing public key. If the recipient’s long-term X-Wing *private* key is later compromised, every capsule that targeted that recipient becomes decryptable, *including capsules created before the compromise*. This is the well-known property of any recipient-targeted file-encryption scheme (PGP, age, S/MIME); it is *not* a defect of PQSAFE, but a property the operator must manage. Five complementary controls reduce the residual risk:

1. **KMS/HSM-mediated unwrap.** Recipient long-term private keys live inside an HSM (Hanzo KMS / luxfi/kms). The HSM enforces policy at unwrap time (recipient identity, manifest TTL, revocation status, optional dual-control authorisation). A leaked recipient *credential* (smartcard PIN) without HSM access does not unwrap capsules. The HSM is itself audit-logged.
2. **Short-lived recipient wrapping keys.** Recipients maintain a hierarchy: a long-term identity key (HSM-bound) and a shorter-lived wrapping subkey (rotated daily/weekly). Capsules target the wrapping subkey. Subkey expiry truncates the retrospective-decryption window if the subkey leaks.
3. **Epoch keys.** The wrapping subkey is parameterised by a monotonic epoch identifier e ($1\,\text{day}, 1\,\text{week}, 1\,\text{month}$, by policy). Capsules embed the epoch. Verifiers refuse capsules outside the active epoch window; the policy registry rotates the active set. An attacker who recovers an epoch-e key can only decrypt envelopes sealed during epoch e.
4. **Rewrap-on-access (key rotation without re-encrypt).** When the recipient's wrapping key rotates, the sender (or a delegated rewrap service) re-encapsulates the DEK for each outstanding capsule under the new recipient key. The bulk chunks are untouched. Old capsules are deleted from storage; the ciphertext is now only decryptable by the new wrapping key.
5. **Threshold release.** Sensitive historical envelopes wrap the DEK under a Shamir-shared (t, n) set of recipient keys. No single compromised recipient unwraps the DEK; t recipients must cooperate (see §4.1).

Summary security statement. *Confidentiality is provided by hybrid X-Wing key establishment (X25519 + ML-KEM-768) with HKDF domain separation per protocol layer. Authenticity is provided by detached hybrid signatures (Ed25519 + ML-DSA-65) over canonical CBOR/COSE manifests and transcript-bound capability negotiation. Bulk payload encryption uses AES-256-GCM for the FIPS-oriented profile or ChaCha20-Poly1305 for the software profile, with sequence-numbered nonces, per-object domain separation, and HKDF re-keying before nonce exhaustion. Transport forward secrecy is provided when both classical and PQ session secrets are ephemeral and erased after derivation; compromise of long-term signing or identity keys does not decrypt prior sessions but may enable impersonation until revocation. Stored envelopes use DEK wrapping and require separate key-rotation, rewrap, revocation, and recovery policy (see Section 4).*

2.4 Lamport / SLH-DSA / XMSS for the long-term root

The profile signs every manifest with the sender's hybrid Ed25519+ML-DSA-65 identity. The *anchors* of that identity – certificate authorities, policy registry roots, firmware signatures, and the safe.apps.mil federation root – are signed by SLH-DSA-192f (stateless hash-based, FIPS 205) by default. SLH-DSA does not require state tracking, which is operationally desirable for the federation root. Where state tracking can be reliably enforced (e.g., an air-gapped HSM with a counter), LMS-SHA-256-H10 or XMSS-SHA-256-H10 (SP 800-208) are alternates: smaller signatures, but stateful, so each signing operation must be journaled or signatures will fly off into compromise. For high-volume manifest signing we keep ML-DSA-65 (lattice, stateless, faster); for root-of-trust signing where the signing rate is small, the stateless hash-based scheme is the better fit.

3 Envelope format

A PQSAFE envelope is a directory or a single `.pqsafe` container file containing:

```
report.pdf.pqsafe/
  manifest.cbor       - CBOR/COSE signed manifest
  manifest.sig        - Ed25519+ML-DSA-65 detached signature
  capsules/           - one per recipient
    01.cbor           - X-Wing(recipient_pub_xwing) -> wrap(DEK)
    02.cbor
    ...
  chunks/             - bulk ciphertext
    000001.bin        - AEAD(DEK, nonce_seed||1, chunk_1)
    000002.bin
    ...
  proof/              - optional Z-Chain inclusion proof
    inclusion.bin
```

3.1 Manifest schema (CBOR/COSE)

Listing 1: Manifest schema (CBOR-decoded to JSON for readability).

```
{
  "profile":            "PQSAFE-IL5-HYBRID-v1",
  "manifest_version":   1,
  "created":            "2026-05-12T22:00:00Z",
  "ttl":                "168h",
  "policy_label":       "CUI",
  "policy_uri":         "policy://safe.apps.mil/policies/CUI-IL5-HYBRID-PQ-v1",
  "bulk_aead":          "AES-256-GCM",
  "hash":               "SHA-384",
  "kdf":                "HKDF-SHA-384",
  "kem":                "X-Wing-MLKEM768",
  "sig_classical":      "Ed25519",
  "sig_pq":             "ML-DSA-65",
  "chunk_size":         65536,
  "chunks": [
    {"index": 1, "len": 65536, "sha384": "..."},
    {"index": 2, "len": 65536, "sha384": "..."},
    ...
    {"index": 8421, "len": 14336, "sha384": "..."}
  ],
  "merkle_root":        "sha384:...",
  "nonce_seed":         "base64url:...",
  "envelope_id":        "sha384:...",          % bound AEAD AAD (see below)
  "sender": {
    "x509_chain_hash":  "sha384:...",
    "did":              "did:lux:...",
    "pq_pub_mldsa65":   "base64url:...",
    "pub_ed25519":      "base64url:..."
  },
  "recipients": [
    {
      "recipient_id":     "alice.mil",
      "pub_xwing":        "base64url:...",
      "capsule":          "capsules/01.cbor",
      "x509_chain_hash":  "sha384:..."
    },
    {
      "recipient_id":     "bob.gov",
      "pub_xwing":        "base64url:...",
      "capsule":          "capsules/02.cbor",
      "x509_chain_hash":  "sha384:..."
    }
  ],
  "audit": {
    "anchor_chain":     "lux-q",
    "audit_uri":        "https://audit.safe.apps.mil/v1/commits",
    "commit_hash":      "sha384:..."
  }
}
```

3.2 Capsule format

Each capsule wraps the per-file DEK for one recipient under that recipient's X-Wing public key.

```
capsule.cbor = {
  "v":      1,
  "kem":    "X-Wing-MLKEM768",
  "kem_ct": <encoded X-Wing ciphertext (~1184 bytes)>,
  "aad":    sha384(manifest.cbor),
  "wrap":   AES-256-GCM(DEK, K_recipient, AAD=aad)
}
```

The recipient executes:

1. Verify manifest signature (Ed25519 *and* ML-DSA-65; both must pass).
2. Look up own capsule by recipient ID.
3. $K_{\text{recipient}} \leftarrow \mathsf{X\text{-}Wing.Decap}(sk_{\text{xwing}}, \text{kem\_ct})$ – this fails if either lattice or DH leg is corrupt, guaranteeing hybrid security.
4. $\mathsf{DEK} \leftarrow \mathsf{AES\text{-}256\text{-}GCM\text{-}Open}(\text{wrap}, K_{\text{recipient}}, \text{aad})$.
5. Decrypt each chunk: $c_i \leftarrow \mathsf{AEAD.Open}(\mathsf{DEK}, \text{nonce\_seed} \,\|\, i, \text{ct}_i)$.
6. Verify Merkle root over chunk hashes matches manifest.

3.3 Envelope identifier and AEAD AAD binding

The chunk AEAD takes `envelope_id` as additional authenticated data so that ciphertext chunks cannot be lifted from one envelope and replayed inside another. To make this binding itself verifiable the identifier is computed up-front from inputs that are fully determined before chunk encryption and is then carried inside the signed manifest:

$$\texttt{envelope\_id} \;=\; \mathsf{SHA\text{-}384}\big(\,\texttt{profile} \,\|\, \texttt{version}_{32} \,\|\, \texttt{pub\_ed25519} \,\|\, \texttt{pq\_pub\_mldsa65} \,\|\, \texttt{nonce\_seed} \,\|\, \texttt{created} \,\|\, \big\|_i \,(\texttt{r}$$

At Open time the verifier recomputes `envelope_id` from the signed manifest fields and refuses if the recomputed value does not byte-match the `envelope_id` carried inside the manifest. The manifest signature already covers `envelope_id`, so any attempt to substitute a different AAD is caught either by the recomputation check or by the hybrid signature check.

3.4 Algorithm allow-list at verify

The manifest fields `hash`, `kdf`, `kem`, `sig_classical`, `sig_pq`, and `chunk_size` are not advisory. PQSAFE-IL5-HYBRID-v1 verifiers MUST refuse any envelope whose manifest carries values other than the canonical set: `SHA-384`, `HKDF-SHA-384`, `X-Wing-MLKEM768`, `Ed25519`, `ML-DSA-65`, and a `chunk_size` of 65536 bytes. New parameter sets land as a new profile string (`PQSAFE-IL5-HYBRID-v2`, ...), not as parameters inside v1. This eliminates downgrade as a class.

4 Key management

Key	Role	Custody
DEK	per-file symmetric key (256 bits)	ephemeral, never persisted on storage system
KEK	wraps DEK at rest in storage layer (optional)	HSM/KMS (Hanzo KMS / luxfi/kms)
Recipient X-Wing	per-recipient hybrid PQ KEM keypair	user smart card / TPM / HSM; classical X25519 + ML-KEM-768
Recipient signing	per-recipient hybrid sig keypair (Ed25519 + ML-DSA-65)	smart card / TPM; manifest countersignatures, optional
Sender identity	per-sender hybrid sig keypair (Ed25519 + ML-DSA-65)	smart card / TPM; ALWAYS signs manifest
Federation root	SLH-DSA-192f keypair (FIPS 205)	air-gapped HSM; signs CA certs + policy registry
Recovery quorum	M-of-N threshold of recovery shares	distributed across agencies; rebuild DEK without single decryptor

Table 2: Key roles and custody.

4.1 Recovery without a master key

PQSAFE forbids a universal master decrypt key. Recovery requires threshold cooperation:

1. DEK is Shamir-shared (t, n) at creation (typical: $t = 3, n = 5$ across agency CISOs) over $\mathsf{GF}(2^{256})$.
2. Shares are wrapped under each share-holder's recipient X-Wing public key into capsules in the envelope's `recovery/` subdirectory.
3. Reconstruction requires t share-holders to participate; no one party can decrypt unilaterally.
4. Audit log records every reconstruction event with the manifest hash, share-holder hashes, and timestamp.

This matches LP-019 (Threshold MPC) [17] and uses the same threshold signing kernel that the Lux primary network deploys for QuasarCert: luxfi/corona (Module-LWE, Ringtail-derived) and luxfi/pulsar (Module-LWE) threshold ML-DSA. The *file* recovery quorum and the *chain* finality quorum share no key material, but they share the same Pedersen DKG + proactive resharing lifecycle.

5 Transport: Z-Wing and RNS

PQSAFE envelopes can be transferred over any channel; the envelope is self-protecting. Two transport profiles are recommended:

5.1 Z-Wing (TCP / IP)

Z-Wing (`luxfi/zwing` [13]) is the canonical Lux post-quantum secure channel. It exposes a `net.Conn` so application protocols (file transfer, ZAP RPC, HTTP/2, anything that wants a stream) run unchanged on top.

Layer	Z-Wing v1
KEM	X-Wing (X25519 + ML-KEM-768) per the IETF draft [7]
Identity	Ed25519 + ML-DSA-65 hybrid signatures (FIPS 204)
AEAD	ChaCha20-Poly1305 with sequence-numbered nonces
Replay protection	64-bit sequence counter; per-direction key derivation
Forward secrecy	Ephemeral X-Wing per session; long-term keys do not encrypt
Negotiation	None. Z-Wing v1 is one suite.
Layering	Routes over plain TCP today; RNS in v2

Table 3: Z-Wing channel surface.

The "no negotiation" choice is deliberate. Cipher negotiation has been the source of every major TLS downgrade attack. Two endpoints either speak Z-Wing v1 or they don't talk. A future Z-Wing v2 (e.g., ML-KEM-1024 + Ed448 + ML-DSA-87) is a new ALPN-style identifier, not a parameter inside v1.

5.2 Reticulum Network Stack (RNS)

For deployments where TCP/IP isn't available – mesh radio, LoRa, disconnected networks, contested EW environments – PQSAFE rides over RNS (LP-9701 [16]). RNS is a cryptographic-mesh transport that uses identity-based addressing and hop-by-hop authenticated encryption, originally designed for amateur radio and now widely adopted by survival-mesh communities.

The Lux RNS profile (see `node/network/dialer/rns_*.go`) keeps RNS's classical identity (Ed25519 + X25519) and adds a hybrid PQ identity layer parallel to Z-Wing's:

Purpose	Classical (RNS native)	Post-quantum (Lux profile)
Identity signing	Ed25519	ML-DSA-65 (FIPS 204)
Key exchange	X25519	ML-KEM-768 (FIPS 203)
Session encryption	AES-256-GCM	–
Key derivation	HKDF-SHA256	–

Table 4: RNS hybrid PQ suite (LP-9701).

Forward secrecy properties:

- Fresh X25519 + ML-KEM-768 keypairs per session.
- Ephemeral private keys zeroed after the handshake completes.
- Combined secret = $\mathsf{HKDF}(\text{X25519 shared} \,\|\, \text{ML-KEM shared})$ – both components must break to retrospectively decrypt.
- AES-256-GCM session keys are rotated on RNS's standard ratchet cadence; nonces are sequence-numbered.

Wire-format growth from classical RNS to the PQ-hybrid profile:

Component	Classical	Hybrid	Δ
Public identity	64 B	∼3.2 KiB	+3.1 KiB
Signature	64 B	∼2.5 KiB	+2.4 KiB
Key exchange ct	64 B	∼1.2 KiB	+1.1 KiB
Handshake total	∼256 B	∼7.5 KiB	+7.2 KiB

Table 5: Wire-format growth (per LP-9701 LP-9701 measurements).

A capability-exchange byte in the announce frame advertises PQ support. A peer with `requirePostQuantum: true` refuses classical-only peers; a peer with `requirePostQuantum: false` falls back gracefully to classical RNS so mixed networks coexist during the migration window.

6 Storage profile

PQSAFE-IL5-HYBRID-v1 treats every back-end as ciphertext-only:

- **DoD SAFE (existing).** Upload the `.pqsafe` envelope as a single file; SAFE never sees plaintext or recipient identity (only opaque manifest hashes). Recipients pull the file through SAFE, then decrypt locally.
- **S3-compatible object storage.** Chunks become S3 objects under a per-envelope prefix. The manifest is a small object alongside. Server-side encryption is irrelevant – the data is already encrypted at the client.
- **SharePoint.** The envelope uploads as a single `.pqsafe` archive; SharePoint metadata stores only the manifest hash (not recipient lists or labels).
- **IPFS / content-addressed.** Each chunk is content-addressed by SHA-384; the manifest carries the CIDs and the recipient list. Loss of access control at the IPFS layer is irrelevant – the data is encrypted.
- **Lux storage chain.** Envelopes are stored as opaque blobs; the chain holds only audit hashes.

6.1 File-at-rest with luxfi/age

When end-to-end transfer is not possible (offline replication, backup, legacy ingest), the envelope is re-keyed at rest with **luxfi/age** [12]. `age` is Filippo Valsorda's modern file-encryption tool [11]; the Lux fork adds an `age-mlkem768` recipient stanza that wraps the file key under the recipient's ML-KEM-768 public key in addition to (or instead of) the classical X25519 recipient stanza. The same hybrid policy as the envelope applies: production deployments accept either the `age-mlkem768` stanza alone (pure PQ) or both X25519 + `age-mlkem768` stanzas (hybrid). The age binary format is unchanged, so existing tooling and audit pipelines work without modification.

Listing 2: luxfi/age usage examples.

```
# Encrypt to a recipient's ML-KEM-768 public key
age -r mlkem1q5...4q3 -o report.pdf.age report.pdf

# Hybrid (X25519 + ML-KEM-768): both stanzas must decrypt
age -r age1lk...3qm -r mlkem1q5...4q3 -o report.pdf.age report.pdf

# Decrypt with an identity file containing both legs
age -d -i ~/.config/age/identity.txt report.pdf.age
```

6.2 Replication: hanzo/replicate

For data that crosses agency boundaries via replication rather than point-to-point transfer, the same envelope flows through `hanzo/replicate`'s compaction pipeline. Chunks are content-addressed by SHA-384; the replication layer sees only hashes and ciphertext. The Lux/Hanzo Virtual File System (`hanzo/vfs`) exposes the envelope as a mount-point at the recipient, decrypting chunks lazily on read with the recipient's X-Wing private key. A revocation in the manifest immediately blocks subsequent reads even if the ciphertext chunks have been replicated.

7 Audit and revocation

7.1 Audit log content (what goes on-chain)

```
{
  "event":                "sealed" | "transferred" | "opened" | "revoked",
  "manifest_hash":        "sha384:...",
  "sender_cert_hash":     "sha384:...",
  "recipient_set_hash":   "sha384:...",       // ordered SHA-384 of recipient IDs
  "policy_label":         "CUI",              // category only, no specific PII
  "timestamp":            "2026-05-12T22:00:00Z",
  "anchor_chain":         "lux-q",            // Q-Chain (Quasar finality)
  "profile_version":      "PQSAFE-IL5-HYBRID-v1"
}
```

What is NOT on-chain: recipient emails, file names, passphrases, claim IDs, PHI/PII, CUI descriptions, manifest contents, DEK, or any cipher- or plaintext.

7.2 Revocation

A revocation event publishes the manifest hash to the policy registry. Verifiers refuse to honour any capsule referencing a revoked manifest. The capsule's kem_ct is content-bound to the manifest, so a malicious actor cannot retroactively forge a capsule referencing a different manifest.

Revocation is irreversible-by-design. To resume sharing after an accidental revocation, a new manifest is created with a new DEK and a fresh recipient set. Chunks are re-encrypted (cheap: $O(|\text{file}|)$ on the sender) and a new envelope is published.

7.3 Expiry

Every manifest carries a `ttl` field. Verifiers SHOULD refuse to open envelopes past their TTL. The audit log records expiry events implicitly through the timestamp.

8 Crypto agility

Every parameter is versioned at the manifest layer:

Field	Wire identifier	Allowed values (v1)
`profile`	ASCII string	"PQSAFE-IL5-HYBRID-v1" only
`kem`	ASCII string	"X-Wing-MLKEM768" or "X-Wing-MLKEM1024"
`bulk_aead`	ASCII string	"AES-256-GCM" or "ChaCha20-Poly1305"
`hash`	ASCII string	"SHA-384" or "SHA-3-384"
`kdf`	ASCII string	"HKDF-SHA-384"
`sig_classical`	ASCII string	"Ed25519" or "Ed448"
`sig_pq`	ASCII string	"ML-DSA-65" or "ML-DSA-87" or "SLH-DSA-SHA2-192f"

Table 6: Versioned wire identifiers.

When a primitive must be deprecated (e.g., FIPS validation status changes, or a new attack reduces the security level), a new profile identifier is minted (PQSAFE-IL5-HYBRID-v2). Verifiers route by profile identifier; the v1 verifier does not attempt to parse v2 envelopes. The two-byte profile-identifier check is the entire forward-compatibility surface, by design.

8.1 The "PQ-only" exit ramp

When DoD policy and PKI catch up to PQ-only operation, PQSAFE migrates through three explicit profile versions:

1. **PQSAFE-IL5-HYBRID-v1** (this profile): X-Wing hybrid; Ed25519+ML-DSA-65 hybrid signatures. Production today.
2. **PQSAFE-IL5-HYBRID-v2**: ML-KEM-1024; ML-DSA-87; SHA-3-384 default. Migration window.
3. **PQSAFE-IL5-PURE-PQ-v1**: ML-KEM-768/1024 (no X25519); ML-DSA-65/87 only (no Ed25519). Activated when CNSA-3.0 / equivalent policy permits classical-curve removal.

The bulk AEAD path (AES-256-GCM, ChaCha20-Poly1305) survives all three versions: it is symmetric, post-quantum-resistant for the foreseeable future (Grover gives only $\sqrt{N}$ speed-up, leaving AES-256 at 128-bit effective security), and FIPS-aligned.

9 DoD operational controls

Area	Control
Classification boundary	No classified data unless the whole system is accredited for that domain. PQSAFE is NOT a cross-domain solution.
CUI / PII / PHI / IL5	Client-side encryption before upload; recipient authorization verified outside the crypto layer (ABAC/RBAC + cert-bound identity).
Crypto module	Prefer FIPS 140-3 validated modules where available; algorithms restricted to NIST/CNSA set. No custom cryptography.
PQ mode	Hybrid classical + PQ in v1. Pure PQ only on the explicit exit-ramp profile (Section 8).
Key custody	HSM/KMS-backed KEKs; per-file/per-chunk DEKs ephemeral; private keys never leave secure elements / smart cards.
Audit	Tamper-evident audit trail, but no secrets and no sensitive metadata on public ledgers.
Metadata minimisation	No filenames, recipients, CUI labels, passphrases, claim IDs, or mission data on-chain.
Crypto agility	Every algorithm, parameter set, KDF, AEAD, signature, manifest schema is versioned.
Revocation	Manifest hash blocked at policy registry; capsule content-bound to manifest.
Interop	CBOR/COSE envelope; deterministic test vectors; CLI; SDK; policy engine.

Table 7: Operational controls required for a DoD-grade deployment.

10 Lux stack integration

PQSAFE-IL5-HYBRID-v1 is implemented by composing existing Lux components. No new crypto primitives are introduced.

Lux component	Role in PQSAFE
`luxfi/crypto`	Primitive provider: ML-KEM, ML-DSA, SLH-DSA, AES-GCM, ChaCha20-Poly1305, HKDF, SHA-384/3. Single source of truth for parameter sets.
`luxfi/zwing`	Z-Wing secure channel. PQSAFE transfers ride on top; receives the `.pqsafe` envelope as opaque bytes.
`luxfi/age`	File-at-rest envelope when E2E transfer is impossible: replication, backup, legacy ingest. ML-KEM-768 recipient stanza.
`luxfi/corona`	Module-LWE threshold signature kernel (Ringtail-derived). Recovery quorum and federation root signing.
`luxfi/pulsar`	Module-LWE threshold signature kernel. Output byte-equal to FIPS 204 ML-DSA, so verifiers consume PQSAFE manifests with a single FIPS-validated ML-DSA library.
`luxfi/kms`	HSM-backed key custody: long-term identity keys, federation root, recovery-share KEKs.
`hanzoai/iam`	Identity provider: certificate issuance, recipient public-key lookup, revocation registry.
`hanzo/vfs`	Mount-point view of decrypted envelopes; lazy chunk decryption with the recipient's X-Wing private key.
`hanzo/replicate`	Replication of ciphertext chunks across storage back-ends; sees only SHA-384 content addresses.
`lux Q-Chain`	Audit anchor: opaque hash commitments only. No sensitive metadata.
`lux M-Chain`	MPC ceremonies for recovery-quorum DKG / proactive resharing (LP-134).

Table 8: Lux components in the PQSAFE architecture.

10.1 Component CLI: `pqsafe`

```
# Seal a file with PQSAFE
pqsafe seal \
  --in report.pdf \
  --out report.pdf.pqsafe \
  --recipient alice.mil \
  --recipient bob.gov \
  --label CUI \
  --ttl 7d \
  --hybrid mlkem768+x25519 \
  --sign mldsa65+ed25519 \
  --audit lux-q

# Open a sealed envelope
pqsafe open --in report.pdf.pqsafe --out report.pdf

# Verify only (no decrypt) - checks manifest signatures + Merkle root
pqsafe verify --in report.pdf.pqsafe

# Revoke
pqsafe revoke --manifest-hash sha384:abcd... --reason "recipient compromised"

# Re-wrap (rotate KEK without re-encrypting chunks)
pqsafe rewrap --in report.pdf.pqsafe --add-recipient charlie.gov

# Replicate via hanzo/replicate
pqsafe push report.pdf.pqsafe replicate://safe.apps.mil/cui/
```

11 Deployment for safe.apps.mil

The proposed federated deployment at `safe.apps.mil` fronts the existing DoD SAFE storage with the PQSAFE envelope layer.

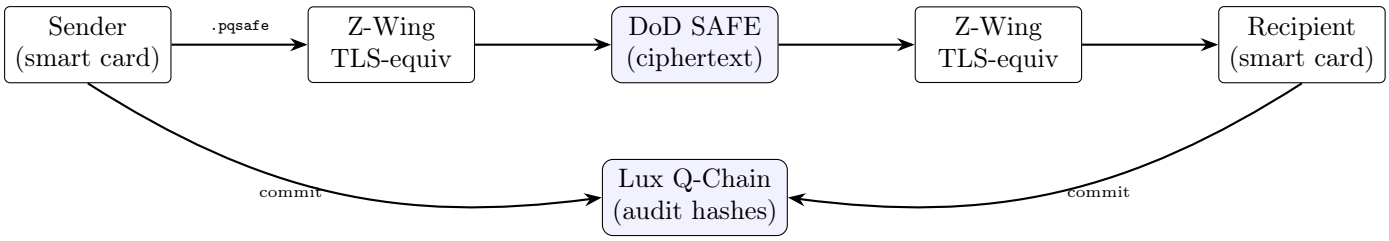

Figure 1: Sender and recipient hold the only plaintext. Z-Wing protects the channel; DoD SAFE / object storage / SharePoint / etc. holds ciphertext; Q-Chain holds audit hashes only.

11.1 Migration from DoD SAFE today

DoD SAFE currently provides TLS-protected file drop-off with sender-set expiry. The migration path:

1. **Phase 1 (PQSAFE alongside SAFE).** Users upload PQSAFE envelopes through SAFE unchanged. SAFE sees only ciphertext; the existing SAFE infrastructure is untouched.
2. **Phase 2 (PQSAFE-native ingest).** A `safe.apps.mil` endpoint accepts PQSAFE envelopes directly, fronted by Z-Wing for endpoint-to-endpoint PQ confidentiality and an HSM-backed certificate chain for ABAC enforcement.
3. **Phase 3 (legacy SAFE deprecation).** SAFE TLS-only mode is retired; `safe.apps.mil` requires either Z-Wing or PQSAFE-on-TLS-1.3-hybrid. Backward-compat preserved through the hybrid TLS cipher suite (X25519MLKEM768, IANA `0x11ec`).

12 What we don't do

- Don't say "blockchain makes files secure." The chain is an audit anchor, nothing more.
- Don't store plaintext, filenames, recipient identities, or passphrases on-chain.
- Don't use PQ-only before the hybrid migration is policy-accepted.
- Don't invent a new KEM, signature, or AEAD.
- Don't treat DoD SAFE as PQ just because it uses TLS.
- Don't treat Lux primitive claims as DoD-approved without FIPS 140-3 validation, code audit, SBOM, threat model, and ATO evidence.
- Don't use Pulsar/Corona threshold consensus signatures as if they were file encryption.
- Don't use smart contracts as a key manager for CUI/PII/PHI.

13 Conclusion

PQSAFE-IL5-HYBRID-v1 is what DoD asks for when DoD says "replace SAFE with something post-quantum and crypto-agile." It is not a new cryptosystem. It is a small layer of envelope discipline applied above NIST-standardised primitives, with the Lux PQ stack supplying the parts (Z-Wing for the channel, age + ML-KEM for at-rest, corona + pulsar for threshold recovery, the Q-Chain for audit anchoring, KMS for HSM-backed key custody). The win is not novelty – it is that *every byte that leaves a DoD endpoint is encrypted with a hybrid classical + post-quantum primitive that NIST has standardised, the sender has signed, the manifest has bound, and the audit log has witnessed, without trusting the storage system, the network, or the auditor with anything beyond a hash.*

References

[1] NIST. *FIPS 203: Module-Lattice-Based Key-Encapsulation Mechanism Standard.* August 2024. https://csrc.nist.gov/pubs/fips/203/final

[2] NIST. *FIPS 204: Module-Lattice-Based Digital Signature Standard.* August 2024. https://csrc.nist.gov/pubs/fips/204/final

[3] NIST. *FIPS 205: Stateless Hash-Based Digital Signature Standard.* August 2024. https://csrc.nist.gov/pubs/fips/205/final

[4] NIST. *FIPS 197: Advanced Encryption Standard (AES).* 2001 / updated 2023.

[5] NIST. *Special Publication 800-208: Recommendation for Stateful Hash-Based Signature Schemes.* October 2020.

[6] NSA. *Commercial National Security Algorithm Suite 2.0 (CNSA 2.0).* September 2022. https://www.nsa.gov/Press-Room/Press-Releases-Statements/Press-Release-View/Article/3148990/

[7] Manuel Barbosa, Deirdre Connolly, João Diogo Duarte, Aaron Kaiser, Peter Schwabe, Karoline Varner, Bas Westerbaan. *X-Wing: The Hybrid KEM You've Been Looking For.* IACR ePrint 2024/039 and IETF `draft-connolly-cfrg-xwing-kem`. https://eprint.iacr.org/2024/039

[8] Y. Nir, A. Langley. *RFC 8439: ChaCha20 and Poly1305 for IETF Protocols.* 2018.

[9] C. Bormann, P. Hoffman. *RFC 8949: Concise Binary Object Representation (CBOR).* 2020.

[10] J. Schaad. *RFC 8152: CBOR Object Signing and Encryption (COSE).* 2017.

[11] Filippo Valsorda. *age: A simple, modern and secure file encryption tool, format, and Go library.* https://age-encryption.org/

[12] Lux Industries Inc. *luxfi/age: hybrid PQ recipient stanza for age.* https://github.com/luxfi/age

[13] Lux Industries Inc. *luxfi/zwing: Z-Wing post-quantum secure channel + canonical X-Wing / hybrid-identity primitives.* https://github.com/luxfi/zwing

[14] Lux Industries Inc. *luxfi/pqsafe: reference implementation of PQSAFE-IL5-HYBRID-v1.* https://github.com/luxfi/pqsafe

[15] Lux Industries Inc. *luxfi/pqrns: PQ-RNS hybrid post-quantum profile for the Reticulum Network Stack.* https://github.com/luxfi/pqrns

[16] Lux. *LP-9701: Reticulum Network Stack (RNS) — hybrid PQ profile.* https://github.com/luxfi/lps/blob/main/LPs/lp-9701-reticulum-network-stack.md

[17] Lux. *LP-019: Threshold MPC.* https://github.com/luxfi/lps/blob/main/LPs/lp-019-threshold-mpc.md

Reproducibility

Source code. `github.com/luxfi/pqsafe` (envelope tool, commit `TBD`); `github.com/luxfi/age` (PQ-extended age fork); `github.com/luxfi/zwing` (identity / signing); `github.com/luxfi/pqrns` (Reticulum PQ profile).
Build. `go build ./...` for pqsafe / zwing / pqrns; `cargo build -release` for the age fork.
Hardware tested. TBD (commodity x86_64 Linux with AES-NI; Apple Silicon).
Standards referenced. FIPS 140-3, FIPS 180-4 (SHA-2), FIPS 197 (AES), FIPS 202 (SHA-3/SHAKE), FIPS 203 (ML-KEM-768), FIPS 204 (ML-DSA-65), FIPS 205 (SLH-DSA-SHA2-192f), NIST SP 800-38D (GCM), NIST SP 800-56C (KDF), NIST SP 800-208 (stateful hash-based), RFC 8032 (Ed25519), RFC 9180 (HPKE), CNSA 2.0.
Contact. security@lux.network.