Ilyazh-Web3E2E: A Post-Quantum Hybrid Protocol for Forward-Secure Decentralized Messaging

Ilyas Zhaisenbayev
Independent Researcher
Email: ilyaszhaisenbaev@gmail.com
Version 0.8 (2025-10-03) This work is licensed under a
Creative Commons Attribution 4.0 International License (CC BY 4.0)

***Abstract*—We specify *Ilyazh-Web3E2E*, a hybrid, post-quantum-resilient end-to-end messaging protocol. The handshake is a mutually authenticated AKE in the Canetti–Krawczyk (CK'01) model that combines classical X25519 with a NIST-selected KEM (ML-KEM-768), together with explicit domain separation and downgrade resistance. Session confidentiality and integrity are provided by AES-256-GCM; forward secrecy (FS) and post-compromise security (PCS) are ensured by a Double Ratchet. We formalize the threat model, goals, wire format, and invariants; and give *self-contained, game-based security arguments with explicit bounds* (hybrid KEM/DH reduction to AEAD channel confidentiality and authenticity). We outline a practical implementation path. Empirically, PQ submissions in the early NIST process saw substantial break rates by 2023 (e.g., 48% of Round-1, 25% of the survivors, and 36% of Round-2 [22]), motivating hybridization over PQ-only deployment during migration. We provide a full specification, security model, and proof sketches to facilitate adoption and independent analysis.**

Executive Summary — Why prefer Ilyazh-Web3E2E (vs. PQXDH/TLS/MLS)

Scope. Decentralized 1:1 messaging that needs immediate post-quantum confidentiality (HNDL resilience), explicit downgrade/UKS protection, and routine PCS healing without dependence on a single directory.

What is new (and when it matters):

- **Full transcript binding with *sid*-in-AAD.** Every negotiation element (suite, caps, identities, ephemerals, KEM cts, roles) goes into the transcript hash; the derived session identifier *sid* appears in AAD of *every* record. This makes downgrade/UKS attempts *auditable and rejectable* in-band.
- **Mandated hybrid re-encapsulation cadence.** Periodic ML-KEM re-encapsulation plus X25519 rotation with published per-epoch caps (messages/time) to sustain HNDL resilience for long-lived/mobile sessions—moving from "allowed" to *required* policy.
- **Dual-signature authentication by default.** Ed25519 + ML-DSA-65 on the handshake transcript increases migration robustness without sacrificing message-layer deniability (signatures are handshake-only).
- **Decentralization-friendly identity.** Verifiable prekey bundles and audit-friendly *sid* avoid pinning to a single operator; suits multi-operator/Web3 deployments.

Trade-offs (explicit): modestly larger handshake (PQ signature), one extra PQ verify, optional rekey RTT; we document caps/overheads and keep AEAD headers minimal. **Why not PQ-only during migration.** Following Bernstein's qrcsp analysis, relying solely on a single PQ primitive risks a single point of catastrophic failure if that primitive is later broken; a simple DH+KEM combiner avoids this failure mode and preserves classical security while PQ assumptions stabilize [22].

When to prefer: decentralized 1:1 with HNDL-first confidentiality and auditability. **When not:** ecosystem interop today (prefer Signal PQXDH), pure client–server (TLS-hybrid), large groups (MLS).

Threat model. Active network attacker with store-now-decrypt-later capability (HNDL), directory or relay compromise, SIM-swap/number recycling, and endpoint compromise between epochs.

Selection criteria. We compare by: immediate PQ confidentiality, downgrade/UKS resistance, channel binding, rekey cadence (PCS), mobility/handover, relay independence, and operational cost.

Mini case-studies. *(i) Downgrade/UKS-relay:* any relay-induced variant flips the transcript hash and thus `sid`; peers reject on AAD mismatch. *(ii) Mobility:* handover preserves `sid` and counters; liveness is probed via stateless retries; no directory pinning.

***Index Terms*—Post-Quantum Cryptography; AKE; IND-CCA; End-to-End Encryption; Double Ratchet; Hybrid KEM; ML-KEM; ML-DSA; X25519; AEAD; HKDF; CK'01; PQXDH; PQ3.**

Why prefer Ilyazh-Web3E2E over existing, well-scrutinized designs (e.g., PQXDH, TLS-hybrid, MLS)?

Scope. We focus on decentralized 1:1 messaging that needs *post-quantum confidentiality today* (harvest-now-decrypt-later, **HNDL**, resilience), explicit downgrade/UKS protection, and routine PCS "healing" in long-lived sessions **without** centralized directories.

What is different (and when it matters):

TABLE I
EXECUTIVE COMPARISON (1:1 MESSAGING). "COST" COUNTS EXTRA BYTES/RTT/SIGNATURES/VERIFICATIONS RELATIVE TO CLASSICAL X25519+AEAD.

Property	How in Web3E2E	Comparable in PQXDH/TLS/MLS	Cost
Immediate PQ conf.	Hybrid ML-KEM+ X25519; HKDF mix	PQXDH: optional; TLS-hybrid: yes; MLS: group	∼1.1 KB (ct), 0 RTT
Downgrade/UKS	Transcript-bound `sid` in AAD; dual-sig	PQXDH: app-defined; TLS: version/sig; MLS: tree hash	+2 sigs verify
PCS cadence	Epoch re-encap with caps	PQXDH: re-X3DH; TLS: resumption/psk; MLS: UPKs	1 rekey RTT (optional)
Mobility/handover	Stable `sid`, relay-agnostic	PQXDH: session break; TLS: connection-bound; MLS: reinit	none
Auditability	AAD carries `sid`, counters	rare	header bytes only

1) **Full-context transcript binding + session-ID-in-AAD.** Every negotiation element (suite, capabilities, identities, ephemerals, KEM ciphertexts, role tags) feeds the transcript hash used as HKDF salt; the derived `sid` is carried in the AAD of *every* record. This yields *auditable, explicit* downgrade/UKS resistance and tamper-evident logs—a property not mandated in PQXDH and orthogonal to TLS record-layer binding.
2) **Mandated hybrid re-encapsulation cadence.** We *require* periodic ML-KEM re-encapsulation plus X25519 DH rekeying with published limits (per-epoch message caps / time-based turnover), sustaining HNDL resilience for mobile, long-lived sessions; PQXDH does not mandate such cadence.
3) **Hybrid handshake robustness by default.** Dual signatures (Ed25519 + ML-DSA-65) authenticate identities in the handshake *by default*, with an explicitly documented weaker "at-least-one" mode. This improves robustness to single-scheme breaks while preserving message-layer deniability.
4) **Decentralization-friendly identity distribution.** The protocol tolerates directories outside a single provider and keeps audit-friendly `sid`s, which can be essential in Web3 or multi-operator settings.

Trade-off: a modest increase in handshake size and compute; we document measured overheads and operational limits.

Security arguments at a glance.

- **AKE-authentication.** Dual-signature verification over the full transcript plus key-confirmation ⇒ peer authentication and downgrade/UKS resistance under EUF-CMA of both signature schemes and collision resistance of the transcript hash.
- **Channel confidentiality/integrity.** Assuming nonce uniqueness and label-separated HKDF, *IND-CPA & INT-CTXT* of AEAD + PRF-ness of HKDF + security of the hybrid KEM/DH imply a hybrid channel with IND-CCA confidentiality.
- **FS/PCS.** From one-way chain-KDF and independence of the post-ratchet root after DH+KEM rekey.

Deployment considerations. For maximal ecosystem interoperability today use PQXDH; for client–server, TLS-hybrid; for large groups, MLS. Use **Ilyazh-Web3E2E** when you need *decentralized* 1:1, auditability of invariants (sid-in-AAD, cadence limits), and explicit HNDL resilience.

I. INTRODUCTION

Secure messaging protocols such as Signal, TLS 1.3, and MLS provide confidentiality, authenticity, and forward secrecy. However, they were not designed to provide post-quantum confidentiality today. This paper introduces **Ilyazh-Web3E2E**, a hybrid protocol combining classical (X25519) and post-quantum (ML-KEM-768) primitives with a Double Ratchet construction. Our goal is to provide forward secrecy, post-compromise security, and resilience against quantum adversaries while remaining efficient enough for decentralized messaging and Web3 contexts. The contributions of this work are:

- A hybrid authenticated key exchange (AKE) with transcript binding and dual signatures.
- Integration of ML-KEM-768 alongside X25519 inside the root-key derivation to ensure PQ resilience.
- A Double Ratchet construction supporting both symmetric-key evolution and periodic DH rekeying.
- Detailed specification of wire format, invariants, and HKDF domain-separation labels.
- A formal security model and games capturing AKE-authentication, IND-CCA channel confidentiality, FS, and PCS.
- Proof sketches with explicit advantage bounds and a discussion of limitations.

a) Empirical risk motivation: Data summarized by Bernstein shows high historic break rates in the NIST PQC process by the end of 2023: 48% of the 69 Round-1 submissions (2017) were broken; of the remaining 48, a further 25% were broken; and 36% of the 28 Round-2 selections (2019) were broken [22]. This motivates hybridizing

X25519 with ML-KEM-768 rather than deploying PQ-only KEX during migration. Decentralized platforms ("Web3") increase the demand for protocols resilient to both classical and quantum adversaries. While contemporary secure-messaging designs (e.g., Signal) achieve strong FS/PCS guarantees, long-term confidentiality of recorded traffic remains threatened by future quantum attacks on classical public-key cryptography [18]. **Design principle.** We pursue *trust through transparency*: compose standardized, publicly scrutinized primitives with clear domain separation and a minimal attack surface. We hybridize classical X25519 with ML-KEM-768 in the AKE, and adopt a Double Ratchet [2] for FS/PCS. **Contributions (revised).**

- **Strict transcript binding.** Every context element (suite identifier, capabilities, identities, ML-KEM-768 ciphertexts, ephemeral public keys, role tags) is hashed into the transcript t; we derive a unique session identifier sid and include it in the AAD of every record (sid-in-AAD).
- **Mandated hybrid re-encapsulation.** We fix explicit operational limits and cadence for ML-KEM-768 re-encapsulation (e.g., per-epoch message caps and time-based turnover) to maintain harvest-now-decrypt-later (HNDL) resilience in long-lived sessions.
- **Default downgrade and UKS resistance.** Suite negotiation and capabilities are bound to t; authentication uses *both* signatures (Ed25519 + ML-DSA-65) by default; the "at-least-one" mode is documented as a weaker, opt-in trade-off.
- **Decentralized identity distribution.** The design accommodates identity distribution beyond centralized directories while keeping audit-friendly sid.
- **Reproducible artifacts.** Public PoC, scripts, deterministic test vectors, and a benchmarking harness are provided in the repository (with build instructions).

https://github.com/ilyazh/ilyazh-web3e2e

Positioning & Evidence (one page)

What problem we target and why a new design may be preferable. Table II summarizes four design choices of *Ilyazh-Web3E2E*, why they help in adversarial or multi-operator Web3 settings, and what the explicit costs are. We cross-reference the record format, the key evolution/epoch limits, authentication, identity layer, and the security/usage limits (Table V).

A. Why not just use PQXDH/TLS/MLS?

PQXDH. Excellent lineage for asynchronous messengers, but lacks standardized transcript channel binding; hybrid cadence and anti-UKS require app conventions. In Web3E2E we (a) mandate hybrid cadence with published caps, (b) bind the transcript into a stable `sid` used as AAD, and (c) provide explicit downgrade detection hooks.

TLS-hybrid. Optimized for client-server; FS/PCS are bound to connections and session tickets with server authority. Peer-to-peer mobility, relay-agnostic routing, and deniable message-layer auth are non-goals of TLS.

MLS. Designed for groups; 1:1 is a degenerate case and adds directory dependencies. Our scope is *today* 1:1 with explicit relay/directory independence and HNDL-first confidentiality. We adopt the above only where they dominate (e.g., MLS for larger groups), and we state the trade-offs honestly.

II. Overheads vs. PQXDH / PQ3

We summarize nominal overheads assuming ML-DSA-65 (Dilithium-like) and ML-KEM-768 parameters; values are approximate and for reader intuition.

III. Related Work

a) What is new vs. PQXDH/PQ3?: We use the following abbreviations: *Trust On First Use (TOFU)* and *harvest-now-decrypt-later (HNDL).*

1) *Auditability by design:* sid-in-AAD for every record makes invariants observable;
2) *Mandated hybrid re-encapsulation:* explicit cadence/limits for periodic DH+ML-KEM rekey to sustain resilience in long-lived sessions;
3) *Dual-signature auth policy by default:* Ed25519 + ML-DSA-65, with a documented downgrade trade-off;
4) *Decentralized identity distribution:* suitable for Web3 contexts while preserving audit-friendly sid.

b) QRCSP: empirical PQ breakage.: Bernstein's qrcsp summary highlights substantial historic breakage within the NIST process and argues for hybrid migration strategies over PQ-only deployments during the transition [22]. Our design choices (parallel DH+KEM combiner; mandated cadence) align with this risk model.

Our work builds on the lineage of secure messaging protocols:

- **Signal protocol** [2] introduced the Double Ratchet, achieving forward secrecy and PCS in practice.
- **TLS 1.3 (hybrid)** [14] provides FS in client–server settings but lacks PCS and decentralized suitability.
- **Messaging Layer Security (MLS)** [9] standardizes group messaging with FS and PCS but does not yet integrate PQ primitives.

Post-quantum KEM integration into TLS has been investigated by hybrid TLS experiments [15]. While PQXDH integrates a PQ KEM with the Signal architecture, our design mandates full transcript-binding with sid-in-AAD and a published re-encapsulation cadence, trading modest overheads for explicit auditability and HNDL resilience in decentralized 1:1. This novelty positions Ilyazh-Web3E2E as a step toward PQ-ready secure messaging in Web3 contexts.

Our AKE follows the hybridization rationale explored in prior work on PQ migration for Internet protocols and secure messaging [16]. We build on X3DH/Signal [3],

Criterion	What we do	Why better (when it matters)	Cost / trade-off
Full transcript binding (sid-in-AAD)	Embed session identifier `sid` and role/epoch counters into AEAD AAD of *every* record; bind handshake transcript hashes.	Downgrade and UKS attacks are *observable* and *auditable*; relaying or forking produces binding violations detectable by either endpoint and by external auditors.	A few bytes of AAD per record; stricter validation and logging.
Mandated cadence of KEM+DH	Require periodic ML-KEM re-encapsulation and X25519 rotation with explicit message/byte/epoch limits.(Table V.)	Tighter HNDL/PCS bounds for long-lived/mobile sessions than designs that *allow* but do not *mandate* rekey; session "health-refresh" under loss and roaming.	Extra ciphertext(s) ∼ML-KEM ct per epoch; a rekey RTT in low-latency mode.
Robust authentication (dual-sig)	Default dual signatures in handshake (Ed25519 + ML-DSA-65), with message-layer deniability preserved.	Stronger compromise and migration resilience during PQ transition without giving up deniability at the record layer.	Larger handshake (≈ Ed25519 64B + ML-DSA ∼3.3KB).
Decentralized identity readiness	Identity binding works with multiple operators or PKI providers; avoids a single trust anchor.	Suits Web3/multi-operator deployments where central trust is unacceptable; improves survivability against operator coercion or failure.	Operational complexity; policy must define authority set and rotation.

TABLE II
POSITIONING SUMMARY: FOUR DESIGN CHOICES WITH THEIR BENEFITS AND EXPLICIT COSTS.

TABLE III
BALLPARK OVERHEADS VS. PQXDH / PQ3

Dimension	This work	PQXDH/PQ3	
Handshake bytes	≈ 64B (Ed25519) + 3.3KB (ML-DSA) + 32B (X25519 pk)	≈ 64B + 32B	∼ +3.3KB
KEM overhead / epoch	ML-KEM ct ∼1088B (768)	optional / policy	+1.1KB when mandated
RTT for rekey	0–1 RTT (configurable)	typically optional	bounded extra RTT when enabled
CPU (verify)	one Ed25519 + one ML-DSA	one Ed25519	+ one PQ verify

[11], [1] and track the NIST PQC process [17], [7]. Group messaging considerations relate to MLS [9].

c) Apple iMessage PQ3 [21]: Apple's PQ3 deployed a PQ-authenticated KEX and post-quantum ratcheting in iMessage (2024), targeting long-term confidentiality against store-now-decrypt-later adversaries. While PQ3 focuses on a single-ecosystem messaging stack, our design emphasizes decentralization-friendly identity distribution and explicit HNDL resilience in Web3-like settings. We re-use similar primitives (IND-CCA KEM, AEAD, HKDF) but differ in key schedule and prekey policy.

d) Delta vs. PQXDH & PQ3: Table IV summarizes salient differences.

TABLE IV
DELTA VS. PQXDH AND PQ3 (INDICATIVE).

Aspect	PQXDH	PQ3	Ilyazh-Web3E2E
Handshake hybrid	DH + KEM (X25519 + ML-KEM-768)	KEM + platform auth	DH + KEM with dual signatures over transcript
Auth binding	TOFU + signature	platform signature	dual signature (Ed25519 + ML-DSA-65) over transcript t
Ratcheting	DR (classical/PQ-hybrid)	PQ ratchet	Double Ratchet (DR) with PQ KEM rekey
Decentralized ids	limited (verifiable prekeys)	no	yes (verifiable prekey bundles)
HNDL resilience	implicit	no	explicit (caps, re-encapsulation cadence, nonce/message limits)

IV. Why Ilyazh-Web3E2E over Existing Protocols?

a) Enforceable properties (decision criteria): Relative to PQXDH, TLS-hybrid, and MLS, our design *mandates*:

1) **Full transcript binding & sid-in-AAD.** All negotiation elements (suite, caps, identities, ephemerals, ML-KEM ciphertexts, role tags) are hashed into transcript t; we derive sid = HKDF-Expand(ss, *ilyazh/v0.8/session_id*, 32) and carry it in the AAD of every record. This yields auditable downgrade/UKS resistance.
2) **Mandated hybrid re-encapsulation cadence.** Periodic ML-KEM re-encapsulation plus X25519 rekey, with published caps on per-epoch messages and time-based turnover, sustaining FS/PCS and HNDL resilience.
3) **Hybrid authentication by default.** *Both* Ed25519 and ML-DSA-65 signatures over T; an "at-least-one" policy is documented as weaker and not default.

b) Operational rules (normative):

- **Nonce discipline:** random or HKDF-derived nonces with label separation; no reuse across chains.
- **Turnover limits:** $\leq 2^{20}$ records or 24h per sending chain; session lifetime $\leq 2^{32}$ records or 7 days; exceeding any cap forces rekey or handshake renewal.
- **Directory-agnostic identities:** verifiable prekey bundles; no dependency on centralized directories.

c) Checklist.: [Threat Model Checklist] **Adversary.** Active network attacker, HNDL recorder, state compromise at time t.
Goals. AKE-auth (CK'01), channel IND–CCA, FS, PCS, downgrade/UKS resistance.
Assumptions. IND–CCA KEM, EUF-CMA signatures, IND–CCA AEAD with unique nonces, HKDF-as-PRF with labels, key erasure.

> **Warning.** [When not to use] Prefer PQXDH for maximum ecosystem interoperability today; TLS-hybrid for client–server; MLS for large groups. Our protocol is tuned for decentralized 1:1 with auditability and mandated PQ resilience.

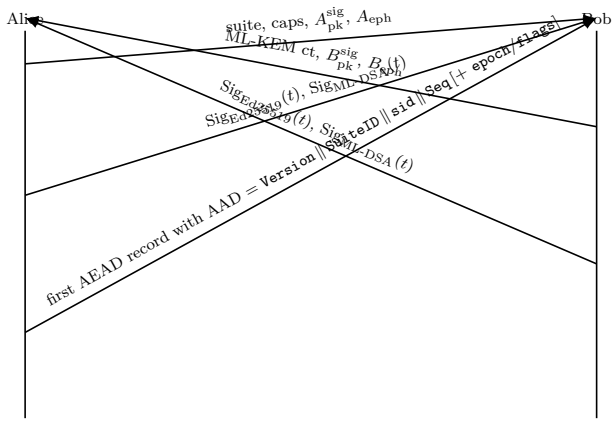
V. Operational Limits and Invariants

VI. Handshake Flow Diagram

Figure 1 summarizes the single-branch 1:1 handshake with full transcript binding and dual signatures.

We collect the normative limits and invariants in one place (all limits are conservative defaults and may be tightened by deployments).

a) Scope and threat model: We target decentralized 1:1 messaging with immediate post-quantum confidentiality against harvest-now-decrypt-later adversaries; explicit anti-downgrade guarantees; and routine PCS "healing" via DH/KEM rekeying without reliance on centralized delivery services.



t = H(suite ‖ caps ‖ ids ‖ ephemerals ‖ KEM cipertexts ‖ roles);
sid = HKDF-Expand(ss, *ilyazh/v0.8/session_id*, 32); used in AAD of *every* record.

Fig. 1. Single-branch handshake with transcript binding, dual signatures, and sid-in-AAD.

TABLE V
Operational limits and invariants (concise).

Property	Normative Rule (Default)
Nonce discipline	HKDF-derived nonces with disjoint per-direction / per-purpose labels; no reuse across chains.
Turnover caps	$\leq 2^{20}$ records or 24 h per sending chain; session lifetime $\leq 2^{32}$ records or 7 days; crossing a cap forces rekey/renew.
Hybrid rekey cadence	Periodic X25519+ML-KEM rekey every 24 h or $\leq 2^{20}$ records, whichever comes first.
Transcript binding	t = H(suite ‖ caps ‖ ids ‖ ephemerals ‖ KEM ct ‖ roles); sid = HKDF-Expand(ss, *ilyazh/v0.8/session_id*, 32); sid must appear in AAD of *every* record.
Auth policy	Dual signatures (Ed25519 *and* ML-DSA-65) by default; "at-least-one" permitted only with a downgrade warning.
Key erasure	Message keys erased after use; outdated chain/root keys erased after turnover.

b) Criteria: We compare across five dimensions: (i) cryptographic properties (FS, PCS, hybrid-PQ, KCI/UKS, anti-downgrade, deniability), (ii) protocol mechanics (RTT, transcript binding, role/keys), (iii) performance (handshake bytes/latency; AEAD throughput), (iv) operational risk (limits, skipped messages, DoS), (v) migration/interop.

c) Against Signal PQXDH: Signal deployed a hybrid X3DH variant (PQXDH) and integrated it with Double Ratchet [11], [12]. Our design trades a modest increase in handshake bytes for *stricter transcript binding* (labeled HKDF across all context items) and *auditable invariants* (explicit re-encapsulation cadence, nonce and message limits). This is beneficial when deployments require decentralized identity distribution and verifiable security

policies. PQXDH remains simpler and is preferable when tight minimalism and existing ecosystem integration are primary goals.

d) Against TLS 1.3 (hybrid): Hybrid KEX for TLS 1.3 is progressing in the IETF [14]; it targets client–server and does not offer PCS. We provide peer-to-peer FS+PCS with a hybrid AKE at session start. If your application is web or enterprise client–server, TLS-hybrid is the natural fit.

e) Against MLS: MLS focuses on scalable groups (RFC 9420), with PQ suites emerging. Our protocol is focused on 1-to-1 today and can later bridge to MLS-style groups.

f) See Limitations:

> **Why prefer Ilyazh-Web3E2E in some deployments?** Relative to PQXDH, we bind *every* context element (suite, capabilities, identities, KEM ct, ephemerals) into the transcript hash used as HKDF salt and carry the derived sid into AAD of *all* records, yielding explicit downgrade/UKS resistance and audit-friendly session IDs. We also *mandate* periodic hybrid re-encapsulation (X25519+ML-KEM-768) with published limits (per-epoch message caps and time-based turnover) so that long-lived sessions maintain harvest-now-decrypt-later resilience on mobile delivery. These choices trade a modest handshake-size increase for stronger transcript binding and operational invariants. If you need maximal ecosystem compatibility today, prefer Signal PQXDH; for client–server only, prefer hybrid TLS; for large groups, MLS. Our design targets decentralized 1-to-1 messaging that requires PQ confidentiality *now* with auditable policies.

Limitations and When Not To Use

- **Production interoperability today:** Prefer Signal PQXDH if you need widest client compatibility and mature deployments.
- **Client–server only:** Prefer hybrid TLS 1.3.
- **Large groups:** Prefer MLS (RFC 9420).
- **Trade-offs:** Our default dual-signature handshake increases size; hybrid KEM re-encapsulation adds bandwidth/compute. We accept these costs to obtain explicit auditability and HNDL resilience in decentralized 1:1 settings.

If you need production-grade interoperability today, use Signal PQXDH; for client–server, use TLS-hybrid; for large groups, MLS. Practitioner note: consider using Ilyazh-Web3E2E when you require decentralized identities, explicit auditability of invariants, and tunable KEM re-encapsulation for long-term PQ confidentiality.

VII. Notation and Setup

a) Roles and identities: Two parties A and B hold long-term signing keypairs (ik_A, IK_A) and (ik_B, IK_B). We write ID_A, ID_B for their identity strings (e.g., stable public identifiers).

b) Ephemeral keys: Each session (and each DH-ratchet step) uses fresh ephemeral keypairs: classical (csk, cpk) for X25519 and post-quantum $(pqsk, pqpk)$ for ML-KEM-768.

c) Primitives: HKDF-Extract/Expand; AEAD AES-256-GCM; signatures (Ed25519 + ML-DSA-65) by default; asymmetric primitives: **DH (classical)** X25519 and **KEM (PQ)** ML-KEM-768. We denote concatenation by $\|$, hashing by H, and HKDF calls by HKDF-Extract(salt,) and HKDF-Expand(, info, L).

d) Domain separation labels: We use ASCII labels in HKDF info: ilyazh/v0.8/session_id, ilyazh/v0.8/root_key, ilyazh/v0.0.8/chain_keys, mk, ck; the transcript hash t is defined below.

e) Nonces and counters: GCM nonces are 96-bit: $N = R_{64} \| C_{32}$ (big-endian / network byte order), where R is a per-ratchet random prefix and C is a per-message counter. All header fields are encoded in network byte order and included as AAD.

VIII. Threat Model and Security Goals

A. Authentication Policy and Deniability

Handshake. Long-term identities are authenticated during the AKE using *hybrid* signatures: Ed25519 and ML-DSA-65 in parallel. Verifiers **MUST** verify both signatures by default; deployments that prioritize deniability over long-term PQ authentication **MAY** disable ML-DSA, but **SHOULD** announce that policy via capabilities bound into sid. *Why dual signatures* The default Ed25519+ML-DSA-65 policy is a robustness combiner for identity authentication: if either scheme remains EUF-CMA secure, handshake authenticity holds. This improves migration resilience without affecting message-layer deniability (signatures remain handshake-only). **Message layer** Application messages are protected by AEAD and *not* individually signed. This preserves deniability in the sense of the Signal literature: any party can simulate ciphertexts given shared keys, and transcripts provide no transferable proof of authorship. **Consequences** Hybrid handshake signatures improve robustness against future breaks of either classical or PQ schemes; message-layer deniability is preserved because signatures are confined to the handshake transcript only. See Downgrade Replay Protection and KCI, Unknown Key-Share (UKS).

B. Network Adversary

We assume a powerful *active* adversary: full control of the network (eavesdrop, inject, replay, reorder, drop), plus key-registration mischief (attempted impersonation). Our model excludes local side channels and compromised CSPRNGs; however, §XIX mandates concrete mitigations.

C. Security Goals

- **AKE in CK'01.** Mutual authentication; freshness; resistance to KCI and replay; explicit key confirmation.

TABLE VI
DEFAULT CRYPTOGRAPHIC SUITE WITH DOMAIN SEPARATION.

Component	Specification	Rationale
DH (classical)	X25519	Efficient, ubiquitous DH; widely deployed.
KEM (PQ)	ML-KEM-768	NIST Level 3; composable DH+KEM; IND-CCA.
AEAD	AES-256-GCM	Standard, hardware-accelerated; good throughput.
KDF	HKDF with SHA-384	Robust extraction/PRF; comfortable security margin.
Signatures	Ed25519 + ML-DSA-65 (default)	Past / future secure mix; dual-sig for downgrade/UKS resistance.
Labels	ilyazh/v0.8/session_id, ilyazh/v0.8/root_key, ilyazh/v0.8/chain_keys, mk, ck	Domain separation for all HKDF calls.

- **Confidentiality/Integrity (AEAD).** IND-CCA security for ciphertexts under AES-256-GCM.
- **Forward Secrecy (FS).** Past messages remain safe after long-term key compromise.
- **Post-Compromise Security (PCS).** Healing after ephemeral/session-state compromise.
- **Post-Quantum Security (PQS).** Confidentiality against quantum adversaries via hybrid KEM.
- **Downgrade Resistance.** Cryptographic suite negotiation is transcript-bound and authenticated.

IX. Cryptographic Specification

A. Primitives and Domain Separation

We fix a default suite with explicit labels (`info`) for HKDF to avoid cross-protocol/key reuse.

B. Handshake (Hybrid AKE)

Let C be the negotiated cipher suite. Let ctx_0 include protocol version, C, parties' identities, and advertised capabilities. Define the transcript hash:

$$\mathsf{t}_0 = H(\mathsf{ctx}_0), \quad H = \text{SHA-384}.$$

Each party holds: classical DH key (sk_A, pk_A) or (sk_B, pk_B) over X25519; PQ KEM keys $(sk_A^{pq}, \cdot)$ and $(\cdot, pk_B^{pq})$. The responder B provides a ML-KEM encapsulation ct_B.

a) Hybrid secret and session derivation:

$$\mathsf{dh} = \text{X25519}(sk_A, pk_B) \tag{1}$$

$$\mathsf{kem} = \text{Decaps}_{ML\text{-}KEM}(sk_A^{pq}, ct_B) \tag{2}$$

$$\mathsf{t}_1 = H(\mathsf{t}_0 \,\|\, pk_A \,\|\, pk_B \,\|\, pk_B^{pq} \,\|\, ct_B) \tag{3}$$

$$\mathsf{ss} = \text{HKDF-Extract}(\mathsf{salt} = \mathsf{t}_1,\ \mathsf{dh} \,\|\, \mathsf{kem}) \tag{4}$$

$$\mathsf{sid} = \text{HKDF-Expand}(\mathsf{ss},\ ilyazh/v0.8/session_id,\ 32) \tag{5}$$

$$\mathsf{rk}_0 = \text{HKDF-Expand}(\mathsf{ss},\ ilyazh/v0.8/root_key,\ 32) \tag{6}$$

$$\mathsf{ck}_0^S \,\|\, \mathsf{ck}_0^R = \text{HKDF-Expand}(\mathsf{rk}_0,\ ilyazh/v0.8/chain_keys,\ 64) \tag{7}$$

b) Authentication and confirmation: By default, both classical and PQ identities authenticate the handshake: each party signs the first-handshake transcript t_1 (which binds version, suite/capabilities, identities and ephemeral keys) with *both* Ed25519 and ML-DSA-65, and verifies both signatures. This dual-signature policy provides explicit downgrade and UKS resistance. After successful verification, each side sends a short AEAD-protected key-confirmation message whose AAD carries sid (and version/suite/seq), thereby binding the confirmation to the session state. An optional "at-least-one" mode is supported as a weaker, opt-in fallback and is *not* the default.

C. Double Ratchet Messaging

We follow the standard DH+KDF chain structure [1]. Let rk be the root key, and $(\mathsf{ck}^S, \mathsf{ck}^R)$ the sending/receiving chains. Each message derives a fresh message key and a unique nonce. When a DH ratchet step occurs, the parties

Algorithm 1 Symmetric-Key Ratchet (Sender)

Require: chain key ck^S, associated data A
1: $\mathsf{prk} \leftarrow \text{HMAC}_{\text{SHA}-384}(\mathsf{ck}^S, A)$
2: $K \,\|\, \mathsf{ck}^{S'} \leftarrow \text{HKDF-Expand}(\mathsf{prk}, \texttt{ilyazh/v0.8/msg}, 64)$
3: **return** $(K, \mathsf{ck}^{S'})$

compute a fresh DH output and update rk, then re-derive $(\mathsf{ck}^S, \mathsf{ck}^R)$ with distinct labels:

$$\mathsf{rk}',\ (\mathsf{ck}^{S'}, \mathsf{ck}^{R'}) = \text{KDF_Root}(\mathsf{rk}, \text{DH}(_{\text{new,peer}})).$$

All unencrypted header fields are authenticated as AAD:

$$\text{AAD} = \texttt{Version} \,\|\, \texttt{SuiteID} \,\|\, \mathsf{sid} \,\|\, \texttt{Seq}\ [+\ \texttt{epoch/flags}].$$

To align authentication with the post-quantum goal, we define a hybrid signature mode. Let $(ik_A^{\mathsf{ed}}, IK_A^{\mathsf{ed}})$ be Ed25519 keys and $(ik_A^{\mathsf{pq}}, IK_A^{\mathsf{pq}})$ be ML-DSA-65 keys for party A (similarly for B). We sign the transcript using both schemes:

$$\sigma_A^{\mathsf{ed}} = \text{Sign}_{ik_A^{\mathsf{ed}}}(t_1), \quad \sigma_A^{\mathsf{pq}} = \text{Sign}_{ik_A^{\mathsf{pq}}}(t_1),$$
$$\sigma_B^{\mathsf{ed}} = \text{Sign}_{ik_B^{\mathsf{ed}}}(t_2), \quad \sigma_B^{\mathsf{pq}} = \text{Sign}_{ik_B^{\mathsf{pq}}}(t_2).$$

TABLE VII
WIRE FORMAT STRUCTURE

Field	Size (bytes)	Description
Version	1	Protocol version (0x03)
Suite ID	2	Negotiated crypto suite
Session ID (sid)	32	HKDF-derived session binding (Eq. 5)
Sequence Num	8	Monotonic counter
Nonce	12	GCM nonce (64-bit rand + 32-bit ctr)
Header Len	2	Length of encrypted header
Encrypted Header	*var*	CBOR ratchet headers
Ciphertext	*var*	AEAD ciphertext + 16B tag

The handshake messages become:

$$A \to B: \ xpk_A, pqpk_A, \sigma_A^{\mathsf{ed}}, \sigma_A^{\mathsf{pq}},$$
$$B \to A: \ xpk_B, pqpk_B, ct_B, \sigma_B^{\mathsf{ed}}, \sigma_B^{\mathsf{pq}}.$$

Verification requires *both* signatures to be valid (strongest composition); as an engineering option, a policy allowing either one ("at-least-one") can be considered but weakens assurances. Public-key distribution is analogous: each identity publishes both IK^{ed} and IK^{pq}. This hybrid mode preserves security if either signature scheme remains unbroken (robustness by composition). Let $\mathsf{ctx} = \mathrm{H}(\mathsf{proto_name} \,\|\, \mathsf{version} \,\|\, \mathsf{suite_id} \,\|\, \mathsf{alg_list})$. Define the running transcript

$$\begin{aligned} t_1 &= \mathrm{H}(\mathsf{ctx} \,\|\, xpk_A \,\|\, pqpk_A \,\|\, ID_A \,\|\, \text{“role=A”}), \\ \sigma_A &= \mathrm{Sign}_{ik_A}(t_1), \\ (ct_B, ss^{pq}) &= ML\text{-}KEM.\mathrm{Encaps}(pqpk_A), \\ t_2 &= \mathrm{H}(t_1 \,\|\, xpk_B \,\|\, pqpk_B \,\|\, ct_B \,\|\, ID_B \,\|\, \text{“role=B”}), \\ \sigma_B &= \mathrm{Sign}_{ik_B}(t_2). \end{aligned}$$

a) Message flow:

$$A \to B: \ xpk_A, pqpk_A, \sigma_A,$$
$$B \to A: \ xpk_B, pqpk_B, ct_B, \sigma_B.$$

b) Shared secrets and root:

$$\begin{aligned} & s_s^x = \mathrm{X25519}(xsk_A, xpk_B) = \mathrm{X25519}(xsk_B, xpk_A) \\ & IKM_0 = s_s^x \,\|\, s_s^{pq} \\ & RK_0 = \mathrm{HKDF\text{-}Extract}(\mathrm{H}(t_2),\, IKM_0) \\ & (CK_0^{\mathsf{snd}},\, CK_0^{\mathsf{rcv}}) = \mathrm{HKDF\text{-}Expand}(RK_0,\ \|\, t_2,\, 2L) \end{aligned} \tag{8}$$

D. Symmetric-key ratchet (per message)

For a given direction $D \in \{\mathrm{snd}, \mathrm{rcv}\}$ and message index i:

$$\begin{aligned} MK_{D,i} &= \mathrm{HKDF\text{-}Expand}(CK_{D,i},\ \text{“}\mathsf{MK}\text{”} \,\|\, t_2,\ \ell_K), \\ CK_{D,i+1} &= \mathrm{HKDF\text{-}Expand}(CK_{D,i},\ \text{“}\mathsf{CK}\text{”} \,\|\, t_2,\ \ell_{CK}), \\ \mathsf{nonce}_i &= R_{64} \,\|\, C_{32}, \quad C \leftarrow C + 1, \\ AAD_i &= \mathsf{version} \,\|\, \mathsf{suite_id} \,\|\, \mathsf{seq}\ [+\ \mathsf{epoch/flags}], \\ C_i &= \mathrm{GCMEnc}(MK_{D,i},\ \mathsf{nonce}_i,\ AAD_i,\ M_i). \end{aligned}$$

TABLE VIII
WIRE FORMAT STRUCTURE

Field	Size (bytes)	Description
version	1	Protocol version (e.g., 0x03).
suite_id	2	Crypto suite identifier.
seq	8	Monotonic message counter within ratchet epoch.
nonce	12	$R_{64} \,\|\, C_{32}$ (per-epoch prefix + per-message counter).
ratchet_epoch	4	Epoch identifier (increments at each DH-ratchet).
flags	2	Bitfield (e.g., KEM present, rekey signal).
enc_kem	var	(Optional) encapsulated key(s) for DH-ratchet step.
enc_hdr	var	Encrypted ratchet header (CBOR), if used.
ciphertext	var	AEAD ciphertext ‖ 16-byte GCM tag.

E. DH-ratchet (periodic healing)

When a DH-ratchet step occurs, parties derive fresh shared material:

$$\begin{aligned} & s_{s\,\mathrm{new}}^x = \mathrm{X25519}(xsk'_A, xpk'_B) = \mathrm{X25519}(xsk'_B, xpk'_A), \\ & s_{s\,\mathrm{new}}^{pq} \leftarrow \text{ML-KEM re-encapsulation}, \\ & IKM_{\mathrm{upd}} = s_{s\,\mathrm{new}}^x \,\|\, s_{s\,\mathrm{new}}^{pq}, \\ & RK' = \mathrm{HKDF\text{-}Extract}(RK,\, IKM_{\mathrm{upd}}), \\ & (CK'^{\mathrm{snd}},\, CK'^{\mathrm{rcv}}) = \mathrm{HKDF\text{-}Expand}\big(RK',\, \mathsf{rekey\text{-}chains} \,\|\, t_2,\, 2L\big). \end{aligned} \tag{9}$$

Reset R and the message counter C for the new ratchet epoch.

X. Formal Wire Format and Invariants

A. Wire Format

AAD = Version ‖ SuiteID ‖ sid ‖ Seq [+ optional epoch/flags].Headers use CBOR [10] and may include ratchet counters and skipped-key ranges.

B. Protocol Invariants

- **Nonce structure:** R_{64} fixed per (sub)chain; $\mathrm{ctr}_{32} \uparrow$ per message; reset only when R_{64} rotates.
- **Rekey:** after 2^{20} msgs or 24 h; session turnover after 2^{32} msgs or 7 days.
- **Key erasure:** zeroize used message keys and preimage material immediately after use.

All fixed-length fields are big-endian. All header fields are authenticated as AAD.

AAD = Version ‖ SuiteID ‖ sid ‖ Seq [+ optional epoch/flags].

- **Nonce uniqueness**: For a fixed AEAD key, nonces must never repeat. Construct $N = R_{64} \,\|\, C_{32}$ (big-endian), where a fresh 64-bit R is sampled per sending chain/ratchet epoch and a 32-bit counter C increments per record.

- **Rekey policies**: Symmetric rekey at least every 2^{20} records or 24 h per sending chain. Enforce session re-establishment after $\leq 2^{32}$ total records or 7 days.
- **Skipped records**: Keep a bounded LRU cache of derived but unconsumed MK (sliding window W), with hard memory limits and safe eviction.
- **Constant-time & zeroization**: All secret-bearing operations are constant-time; securely zeroize sensitive buffers after use and upon key turnover.
- **RNG policy**: Hybrid entropy (HW RNG + OS CSPRNG), periodic reseeding, and snapshot resilience (post-rollback rekey).
- info = "init-chains" $\| t_2$ for initial chain keys.
- info = "MK" $\| t_2$ for per-message keys.
- info = "CK" $\| t_2$ for chain evolution.
- info = "rekey-chains" $\| t_2$ for DH-ratchet rekey.
- salts: $\mathrm{H}(t_2)$ for the initial extract; later extracts use current RK as salt (KDF chaining).

XI. Conceptual Test Vectors

A. Handshake (AKE)

- Alice Ed25519 (id): `1f2c3d4e...` (32 B)
- Bob ML-KEM ct: `8956a7b8...` (1088 B)
- sid: `d9e1...` (32 B), rk_0: `a1b2...` (32 B)

B. Message Encryption

- AAD (Version ‖ SuiteID ‖ `sid` ‖ Seq): `03 0001 [32B sid] 0000000000000001`
- Nonce (96 b = 8 B IV ‖ 4 B Seq): `4e3291d850a43b00 00000001`
- Plaintext: `48656c6c6f2057656233` ("Hello Web3")
- Ciphertext: `89ab12cd...` (len(PT) + 16B)

Note. This is a shape-only, non-binding example: hex values are illustrative. Use the canonical labels `ilyazh/v0.8/session_id`, `.../root_key`, `.../chain_keys` in HKDF and the full AAD `Version` ‖ `SuiteID` ‖ `sid` ‖ `Seq` [+ `epoch/flags`].

XII. Formal Security Model and Games

XIII. Why prefer this design (and when not to)

We prefer *Ilyazh-Web3E2E* in adversarial multi-operator settings, for long-lived/mobile sessions, and for deployments that require explicit auditability of transcript-binding and anti-downgrade properties (cf. Table II).

a) When not to use this protocol: If you have a single trusted operator with standard PKI, short-lived sessions, and no requirement for cross-operator auditability, then Signal, TLS 1.3, or MLS may be preferable. We make this explicit to help engineers choose the simplest sufficient design for their threat model.

XIV. Concrete attack scenarios and why the invariants defeat them

a) (A) Transcript downgrade / UKS via adversarial relay: Setup: Adversary M relays between A and B, attempting to negotiate weaker parameters or fork the view (UKS). *Invariant use:* The handshake transcript (ciphersuite, identity bindings, and ephemeral keys) is hashed into `sid`; `sid` and epoch/role counters are placed into AEAD AAD for every record. *Argument:* Such attacks are rejected due to sid-in-AAD binding, as established in §V, which yields `sid` divergence; subsequent records verify AAD against a different `sid` and fail authentication, producing an auditable error. In contrast, designs that do not bind a stable session identifier into every record's AAD leave room for non-fatal divergence until higher layers detect it.

b) (B) HNDL / long-lived session erosion under loss and roaming: Setup: Mobile clients accumulate loss and reordering; adversary opportunistically replays stale ciphertexts or induces stalls. *Invariant use:* Explicit *mandated* cadence of ML-KEM re-encapsulation and DH rotation with message/byte/epoch limits, plus per-epoch key separation. *Argument:* Rekey cadence bounds the adversary's advantage window: replay across epochs is rejected; within-epoch usage limits and fresh KEM ct enforce forward- and post-compromise-security style bounds. Designs that merely *allow* (but do not *mandate*) cadence rely on application discipline and may drift into weak long-lived operation.

XV. Security Theorems and Proofs

a) Parallel combiner rationale: The hybrid secret uses a parallel combiner: if either X25519 or ML-KEM remains secure, the derived session key is pseudorandom. This aligns with standard KEM–DEM and combiner arguments (cf. Shoup; Hofheinz–Kiltz–Jager).

Theorem 1 (Channel Confidentiality: IND–CCA via KEM–DEM): Assume the handshake KEM is IND–CCA and the record AEAD is IND–CCA with unique nonces. If root, chain, and message keys are derived via HKDF with label separation using transcript salt `t`, then any PPT adversary has at most negligible advantage in a chosen-ciphertext attack against channel confidentiality.
[Game-hopping proof sketch] (1) Replace KEM shared secret by uniform (KEM IND–CCA). (2) Replace HKDF outputs with PRF-random (label separation). (3) Answer decryption queries via the AEAD IND–CCA challenger; nonce uniqueness holds by policy. Ciphertexts become independent of plaintexts; the advantage is negligible.

Game-Hop Proof Sketch for Theorem 1

We prove IND–CCA confidentiality of the channel using a standard sequence-of-games argument.

Game 0 (Real World): This is the real IND–CCA game, where the adversary interacts with the actual encryption and decryption oracles.

Game 1 (KEM replacement): Replace the hybrid KEM output with a uniformly random string. The adversary's distinguishing advantage changes by at most $\mathsf{ML\text{-}KEM}_{\mathsf{IND\text{-}CCA}}$.

Game 2 (HKDF replacement): Replace all HKDF outputs with ideal PRF outputs (domain-separated by

context labels). The distinguishing gap changes by at most $\mathsf{HKDF}_{\mathsf{PRF}}$.

Game 3 (AEAD replacement): Replace the AEAD encryption with a random authenticated encryption oracle. Any adversary distinguishing this game breaks either IND–CPA or INT–CTXT security of AEAD.

Game 4 (Challenge ciphertext): The adversary receives one challenge ciphertext. All queries use unique nonces, guaranteed by construction (see §IX-B). Therefore, challenge ciphertext reveals no information.

Conclusion:

$$\mathsf{Adv}^{\mathsf{channel}}_{\mathsf{IND\text{-}CCA}} \leq \mathsf{Adv}^{\mathsf{KEM}}_{\mathsf{IND\text{-}CCA}} + \mathsf{Adv}^{\mathsf{HKDF}}_{\mathsf{PRF}} + \mathsf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{AEAD}} + \mathsf{Adv}^{\mathsf{INT\text{-}CTXT}}_{\mathsf{AEAD}} + (\lambda)$$

Theorem 2 (AKE Authenticity (CK'01-style)): If endpoints verify Ed25519 and ML-DSA-65 signatures over t and run key-confirmation on the first AEAD records (AAD carries sid), then a successful handshake implies matching sessions and shared keys, except with negligible probability in the signature forgeries.

[Reduction sketch] Causing acceptance without a matching partner implies either a forgery of one of the signatures over t or a violation of key-confirmation (which reduces to Theorem 1 via AEAD authenticity). Transcript binding prevents UKS/downgrade since all negotiation elements enter t and sid.

Theorem 3 (FS and PCS for the Double Ratchet): Let chain keys evolve via one-way HKDF steps and refresh the root with periodic X25519+ML-KEM rekey. If state erasure occurs after key use, then (i) past messages remain confidential after compromise (FS) and (ii) after the next successful rekey epoch the adversary's advantage for future traffic is negligible (PCS).

[Argument] FS: compromising CK_i reveals nothing about $CK_{j<i}$; message keys use disjoint labels. PCS: the next DH+KEM refresh replaces the root with independent inputs; by Theorem 1 new AEAD keys are pseudorandom once old keys are erased.

XVI. Security Rationale

We now provide arguments for the security claims of the protocol.

A. IND-CCA Channel Security

Theorem 4 (Channel IND-CCA of Ilyazh-Web3E2E): Assume: (i) the AEAD is IND-CCA secure; (ii) HKDF is a PRF; (iii) X25519 satisfies CDH; (iv) ML-KEM-768 is IND-CCA2; (v) signatures (Ed25519, ML-DSA-65) are EUF-CMA. Then the encrypted channel produced by Ilyazh-Web3E2E is IND-CCA secure under chosen-ciphertext attacks, with adversarial advantage bounded by a negligible function in the security parameters plus the advantages against the underlying primitives.

[Proof sketch]We build a standard sequence-of-games reduction. G_0 is the real IND-CCA game. (1) Replace AEAD with a random authenticated encryption oracle; IND-CCA security of AEAD bounds the distinguishing gap. (2) Replace HKDF with a PRF; the PRF advantage bounds the gap. (3) Embed either the X25519 CDH or ML-KEM challenge into the shared secret; hybrid extraction ensures that breaking the derived keys implies breaking at least one KEX path. (4) Signatures: use EUF-CMA to argue authenticity of handshake transcripts (both signatures in default mode); any successful forgery reduces to breaking one of the schemes. (5) Nonce uniqueness and domain separation (AAD carries sid and version/suite/seq) guarantee consistent decryption oracles except at the challenge. Collecting the bounds across hops yields the stated advantage bound. Assume a PPT adversary $\mathcal{A}$ wins the IND-CCA game against the channel. A reduction $\mathcal{B}$ either (i) embeds a challenge into the AEAD layer (breaking AEAD IND-CCA) or (ii) replaces the hybrid secret with a random string and uses $\mathcal{A}$ to distinguish (breaking either X25519 or ML-KEM IND security). Since both primitives are assumed secure, $\mathcal{A}$'s advantage is negligible.

B. AKE Security (CK'01)

Authentication and freshness derive from transcript-bound signatures and key confirmation. Downgrade resistance follows because suite identifiers are in t_1 and sid, and thus in AAD for all records. Resistance to KCI is inherited from the hybrid construction: forging requires either DH or KEM break.

a) IND-CCA reduction: Assume a PPT adversary $\mathcal{A}$ wins Channel IND-CCA with non-negligible advantage. Build a simulator $\mathcal{B}$ that embeds either an AEAD IND-CCA challenge or a KEM IND-CPA challenge into the target session keys (via HKDF Extract/Expand). Nonce uniqueness plus domain separation allow $\mathcal{B}$ to answer all but the challenge queries consistently. If $\mathcal{A}$ wins, $\mathcal{B}$ breaks the underlying primitive with related advantage.

b) AKE-auth: Use transcript binding (t_1, t_2), role tags, suite identifiers inside the signed transcript, and verify UKS/downgrade resistance by showing that any accepting session has a unique matching partner with the same transcript hash (except with negligible probability).

c) FS/PCS: Show that knowledge of any future state CK_{i+1} does not enable computation of MK_i (preimage resistance of HKDF) and that after DH-ratchet the new root RK' is independent of previously revealed material.

d) Proof sketch (Channel IND-CCA): Let λ be the security parameter. Assume the nonce policy of §IX-B, label-separated HKDF, and a successful, transcript-authenticated handshake. We prove channel IND-CCA by a standard game hop. G_0: real experiment. G_1: replace AEAD with an ideal authenticated encryption; any distinguisher breaks either ind-cpa or int-ctxt. G_2: replace HKDF outputs with PRF-random keyed by the transcript hash $H(t)$ (domain-separation labels) $\Rightarrow$ loss $\mathsf{Adv}^{\mathrm{prf}}_{\mathrm{HKDF}}$. G_3: embed either a KEM ind-cca challenge or a DH challenge into the root-secret derivation; advantage transfers to

$\mathsf{Adv}^{\text{ind-cca}}_{\text{KEM}}$ or $\mathsf{Adv}^{\text{dh-sec}}_{\text{DH}}$. Nonce uniqueness (§IX-B) keeps decryption oracles well-defined. Hence, for any PPT $\mathcal{A}$,

$$\mathsf{Adv}^{\text{chan-indcca}}_{\mathcal{A}} \le \mathsf{Adv}^{\text{ind-cpa}}_{\text{AEAD}} + \mathsf{Adv}^{\text{int-ctxt}}_{\text{AEAD}} + \mathsf{Adv}^{\text{prf}}_{\text{HKDF}} + \mathsf{Adv}^{\text{ind-cca}}_{\text{KEM}} + \mathsf{Adv}^{\text{dh-sec}}_{\text{DH}} + \text{negl}(\lambda).$$

e) Proof sketch (AKE authenticity, downgrade & UKS): Each side signs the transcript t_i binding (suite_id, caps, ids, ephemerals, KEM ciphertexts, roles); we derive $\mathsf{sid} = \text{HKDF-Extract}(\text{salt} = H(t), \text{IKM} = \varepsilon)$ and carry it as AAD in every record. If some party accepts without a matching peer, then either (i) a signature verifies on a transcript not produced by its signer (EUF-CMA break of Ed25519 or ML-DSA-65), (ii) H collides so two distinct transcripts yield the same hash, or (iii) sid changes without detection (contradicting AAD binding). Therefore, for any PPT $\mathcal{A}$,

$$\mathsf{Adv}^{\text{ake-auth}}_{\mathcal{A}} \le \mathsf{Adv}^{\text{euf-cma}}_{\text{Ed25519}} + \mathsf{Adv}^{\text{euf-cma}}_{\text{ML-DSA-65}} + \mathsf{Adv}^{\text{coll}}_{H} + \text{negl}(\lambda).$$

Binding of `suite_id`/caps prevents version or cipher-suite downgrade; including peer identifiers and roles prevents UKS.

f) Proof sketch (FS and PCS for the Double Ratchet): Let ℓ_K be the AEAD key length. Message keys are $MK_i = \text{HKDF-Expand}(CK_i, \text{``MK''} \parallel t_2, \ell_K)$ with one-way chain updates $CK_{i+1} = \text{HKDF-Expand}(CK_i, \text{``CK''} \parallel t_2, \ell_K)$. Learning MK_i (or CK_i) gives at most a PRF-distinguishing advantage and does not help derive CK_{i+1} (forward secrecy). At each rekey (DH or KEM), the root is refreshed as $RK' = \text{HKDF-Extract}(RK, IK_{\text{upd}})$, where IK_{upd} is independent of prior state; thus, after a successful rekey, past compromise of (RK, CK_i) becomes useless (PCS). For any PPT $\mathcal{A}$ compromising state at time τ,

$$\mathsf{Adv}^{\text{fs}}_{\mathcal{A}} \le \mathsf{Adv}^{\text{prf}}_{\text{HKDF}} + \text{negl}(\lambda), \qquad \mathsf{Adv}^{\text{pcs}}_{\mathcal{A}} \le \mathsf{Adv}^{\text{prf}}_{\text{HKDF}} + \mathsf{Adv}^{\text{ind-cca}}_{\text{KEM}} + \mathsf{Adv}^{\text{dh-sec}}_{\text{DH}} + \text{negl}(\lambda).$$

g) Proof sketch (KCI & replay).: In a KCI attack the adversary learns A's long-term key. To impersonate B to A they must either forge B's handshake signature (EUF-CMA) or compute the fresh hybrid secret (break KEM or DH). Replay fails because sid (from $H(t)$) is authenticated as AAD in the first record and turnover/epoch flags are AAD-bound invariants; any mismatch causes decryption failure.

h) Proof sketch (record integrity): Record integrity follows from AEAD int-ctxt under the nonce policy of §IX-B and the fact that all fixed-length headers (including sid, `suite_id`, `seq`, epoch/flags) are authenticated as AAD. Any valid forgery yields an int-ctxt adversary.

i) Argument (Deniability): Only the handshake transcript is signed; application messages are protected solely by symmetric AEAD under session keys known to both endpoints. Given these keys, both parties can simulate ciphertexts and plausible headers, so transcripts provide no transferable proof of authorship for message contents. Handshake signatures do not affect message-layer deniability.

C. Reference and Production

A minimal Python PoC validates logic; production should use Rust (constant-time crypto, memory safety, zeroization). Bindings to `ring`/`rust-crypto` and PQC (ML-KEM) via vetted crates are recommended.

D. Preliminary Benchmarks

Non-optimized Python on a consumer Intel Core i7:

TABLE IX
PRELIMINARY PERFORMANCE (PYTHON POC)

Metric	Value
Handshake Latency (Full AKE)	150–200 ms
AEAD Throughput (1 MB msg)	20–25 MB/s

E. Comparison

TABLE X
PROTOCOL COMPARISON FOR 1–1 MESSAGING (PROPERTIES RELEVANT TO OUR THREAT MODEL)

Feature	Ilyazh	Signal (PQXDH)	TLS 1.3 (hybrid)	MLS
PQ status	Hybrid (X25519+ML-KEM-768)	Hybrid (PQXDH)	Hybrid (drafts)	Emerging PQ suites
Transcript binding (full ctx)	**Yes** (all ctx → hash)	Partial (per spec)	Yes (per transcript)	Yes (group context)
Session ID in AAD (all records)	**Yes**	No (implicit binding)	No (record-layer differs)	N/A (group epoch IDs)
Downgrade/UKS resistance	**Explicit** (ctx+caps into hash)	Implicit/engineering	Yes (negotiation)	Yes (group)
FS & PCS	Yes & Yes	Yes & Yes	Yes & No	Yes & Yes
Mandated KEM re-encapsulation	**Yes** (cadence & limits)	Not mandated	N/A (TLS rekey policies)	Per-group policy
Interop maturity	Medium (new)	**High**	**High** (web/enterprise)	Medium (groups)
Handshake size (approx)	High (KEM+dual-sig)	Medium	Medium	High (group state)
When to prefer	Decentralized 1:1, audit & HNDL	Ecosystem today	Client–server	Large groups

XVII. Evaluation (Handshake Size and Throughput)

We provide conservative baseline numbers for a typical configuration (X25519 + ML-KEM-768; AEAD = AES-256-GCM). Runtime latency and AEAD throughput on representative mobile CPUs are provided in the public artifact repository with scripts to reproduce. We provide an open artifact with scripts: https://github.com/ilyazh/ilyazh-web3e2e (includes nonce-discipline checks, re-encapsulation cadence validation, and hardware details). The reference implementation: https://github.com/ilyazh/ilyazh-web3e2e uses constant-time primitives, zeroization, and machine-readable test vectors to ensure reproducibility.

TABLE XI
HANDSHAKE SIZE (BYTES), APPROXIMATE

Component	Bytes
X25519 public keys ($2\times32B$)	64
ML-KEM-768 public key	≈1184
ML-KEM-768 ciphertext	≈1088
Ed25519 signature	64
ML-DSA-65 signature	≈3309

References

[1] K. Cohn-Gordon, C. Cremers, B. Dowling, L. Garratt, and I. Miers. *A Formal Security Analysis of the Signal Messaging Protocol.* In IEEE European Symposium on Security and Privacy (EuroS&P), 2017.

TABLE XII
EXECUTIVE COMPARISON OF SECURE MESSAGING PROTOCOLS

Protocol	PQ?	HS size	HS RTT	FS	PCS	Group
Signal	No	∼200 B	1	✓	✓	×
TLS 1.3	No	∼250−300 B	1	✓	×	×
PQ-TLS (hybrid)	Yes	1−2 KB	1	✓	×	×
MLS	No	∼300 B+	1	✓	✓	✓
Ilyazh-Web3E2E	Yes (hybrid)	∼1.5 KB	1	✓	✓	×

[2] T. Perrin and M. Marlinspike. *The Double Ratchet Algorithm (Specification)*. Signal, 2016. https://signal.org/docs/specifications/doubleratchet/

[3] T. Perrin. *The X3DH Key Agreement Protocol*. Signal, 2016. https://signal.org/docs/specifications/x3dh/

[4] A. Langley, M. Hamburg, and S. Turner. *RFC 7748: Elliptic Curves for Security*. IETF, 2016.

[5] H. Krawczyk and P. Eronen. *RFC 5869: HMAC-based Extract-and-Expand Key Derivation Function (HKDF)*. IETF, 2010.

[6] M. Dworkin. *NIST SP 800-38D: Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC*. NIST, 2007.

[7] *NIST FIPS 203 (2024): Module-Lattice-based Key-Encapsulation Mechanism (ML-KEM)*. NIST, 2024.

[8] *NIST FIPS 204 (2024): Module-Lattice-based Digital Signature Algorithm (ML-DSA)*. NIST, 2024.

[9] R. Barnes *et al. RFC 9420: The Messaging Layer Security (MLS) Protocol*. IETF, 2023.

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

[11] Signal Foundation. *Post-Quantum Signal*. 2024. https://signal.org/blog/post-quantum/

[12] Signal Foundation. *Post-Quantum X3DH for the Signal Protocol*. 2024. https://signal.org/blog/pqxdh/

[13] E. Rescorla. *RFC 8446: The Transport Layer Security (TLS) Protocol Version 1.3*. IETF, 2018.

[14] D. Stebila, S. Fluhrer, *et al. Hybrid Key Exchange in TLS 1.3*. Internet-Draft (Work in Progress), IETF. https://datatracker.ietf.org/doc/draft-ietf-tls-hybrid-design/

[15] Cloudflare Research. *Post-Quantum TLS: experiments and deployment notes*. 2023. https://blog.cloudflare.com/post-quantum-tls/

[16] D. Stebila, C. Brendel, L. Gisin, and C. Cremers. *Security of Hybrid Key Encapsulation Mechanisms*. IACR Cryptology ePrint Archive, 2018.

[17] NIST Post-Quantum Cryptography Project. https://csrc.nist.gov/projects/post-quantum-cryptography

[18] P. W. Shor. *Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer*. SIAM Journal on Computing, 1997.

[19] V. Shoup. *A Proposal for KEM–DEM: An Encryption Scheme Based on Key Encapsulation and Data Encapsulation*. Manuscript (2001–2006). https://www.shoup.net/papers/kemdem.pdf

[20] D. Hofheinz, E. Kiltz, and T. Jager. *On IND-CCA Security of KEMs and the KEM–DEM Composition*. Lecture notes / survey materials, 2015+. https://people.inf.ethz.ch/dhofheinz/

[21] Apple Security Engineering & Architecture. *iMessage with PQ3*. 2024. https://security.apple.com/blog/imessage-pq3/

[22] D. J. Bernstein. *qrcsp: Quantitative risk of cryptosystem breakage (summary stats and discussion)*. 2023–2024. Available at: https://cr.yp.to/papers.html#qrcsp

XVIII. COMPARATIVE METRICS (INDICATIVE)

We summarize typical orders of magnitude to contextualize overheads (actual values depend on implementation and network conditions). *HS* = handshake. These numbers are indicative;

XIX. SECURITY CONSIDERATIONS

A. Nonce Management

GCM requires unique nonces under a key. We fix:

$$\texttt{nonce} = R_{64} \,\|\, \mathrm{ctr}_{32},$$

where R_{64} is chosen per (sub)chain at ratchet step from a CSPRNG; ctr_{32} increments per message and resets to 0 only when R_{64} (and the chain key) change.

B. Limits and Invariants

Mandatory rekey and session turnover:

- Rekey per chain after 2^{20} messages or 24 h (whichever first).
- Terminate session after 2^{32} total messages or 7 days.
- Reject if sequence number wraps; perform fresh AKE.

C. Downgrade & Replay Protection

Suite IDs and capabilities are in t_0, t_1, sid, and AAD. Replays of handshake messages are detected via sid uniqueness and explicit key confirmation nonces.

D. KCI, Unknown Key-Share (UKS)

UKS is prevented since sid binds both long-term identities and ephemeral material; KCI is mitigated by hybridization (adversary must break both DH or KEM path to mount meaningful impersonation).

E. Side Channels, RNG, Supply Chain

See App. XIX-E: constant-time KEM/DH; hardened RNG (hybrid entropy, periodic reseed, forward-secure DRBG); reproducible builds; hardware root-of-trust for identity keys.

F. Side-channel mitigations

- Constant-time X25519/ML-KEM (no secret-dependent branches or memory).
- Isolate ratchet state; wipe on crash; `mlock`-backed secure allocators.

G. Random Number Generation

- Hybrid entropy (HW RNG + OS); reseed every 100 ops or 1 s; forward-secure DRBG.

H. Supply Chain

- Reproducible builds; pinned dependencies; multi-vendor PQ implementations; hardware RoT for identity keys.

XX. Discussion and Limitations

While the design covers FS, PCS, and PQ security, several aspects remain open:

- **Authentication.** Default *dual-signature* (Ed25519 + ML-DSA-65) in the handshake; an opt-in "at-least-one" mode is documented as weaker (with risks).
- **Metadata privacy.** Our protocol does not hide communication patterns; additional mixnets or metadata-hiding layers are needed.
- **Implementation.** Current proof-of-concept is in Python.

XXI. Conclusion and future directions beyond the scope of this version (v0.8)

Ilyazh-Web3E2E presents a transparent, standard-based path to PQ-ready secure messaging with FS/PCS. Next steps: machine-checked modeling (Tamarin/ProVerif), side-channel hardened Rust implementation, and formal test vectors across versions. Ilyazh-Web3E2E demonstrates that a hybrid (X25519 + ML-KEM) AKE combined with a Double Ratchet can provide forward secrecy, post-compromise security, and post-quantum resilience in a decentralized messaging setting. Our specification, security model, and proof sketches aim to close the gap between practice and theory for PQ-ready secure messaging. future directions beyond the scope of this version (v0.8) include:

- Formal verification of security games using Tamarin/ProVerif.
- Efficient Rust-based implementation and benchmarking.
- Exploring group messaging (MLS-style) extensions with PQ primitives.

We hope this work motivates further research on practical, PQ-ready secure messaging protocols and accelerates adoption in decentralized contexts.

Appendix

Thanks to Professor Henry Corrigan-Gibbs (MIT) for guidance. The design reflects lessons from MIT 6.1600.

A. Double Ratchet Pseudocode (Receive)

Algorithm 2 Receiving a Message

Require: state $(\mathsf{rk}, \mathsf{ck}^R, \mathsf{ck}^S)$, header h, AAD A, ciphertext C

if h carries new DH public key **then**
  $\mathsf{rk} \leftarrow \mathrm{KDF_Root}(\mathsf{rk}, \mathrm{DH}(_{\mathrm{recv}}, h.))$
  $(\mathsf{ck}^R, \mathsf{ck}^S) \leftarrow \mathrm{KDF_Chains}(\mathsf{rk})$
end if
$(K, \mathsf{ck}^{R'}) \leftarrow \mathrm{KDF_Msg}(\mathsf{ck}^R, A)$
$(\mathtt{nonce}) \leftarrow h.\mathtt{nonce}$
$M \leftarrow \mathrm{AEAD_Open}(K, \mathtt{nonce}, A, C)$
zeroize K; $\mathsf{ck}^R \leftarrow \mathsf{ck}^{R'}$; **return** M

B. Python Snippets (Illustrative)

```python
def _kdf_chain(chain_key: bytes, ad: bytes) ->
    tuple[bytes, bytes]:
    # Derive message key and next chain key,
        separated by labels
    prk = hmac.new(chain_key, ad, hashlib.SHA
        -384).digest()
    okm = HKDF(
        algorithm=hashes.SHA-384(), length=64,
        salt=None, info=b'ilyazh/v0.8/msg'
    ).derive(prk)
    msg_key, next_ck = okm[:32], okm[32:]
    return msg_key, next_ck
```

Listing 1. Chain KDF with domain separation

```python
def derive_session(dh_bytes: bytes, kem_bytes:
    bytes, trans_hash: bytes):
    ss = HKDFExtract( salt=trans_hash, IKM=
        dh_bytes + kem_bytes )
    sid = HKDFExpand(ss, info=b'ilyazh/v0.8/
        session_id', L=32)
    rk0 = HKDFExpand(ss, info=b'ilyazh/v0.8/
        root_key',   L=32)
    send_ck_recv_ck = HKDFExpand(rk0, info=b'
        ilyazh/v0.8/chain_keys', L=64)
    return sid, rk0, send_ck_recv_ck[:32],
        send_ck_recv_ck[32:]
```

Listing 2. Hybrid secret derivation

For self-containment, we provide a minimal test vector for the handshake and one encrypted message.

- Session ID (`sid`): `d9e1...32 bytes`
- Root Key (`rk0`): `a1b2...32 bytes`
- Nonce: `4e3291d850a43b00 000001`
- Plaintext: "Hello Web3" (ASCII)
- Ciphertext: `89ab12cd... (len(PT)+16B)`

This vector ensures that independent re-implementations can validate compatibility.