AEGIS — A Secure Communication System

Anonymous Ephemeral Group-secured Interchange System Design specification & reference implementation, v1.0

Abstract

AEGIS is a communication system whose security is derived from a small set of hard mathematical problems, composed so that the whole is secure as long as *either* a classical or a post-quantum assumption survives. Its design thesis is deliberate: **content security is essentially a solved problem, so originality should be spent where the real gaps are — metadata resistance and trust establishment.** Accordingly AEGIS reuses vetted primitives for message encryption (a hybrid post-quantum Double Ratchet) and concentrates its inventive budget on two layers that mainstream messengers neglect: a privacy-preserving web-of-trust built on a cryptographic accumulator, and a peer-relayed mix network with constant-rate cover traffic.

The combination — *post-quantum ratchet + accumulator web-of-trust + universal-relay Loopix-style mixnet, presented as one coherent stack* — is the original contribution. Each ingredient rests on published mathematics; the novelty is in composition and architecture, not in unvetted cryptography. That placement is intentional: inventing a new cipher is how systems get broken, so the cryptographic core is boringly conventional on purpose.

A complete, tested reference implementation accompanies this document.

1. Introduction and motivation

Every secure messenger reduces, ultimately, to a handful of hard problems; the security *is* the mathematics. A system's identity therefore comes from *which* problems it leans on and *how* it combines them. Three properties matter:

1. **Confidentiality with forward secrecy (FS) and post-compromise security (PCS).** If a key leaks today, past messages stay secret and future messages heal.
2. **Authenticity without a central authority.** You must know you are talking to the right person without trusting a company.
3. **Metadata resistance.** *Who* talks to *whom*, *when*, and *how often* must not leak.

Property (1) is well solved by the Signal Double Ratchet and now by its post-quantum successor PQXDH. Property (2) is nominally addressed by "safety numbers," but in

practice nobody compares them, so authentication is theatre. Property (3) is where every mainstream system leaks: TLS to a central server encrypts content but hands the operator the entire social graph and lets a global observer correlate traffic.

AEGIS targets the real gaps. It reuses proven cryptography for (1) and spends its originality on (2) and (3).

---

2. Threat model

AEGIS is designed against a strong adversary, and it is explicit about what it does and does not defend.

Adversary capabilities assumed:

- **Global passive network observer.** Sees every link, every packet, all timing and volume. (This is the adversary Signal cannot defeat.)
- **Malicious infrastructure.** Any relay may be adversarial; there is no trusted server. The adversary may run a constant fraction of relays.
- **Active man-in-the-middle** attempting key substitution.
- **Endpoint compromise** of a device at some point in time (for FS/PCS).
- **Harvest-now-decrypt-later** quantum adversary: records ciphertext today, breaks elliptic curves later with a quantum computer.

Defended properties:

Property	Mechanism	Section
Content confidentiality + integrity	AEAD under ratchet-derived keys	6
Forward secrecy	symmetric KDF chain	6
Post-compromise security	DH + KEM asymmetric ratchet	6
Post-quantum confidentiality	X25519 + ML-KEM-768 hybrid	4, 5, 6
Post-quantum authentication	Ed25519 + ML-DSA-65 hybrid	4
Authentication w/o central authority	k-disjoint-path web of trust	7
Trust-graph privacy	RSA accumulator membership proofs	7
Relationship anonymity / unlinkability	onion routing over peer relays	8
Traffic-analysis resistance	constant-rate cover traffic	8
Sender/receiver anonymity from relays	layered encryption, no relay sees both ends	8

Explicitly out of scope: a compromised endpoint reading plaintext *while* compromised; a global *active* adversary who both observes all links and drops packets to perform intersection attacks over very long horizons (partially mitigated, not eliminated, by cover traffic); coercion of the user; and traffic confirmation when the anonymity set collapses to one (see §12.4).

---

3. Notation and primitives

We never invent a cipher. Every primitive is a thin wrapper over a vetted implementation. The only cryptographic originality is *composition*.

Role	Primitive	Standard
Hash	SHA-256	FIPS 180-4
KDF	HKDF-SHA256	RFC 5869
AEAD	ChaCha20-Poly1305	RFC 8439
Classical key exchange	X25519	RFC 7748
Classical signature	Ed25519	RFC 8032
Post-quantum KEM	ML-KEM-768 (Kyber)	FIPS 203
Post-quantum signature	ML-DSA-65 (Dilithium)	FIPS 204

Notation: `H(·)` = SHA-256; `KDF(·)` = HKDF-SHA256; `‖` = concatenation; `DH(a,B)` = X25519 of private `a` with public `B`; `Encaps/Decaps` = KEM operations returning `(ciphertext, shared_secret)` / `shared_secret`.

3.1 Hybrid composition

Hybrid KEM. Public key is a pair `(X_pk, K_pk)` of an X25519 key and an ML-KEM key. `Encaps` runs an ephemeral-DH KEM against `X_pk` giving `(ct_x, ss_x)` and ML-KEM against `K_pk` giving `(ct_k, ss_k)`; the combined secret is

```
ss = KDF(ss_x ‖ ss_k, info = "AEGIS-hybrid-kem")
```

This is secure if *either* the elliptic-curve or the module-lattice problem is hard (KEM combiner robustness).

Hybrid signature. Sign with both Ed25519 and ML-DSA; accept only if *both* verify. Forgery requires breaking a classical *and* a post-quantum scheme.

---

4. Layer 0 — Identity

A user is **not** a phone number or email — the single largest metadata leak in mainstream messengers, where identity *is* the social-graph handle handed to the operator. In AEGIS a user is a **self-certifying identifier**:

```
AegisID = "aegis:" ‖ Base32( H( IK.pub ‖ SIG.pub )[:20] )
```

a 160-bit fingerprint of two long-term hybrid public keys:

- `IK` — long-term **identity KEM** keypair (X25519 + ML-KEM-768).

- `SIG` — long-term **signing** keypair (Ed25519 + ML-DSA-65).

Because the ID *is* the hash of the keys, a responder can locally verify that a claimed ID matches the presented keys — a forged identity is detected without any authority.

Prekeys. To allow asynchronous session setup while offline, each user publishes a `PreKeyBundle`:

- the identity KEM public and signing public,
- a **signed prekey** (medium-term hybrid KEM public, rotated periodically),
- a hybrid signature over the signed prekey,
- an optional **one-time prekey** (single-use hybrid KEM public, best FS).

---

5. Layer 1 — Session establishment (PQXDH-style handshake)

Alice derives a shared secret with an *offline* Bob from his bundle alone. Since our long-term material are hybrid KEMs (not raw DH keys), we use the KEM generalization of X3DH — the approach used by modern post-quantum handshakes (PQXDH, KEMTLS):

```
ss1 = Encaps(IK_B)      # authenticates Bob's long-term identity
ss2 = Encaps(SPK_B)     # forward secrecy from the rotated signed prekey
ss3 = Encaps(OPK_B)     # strongest FS from a single-use prekey (optional)

SK  = KDF(ss1 ‖ ss2 ‖ ss3, info = "AEGIS-handshake-v1")
```

`SK` becomes the initial root key of the ratchet (§6). Bob's identity is authenticated *implicitly* — only he can decapsulate `ct_ik`.

5.1 Initiator authentication and the deniability trade-off

Asynchronously, Bob cannot encapsulate back to Alice, so Alice's identity cannot be authenticated implicitly in the first flight. AEGIS therefore authenticates Alice **explicitly**: she hybrid-signs the handshake transcript

```
transcript = H(AliceID ‖ BobID ‖ IK_A ‖ SIG_A ‖ ct_ik ‖ ct_spk ‖ ct_opk)
```

and Bob verifies both the signature and that `AliceID == H(IK_A ‖ SIG_A)[:20]`.

This trades some *deniability* for simplicity and robustness: an explicit signature is non-repudiable, whereas Signal's X3DH is deniable by construction.

A deniable mode is now implemented (`deniable.py`). Rather than a ring signature, it recovers deniability the same way X3DH does — with an authenticating Diffie-Hellman that only the two real parties can compute. Because AEGIS identity and prekey KEMs are *hybrid*, they carry an X25519 component, so we fold in

```
ss_auth = X25519(IK_A.x25519_priv, SPK_B.x25519_pub)    (Alice's side)
        = X25519(SPK_B.x25519_priv, IK_A.x25519_pub)    (Bob's side)
```

Only the holder of `IK_A`'s private key (the real Alice) can produce it, so authentication is implicit; and Bob could have computed it himself, so a transcript proves nothing to a third party — the exchange is repudiable and no signature ever exists. Honest limitation: the *deniability* of this mode rests on the classical X25519 leg; a fully post-quantum deniable AKE remains open, so the signed handshake stays available when non-repudiation or PQ-strength authentication is preferred. The two modes are interchangeable at session setup.

---

6. Layer 2 — Hybrid Double Ratchet

The message engine gives per-message FS (symmetric chain) and PCS (asymmetric ratchet). AEGIS's twist over Signal: **every asymmetric step mixes both an X25519 DH and an ML-KEM encapsulation** into the root KDF.

6.1 Key schedule

Root update (consumes classical + PQ material):

```
(RK', CK) = KDF_RK(RK, dh_out ‖ kem_ss)
          = split64( HKDF(dh_out ‖ kem_ss, salt = RK, info = "AEGIS-root") )
```

Chain update (per message):

```
CK'  = HKDF(CK, info = "AEGIS-chain-next")
MK   = HKDF(CK, info = "AEGIS-chain-msgkey")
key  = HKDF(MK, info = "AEGIS-aead-key")
```

6.2 The KEM ratchet and why both sides agree

A DH ratchet is symmetric (`DH(a,B) = DH(b,A)`); a KEM is not. AEGIS pairs each root update with one `Encaps` on the side that *opens* a sending chain and one `Decaps` on the side that *receives* it:

- The opener encapsulates to the peer's current ML-KEM ratchet public, obtaining `(ct, ss)`, and ships `ct` in the header.
- The receiver decapsulates `ct` with its matching private key, recovering the same `ss`.

Thus every sending-chain root update on one side is reproduced exactly as a receiving-chain root update on the other. Because the X25519 leg is symmetric and the KEM leg is reconstructed via the shipped ciphertext, both endpoints derive identical `(RK, CK)` sequences. A KEM ratchet step fires **only** when a header carries a *new* remote ratchet

public (exactly the Signal DH-ratchet trigger); repeated headers within a chain are idempotent and never re-run encapsulation.

Cost control. The asymmetric step runs only on conversational turns (direction changes), not per message, so ML-KEM's ~1 KB ciphertext is amortized. Implementations may further advance the PQ leg every `N` epochs to bound overhead; the reference advances it every asymmetric step for maximum PCS.

6.3 Header and message wire format

```
Header  = LP(ratchet_pub) ‖ PN(4) ‖ N(4) ‖ LP(kem_ct)
Message = LP(Header) ‖ nonce(12) ‖ AEAD_ct
AD      = context ‖ Header            # header bound as associated data
```

where `ratchet_pub` is the hybrid KEM public `(X_pub, K_pub)`, `LP(x)` is a 4-byte-length-prefixed `x`.

6.4 Atomic decryption (a subtle but critical rule)

A message that fails authentication must not mutate ratchet state; otherwise a forged header could force a ratchet step and desync the session — a denial-of-service / desync vector. The receiver therefore **snapshots state, attempts decryption, and rolls back on any failure**. (The reference implementation and its test suite verify this explicitly.)

6.5 Out-of-order and dropped messages

Skipped message keys are derived and cached, keyed by `(ratchet_pub, message_number)`, bounded by `MAX_SKIP`. This tolerates reordering and loss, including late delivery across a ratchet boundary.

---

7. Layer 3 — Trust (first original core)

Problem attacked: key authentication is theatre; a server can substitute keys and virtually no user notices. AEGIS makes trust *transitive, quantified, and privacy-preserving.*

7.1 Web of trust with k-vertex-disjoint paths

In-person verification (QR scan) produces a **signed attestation** — a directed edge `voucher → subject` with weight `w ∈ (0,1]`. To evaluate an unknown key we compute trust over the resulting graph:

- **Path trust** = product of edge weights along a path.
- **Aggregation** over paths by noisy-OR: `1 − ∏(1 − s_i)`, capped at 1.

- **k-connectivity requirement.** We require at least `k` **vertex-disjoint** vouching paths. In graph terms this is `k`-connectivity between root and target; forging trust then costs an attacker `k` *independent* compromises, not one central authority.

The reference finds disjoint paths greedily: repeatedly take the maximum-trust path (Dijkstra on `-log w`), remove its internal vertices, repeat up to `k` times. A key is `trusted` iff `independent_paths ≥ k` and aggregate `≥ θ` (default `θ = 0.5`, `k = 2`).

7.2 Cryptographic accumulator for graph privacy

Publishing the trust graph would itself be a metadata catastrophe. An **RSA accumulator** folds the set of trusted keys into one short value:

```
A = g^(∏ x_i)  (mod N),   x_i = HashToPrime(member_i)
```

with a trusted setup that generates `N` and discards its factors. A member proves inclusion with a **witness** `w = g^(∏_{j≠i} x_j)` satisfying

```
w^{x_i} ≡ A  (mod N)
```

revealing nothing about the rest of the set under the strong-RSA assumption. This yields zero-knowledge membership: "my key is in the trusted set" without exposing who else is. (Production should use a 3072-bit modulus or a class-group accumulator to avoid trusted setup; the reference uses 2048-bit.)

---

8. Layer 4 — Transport and metadata protection (second original core)

Problem attacked: content is encrypted, but relationships and timing are not. AEGIS combines four well-studied ideas into one transport.

8.1 Onion routing over peer relays

Each message is wrapped in layers, one per relay on a random path. A layer for relay `R` is `LP(ct) ‖ AEAD_{KDF(ss)}(inner)` where `(ct, ss) = Encaps(R.pub)` (the hybrid KEM). A relay peels exactly one layer and learns only the *next hop* — never both sender and recipient.

Every client is a relay. There is no privileged infrastructure to subpoena; the relay population *is* the user population.

8.2 Constant-rate cover traffic (the Loopix principle)

Anonymity is quantified by the **anonymity set**: the probability of being pinned as the true sender tends to `1/k` as the indistinguishable crowd of size `k` grows. To keep that

crowd full against a *global passive adversary*, each client emits **exactly one packet per round** whether or not it has something to say; a real message simply replaces a dummy "loop" packet to self. The per-client sending pattern is then statistically identical whether one is chatting or silent, defeating volume/timing correlation — the attack Signal cannot stop.

The reference demonstrates this directly: over 20 rounds, "Alice messages Bob every round" and "nobody talks" produce **identical** per-client send counts to a global observer, while Bob still receives all 20 real messages hidden among cover packets.

Honest cost. Constant cover traffic spends bandwidth and battery. That is the genuine, unavoidable price of metadata resistance; every system that omits it is choosing convenience over this property. AEGIS pays it explicitly and lets the rate be tuned (§12.1).

8.3 Packet-size uniformity

Rate uniformity defeats volume correlation, but a global observer could still use per-hop *packet size*. **This is now implemented** (`wire.py`): every packet that crosses any link is padded to a single constant `FIXED_PACKET_SIZE`. Because a layered onion shrinks as it is peeled, each relay **re-pads** the inner packet back up to that constant with fresh random bytes before forwarding, so a global passive observer sees only identically sized packets on every link and cannot use size to locate a message's position on its path. The reference verifies that a 3-hop message is carried as an identical-size packet on all three links.

This deliberately stops short of full **Sphinx**. Post-quantum Sphinx — with its fixed header and single-element blinding — relies on group homomorphisms that KEMs lack and is an open research area, so AEGIS achieves on-wire size uniformity (and hop-to-hop byte changes from layer stripping) without Sphinx's exact per-hop bitwise unlinkability or reply blocks. The honest cost is bandwidth: a short message travels padded to the same size as a handshake.

8.3a Real network transport (implemented)

The reference is no longer in-process only. `transport.py` provides an asyncio mix node that is simultaneously a relay and a client, moving fixed-size onion packets over **real TCP**: it peels and forwards packets, and emits exactly one packet per round (real or cover) per the Loopix discipline. `messenger.py` runs full end-to-end encrypted, deniable, ratcheted conversations over it, and `aegis_chat.py` is a terminal client that discovers peers through a shared directory (`directory.py`) and lets independently launched processes — on one machine or across hosts sharing the directory — actually chat.

Two metadata defences that earlier drafts left as future work are now implemented:

- **Poisson mix delays.** Each relay holds a forwarded packet for an exponentially-distributed delay (mean `mix_delay`) before sending it on, so a global observer

cannot link a packet's arrival at a relay to its departure by timing. This is the per-hop mixing that rate-uniform cover traffic alone does not provide.

- **Provider mailboxes.** The recipient is no longer the final onion hop. Each user has a *provider* relay; senders route the last hop to the recipient's provider, which stores the ciphertext under the recipient's rotating mailbox token, and the recipient later *polls* its provider to collect it (`FRAME_POLL` / `FRAME_MAILBOX`). The penultimate relay therefore sees traffic bound for a provider, not for the recipient's own node, and the recipient's node never appears as a mix endpoint. Loop cover traffic carries a reserved token and is discarded at delivery. (Honest residual: the provider learns that *its* user has mail — as in Loopix — though not who sent it; and the reference derives the mailbox token from the public AegisID rather than a provider-shared secret, so production would authenticate the poll and rotate a secret token.)

8.4 Rotating pseudonymous mailboxes

Recipients are addressed by tokens that rotate per epoch:

```
token_t = KDF(shared_seed, info = "AEGIS-mailbox" ‖ epoch)
```

so no persistent address links messages to an identity over time, and delivery is asynchronous — the recipient need not be online.

---

9. Group messaging

Encrypting once per recipient is `O(n)` per message. AEGIS uses the **sender-keys** pattern: each member derives a symmetric **sender chain** (a hash ratchet) and distributes it once to every peer over the pairwise hybrid ratchets (§6), inheriting their post-quantum protection. Thereafter a member encrypts each group message a single time under its advancing chain.

- **Per-sender forward secrecy:** the sender chain is a KDF ratchet.
- **Post-membership security:** on add/remove, senders rotate and redistribute their sender key, so removed members lose future access.

---

10. Multi-device

Each device holds its own identity keypair; a user is a small *set* of device keys bound together by cross-signatures (each device signs the others). Peers maintain one ratchet per device. This preserves ratchet security (no key sharing across devices) at the cost of `O(devices)` fan-out, resolved in practice by the same sender-key mechanism as groups. Device revocation is a cross-signature update distributed like any other message.

---

11. Key management, revocation, recovery

- **Signed-prekey rotation** on a fixed schedule limits the window of a leaked medium-term key.
- **One-time prekeys** are consumed on use; the responder rejects reuse (replay defence).
- **Identity revocation** is a self-signed revocation certificate propagated through the web of trust; vouchers drop edges to revoked keys, collapsing the target's trust score below threshold.
- **Recovery** avoids central key escrow: a user re-enrols a fresh identity and re-collects attestations. Optional social recovery splits a backup seed via Shamir secret sharing across `t`-of-`n` trusted contacts.

---

12. Security analysis (property by property)

12.1 Confidentiality / FS / PCS. Content keys are AEAD keys derived from a ratchet whose chain KDF is one-way (FS) and whose root is re-randomized by fresh asymmetric secrets each turn (PCS). Compromise of a chain key exposes only forward messages in that chain; the next asymmetric step heals.

12.2 Post-quantum. Every shared secret mixes X25519 and ML-KEM; every signature combines Ed25519 and ML-DSA. Confidentiality and authentication each survive the failure of *either* family, defeating harvest-now-decrypt-later.

12.3 Authentication. Key substitution requires either forging a hybrid signature (breaking two schemes) or forging `k` independent vouching paths in the web of trust. Single-authority substitution is impossible — there is no authority.

12.4 Metadata. Against a global passive adversary, cover traffic makes send-rate independent of conversation; onion routing hides the sender–recipient relation from every individual relay; rotating mailboxes prevent long-term linkage. Residual risk: a global *active* adversary performing long-horizon statistical disclosure / intersection attacks, and anonymity-set collapse when few users are online — partially mitigated by cover traffic and minimum-set thresholds, not eliminated.

12.5 Downgrade / MITM on handshake. The prekey bundle is hybrid-signed and the transcript is authenticated; a tampered bundle or transcript fails verification.

---

13. Comparison with existing systems

Property	Signal	Matrix/ Element	Session	Briar	AEGIS
Content E2EE + FS/ PCS	✔	✔	✔	✔	✔
Post-quantum	partial (PQXDH)	partial	✗	✗	✔ (KEM+sig hybrid, ratchet-wide)
No phone-number identity	✗	✔	✔	✔	✔ (self-certifying)
No central server	✗	federated	onion	P2P	✔ (peer relays)
Transitive/quantified trust	✗	✗	✗	✗	✔ (k-disjoint web of trust)
Trust-graph privacy	n/a	n/a	n/a	n/a	✔ (accumulator)
Cover-traffic metadata resistance	✗	✗	partial	partial	✔ (constant-rate)

(✓ = yes; ✗ = no; partial = partial or experimental. Assessment reflects public designs as of the cutoff and is meant as orientation, not a benchmark.)

The distinguishing combination is the **bottom four rows held simultaneously**: post-quantum-wide ratchet + privacy-preserving quantified trust + universal-relay cover-traffic transport, as one stack.

---

14. Functional and usability trade-offs

"Most secure" and "most functional" are in genuine tension; no mathematics dissolves it — one *chooses* where to sit. AEGIS sits deliberately toward security and names each cost:

- **Cover traffic** costs bandwidth/battery → tunable rate; a low-power profile reduces the rate at the cost of a smaller anonymity set.
- **In-person trust** costs a verification step → offset by transitive vouching so most keys are trusted without a personal meeting.
- **No central directory** costs convenient discovery → offset by opt-in directory relays that store only self-certifying IDs.
- **Larger PQ ciphertexts/signatures** cost space → amortized by ratcheting the PQ leg on turns, not per message.

---

15. Limitations and open problems

1. **Reference vs production.** The reference uses a 2048-bit accumulator and achieves on-wire packet-size uniformity via fixed-size re-padded packets (not full Sphinx); production needs class-group accumulators and, ideally, post-quantum Sphinx (an open problem) for exact per-hop bitwise unlinkability.
2. **Deniability.** Both modes now ship: the signed handshake (non-repudiable) and a deniable handshake (`deniable.py`) whose deniability rests on a classical X25519 authenticating DH. A fully post-quantum *deniable* AKE remains open.
3. **Provider trust and mailbox tokens.** Provider mailboxes keep the recipient from being the final hop, but a user's provider still learns that user has mail (as in Loopix), and the reference derives the mailbox token from the public AegisID; production should use a provider-shared secret token and an authenticated poll. Identity verification via safety numbers (`verify.py`) is available but, like Signal, depends on users actually comparing them.
4. **Active global adversary.** Long-horizon intersection attacks are mitigated by Poisson mixing and cover traffic, not solved — an open research frontier for all mixnets.
5. **Sybil resistance in the web of trust.** k-disjoint paths raise the bar but a resourceful attacker can still farm identities; pairing with a proof-of- personhood or stake is future work.
6. **Formal verification.** The protocol should be modelled in Tamarin/ProVerif; this document argues security informally.
7. **These are engineering designs, not peer-reviewed proofs.** Before protecting anyone at real risk, AEGIS needs independent cryptographic review.

---

16. Reference implementation

The accompanying Python package implements every layer with a passing test suite:

```
aegis/
  crypto.py      primitives + hybrid KEM/signature composition
  identity.py    self-certifying IDs, prekey bundles
  handshake.py   PQXDH-style KEM handshake (signed, non-repudiable)
  deniable.py    deniable handshake (classical-DH implicit auth, no signature)
  ratchet.py     hybrid Double Ratchet (DH + KEM), atomic decrypt
  trust.py       RSA accumulator + k-disjoint-path web of trust
  mixnet.py      in-process onion routing + cover traffic + mailboxes
  groups.py      sender-key group messaging
  session.py     high-level Client API
  persistence.py serialize identities + live ratchet sessions to disk
  wire.py        fixed-size onion packets + TCP framing
  transport.py   asyncio TCP mix node: relay + client + cover traffic,
                 Poisson mix delays, provider mailboxes
  messenger.py   end-to-end encrypted conversations over the mix transport
  directory.py   file-backed peer discovery (nodes + prekey bundles)
  verify.py      safety numbers for out-of-band identity verification
tests/test_aegis.py   25 tests: all layers + networked stack
demo.py               narrated end-to-end run (core layers)
net_demo.py           the whole stack over real TCP sockets, one process
aegis_chat.py         terminal chat client over the live mix network
```

Run `python demo.py` for the core-layer story, `python net_demo.py` to watch the full stack run over real sockets (encrypted, deniable, metadata-resistant, with the passive-adversary measurements), and `python -m pytest -q` for the suite. To chat interactively, launch the client in separate terminals (all sharing one directory folder):

```
python aegis_chat.py --name Alice --port 9001 --dir board
python aegis_chat.py --name Bob   --port 9002 --dir board
python aegis_chat.py --name Relay --port 9003 --dir board   # optional extra relay
```

Then in Alice's terminal: `/msg Bob hello`. The message is end-to-end encrypted with a deniable PQ handshake and hybrid ratchet, and travels as fixed-size onion packets through the mix network amid constant-rate cover traffic.

Appendix A — Design principles (the short version)

1. **Never invent a cipher.** Reuse vetted primitives; be original in architecture.
2. **Spend originality where the gaps are:** metadata and trust, not content.
3. **Hybridize for the quantum transition:** secure if *either* assumption holds.
4. **Name every trade-off.** Security-vs-usability tension is chosen, not hidden; costs (cover traffic, verification) are stated, not glossed.
5. **Decentralize trust and infrastructure**, so there is no single authority to compromise or subpoena.