The replica system — complete design

Date: 2026-05-25 **Status:** Consolidated design for the replica system as of the latest round of James-driven refinements. Supersedes the V1/V2 framing of `replicas-open-set-and-halving-2026-05-25.md` and `replicas-v1-manifest-shadow-attestation-2026-05-25.md` — the technical content largely carries over but is re-presented as a single end-to-end design with mobile attestors integrated as a launch-tier role. Deployment sequencing appears only in §13; everywhere else, the design is described as it actually is, not split into versioned subsets. **Scope:** Plain-English + technical depth on the three-tier replica system (active replicas, server-class shadow attestors, mobile attestors), the manifest mechanism that ties them together, the expansion-trigger rubric, reward economics including halving, transport choices, and a deployment roadmap. **Authority for design changes captured here:** James 2026-05-25 — (a) mobile attestors are a launch tier with tap-to-attest UX; (b) no V1/V2 framing; (c) shadow attestation as the inactive-pool security service with slashing bounty; (d) signed manifest mechanism for dynamic active set; (e) 90-day unbond; (f) halving rewards on 12-month cycle; (g) hybrid token + XMR reward model for active replicas.

> ■■ ***Alignment note (2026-06-28) — read first.*** *This consolidated design predates two later decisions; where they differ, the whitepaper (§9 registry, §11 tokenomics) and `DECISIONS.md` (D-013) are canonical: - **Epoch model is now two-cadence.** A **registry epoch** of 7 days governs snapshot content, credential validity, and revocation; a separate **attestation epoch** of ≈1 hour governs how often shadow/mobile attestors witness the snapshot and how token emission accrues. Throughout this doc, "build a `SnapshotBody` per epoch" means **per registry epoch (weekly)**, while the reward/attestation cadence ("R■ per epoch," "1 attestation per epoch," "every few hours") means **per attestation epoch (≈hourly)**. The reward magnitudes (R■ = 1,000, 8,760 attestation epochs/yr, 8.76M/replica/yr) are current under that reading. - **The 5,000 XMR mint-fee cap was removed (D-013)** — the mint fee is uncapped; there is no post-cap phase.*

---

0. Design at a glance

The replica system has three operator tiers, one signed manifest tying them together, one reward economy, and one set of triggers for growth.

- **Active replicas** sign snapshots that count toward the 3-of-N quorum clients verify. They earn the largest token reward + an XMR fee share.
- **Server-class shadow attestors** run the same protocol in parallel but their signatures don't count toward quorum. They provide divergence-detection and earn ~10% of the active rate plus slashing-bounty payouts when they catch active replicas misbehaving.
- **Mobile attestors** are a lightweight tier — a user opens an app every few hours, taps "attest," and the app signs the current snapshot. Designed for broad participation; rate-limited to one attestation per epoch per user.

A **signed manifest** lists who is currently in each tier. Foundation publishes the manifest; clients verify it against Foundation's published key. The manifest is updated as the active set grows, as shadow attestors are admitted, and as mobile users register.

The active set grows via **seven trigger conditions**, any of which fires an expansion proposal: hosting concentration, geographic concentration, quorum margin, bonded pool depth, marketplace growth, mandatory 12-month review, or a security incident.

Active replicas earn a **halving token reward** (1000 per epoch at launch, halving every 12 months) plus 1% of each registration's XMR mint fee. Shadow attestors earn 10–15% of the active token rate. Mobile attestors earn a per-attestation flat rate, rate-limited.

Operators bond tokens as collateral. **Bond is slashable** for double-signing, refusing valid registrations, attesting falsely. **Unbond cooling is 90 days.**

Replicas can run on Tor onion, clearnet HTTPS, or both — the protocol is transport-agnostic; the manifest tells clients which transport is available per replica.

Part 1 — Plain English

---

1. What the three tiers actually do

1.1 Active replicas

Five (eventually more) servers that keep the cryptographically-verifiable copy of the agent registry. They accept new registrations, gossip with each other, build snapshots once per hour, sign them, and publish them. Clients pull snapshots from any of them; clients verify that at least 3 of the active replicas signed the snapshot before trusting it.

Active replicas are the trust anchor. If they collude, the registry can be forged. The bond + slashing + transparency log + shadow attestor checks all exist to make collusion expensive and detection fast.

1.2 Server-class shadow attestors

Operators who run the same replica software on the same kind of hardware but in a parallel role. They process the same submissions, build the same snapshots, sign them with their own key — and their signature is published, but doesn't count toward the 3-of-N quorum.

Their job is **divergence detection**. If an active replica forges a snapshot, the shadow attestor's parallel-built snapshot will have a different hash. The mismatch is detectable, attributable (we can prove which replica is lying), and slashable. The shadow attestor who catches it earns a slashing bounty.

Server-class shadow attestors are infrastructure-grade operators — dedicated servers, monitored uptime, capable of full continuous attestation. They're the same kind of operator as active replicas, just in a different role.

1.3 Mobile attestors

Users with phones running the Nightshade attestor app. The app design is deliberately lightweight: **every few hours, the user opens the app and taps "Attest Now."** A tap triggers the app to pull the current snapshot from the active replicas, verify their signatures + the manifest, build its own parallel snapshot from the same input, sign it with the user's bonded key, and upload the signature.

Mobile attestors aren't always online and aren't expected to be. Each tap is one attestation. Rate-limited to one attestation per epoch per user (so users can't game it with rapid tapping). Reward is per-successful-attestation, paid in token, batched and distributed periodically.

The point of mobile attestors isn't to replace the server-class attestors — it's to dramatically grow the pool of independent witnesses. **Ten thousand mobile attestors across 100 countries is qualitatively different from fifty server attestors in 10 countries.** Even if each individual mobile attestor is less reliable than a server, the collective adds genuine geographic + jurisdictional + organizational diversity that's hard to corrupt.

1.4 How the tiers fit together at a snapshot

When an epoch closes (every hour), here's what happens:

- All active replicas independently build the same `SnapshotBody` from the gossip-converged input set, sign it with their keys.
- All server-class shadow attestors independently build the same `SnapshotBody`, sign it with their keys.
- Mobile attestors who open the app and tap during the next few hours build the snapshot for the recently-closed epoch and sign it.

A client pulls the snapshot from any active replica + all attestor signatures. They verify: 1. At least 3 of the active replicas' signatures verify (quorum check). 2. The shadow attestors' signatures match the same snapshot hash (divergence check — anyone with a different hash is signaling a forgery). 3. The cumulative shadow + mobile attestor signature count is "healthy" (a soft signal of independent witness depth).

The snapshot is trusted if (1) passes and (2) doesn't produce conflicts. (3) is a quality metric that doesn't affect trust but informs governance about pool depth.

2. The manifest — how the protocol knows who counts

Every client needs to know the current list of active replicas, shadow attestors, and mobile attestors. That list is the **manifest**.

The manifest is a signed document Foundation publishes. It contains:

- The current active replica list with their pubkeys + transports (Tor / clearnet / both).
- The current shadow attestor list with their pubkeys.
- The current mobile attestor count (individual mobile users are too many to list; they're registered separately on a per-user basis).
- The quorum threshold (derived from active replica count — typically `floor(N/2) + 1`).

- Minimum bond requirements per tier.
- An expiry timestamp.
- Foundation's signature over the whole thing.

When Foundation expands the active set or admits new shadow attestors, Foundation signs a new manifest. Clients fetch the new manifest periodically + use it for verification.

The manifest is the same data structure regardless of how the protocol matures. Today Foundation signs it with a key; eventually that key authority may move to an on-chain registry contract instead. The protocol doesn't care about the difference — it just verifies the manifest against whoever's authorized to sign.

3. When the active set grows

The active set doesn't grow at Foundation's whim. It grows when one of seven concrete signals fires:

Signal	Threshold
Hosting-provider concentration	≥40% of active replicas on the same hosting provider
Geographic concentration	≥60% of active replicas in the same legal jurisdiction
Quorum margin too thin	<2 active replicas could go offline before quorum breaks
Bonded pool depth	Bonded pool ≥2× active set size
Marketplace growth	Registration volume up >50% year-over-year
Mandatory review	More than 12 months since last expansion review
Security incident	Any detected attestation divergence, attack attempt, or compromise alert

When any of these fires, Foundation publishes an expansion proposal: which signal fired, which candidates from the bonded pool are being promoted, what the new quorum threshold will be. Token holders vote under Foundation governance. If approved, a new manifest is signed and published; clients pick it up at next refresh.

The triggers are transparent. The reasoning is on-record every time. Foundation can't expand arbitrarily; Foundation also can't refuse to expand when the criteria say it should.

4. The mobile UX in detail

The mobile attestor app is intentionally simple:

First-time onboarding: 1. User downloads the app. 2. User connects their wallet (the wallet that holds the token). 3. App walks them through bonding the minimum mobile-tier bond (provisionally 1,000 token — much

smaller than server tiers). 4. User's wallet signs a "register as mobile attestor" transaction. Foundation includes them in the next manifest update.

Daily use: 1. User opens the app — maybe 4–8 times per day, ideally roughly every 3–6 hours. 2. Main screen shows: their bond, total attestations to date, accumulated rewards, time since last attestation, time until next attestation is available (rate-limited to one per epoch). 3. User taps "Attest Now." 4. App spends ~5 seconds pulling the current snapshot from active replicas, verifying signatures against the manifest, building the parallel snapshot, signing it with the user's bonded key, uploading the signature. 5. User sees confirmation: "Attestation submitted for epoch 73,521. Reward: 0.1 token. Total attestations this week: 23."

The "every few hours" framing means: a user who taps 4 times a day (every 6 hours, on average) earns 4 × per-attestation-reward per day. A user who taps only when they remember (1–2 times a day) earns less. Rate limiting (1 attestation per epoch maximum) caps the maximum a user can earn per day at 24 attestations × per-attestation-reward.

The reward is small per tap but real. At provisional rates (0.1 token per attestation; epoch-rate-limited), a fully-engaged user could earn ~24 token per day, or ~700 token per month. Token value depends on protocol growth.

What the app does in the background: nothing. The app only runs when the user opens it. No background services, no battery drain, no iOS background-fetch hacks. The intentional low-touch design means the app passes app-store review easily and respects user resources.

What happens if the user doesn't tap for days: nothing bad. Their bond stays bonded; they earn no rewards during the inactive period. If they're inactive for >30 days, Foundation can flag them as "dormant" in the manifest and pause their tier participation; reactivation requires opening the app again. If dormant for >90 days, automatic unbond initiation (with the 90-day cooling period starting from the dormant trigger).

5. What active replicas, shadow attestors, and mobile users each earn

Role	Token reward	XMR fee share	Slashing bounty
Active replica	R■ per epoch (1000 at launch, halving every 12 months)	1% of each registration's mint fee, split per-registration among signers	n/a (active replicas can be slashed, not the slashers)
Server-class shadow attestor	0.10 × R■ per epoch (10% of active rate, halving on same schedule)	none	20% of any slashed active replica's bond if their divergence claim is confirmed
Mobile attestor	0.0001 × R■ per attestation, rate-limited to 1 per epoch (so daily cap = 24 × 0.0001 × R■ = 0.0024 × R■; halves on same schedule)	none	none — divergence detection requires continuous attestation, mobile is intermittent

The halving is uniform across all tiers — when active rate halves, shadow rate halves with it, mobile rate halves with it. All three tiers experience the same supply-curve evolution.

At launch (R■ = 1000 token/epoch): - Active replica: 1000 × 8760 = 8.76M token/year + ~5,000 XMR-equivalent in fee share - Server shadow: 100 × 8760 = 876k token/year - Mobile attestor (fully engaged at 24 epochs/day): 0.1 × 8760 = 876 token/year

After 1 halving (year 1+): all numbers halve. After 4 halvings (year 4+): all at 1/16 of launch. The hierarchy of rewards stays consistent; the absolute amounts decay together.

The economic logic: server operators (active + shadow) carry continuous protocol work; their compensation reflects that. Mobile users carry intermittent work + bond-as-capital-at-risk; their per-attestation compensation reflects that. A fully-engaged mobile user earns about 10% of what a server shadow attestor earns per year — about 1% of an active replica. That ratio is calibrated so all three tiers have positive economics but no tier is overcompensated relative to the work done.

6. The bond and slashing rules

Bond requirements (per the current manifest): - Active replica: 100,000 token + must be the operator of the replica (not custodial) - Server-class shadow attestor: 10,000 token (1/10 of active) - Mobile attestor: 1,000 token (1/100 of active) - Mint authority: 1,000,000 token (10× active replica, reflecting its higher trust position)

Slashing exposures:

Misbehavior	Active replica	Shadow attestor	Mobile attestor
Double-signing (signing two divergent snapshots)	100%	100%	100%
Refusal of valid registrations	50–100%	n/a	n/a
Stale gossip state (>2 epochs behind)	10–25%	5–15%	n/a
Late emergency overlay refresh	5–15%	n/a	n/a
Filing unfounded divergence claims	n/a	10–25%	5–15%
Persistent offline (>30 days at active/shadow tier)	flag as dormant; slash 5% per month after grace	flag as dormant; slash 5% per month	flag as dormant; no slash (user-initiated retry expected)

Unbond cooling: 90 days for all tiers (mobile users included). Slashable during cooling. After 90 days without slashing, bond returns to the user's wallet.

Slashed tokens are burned. Not redistributed. Slashing-bounty payouts (20% of slashed bond going to the catcher) come from the slashed pool before the rest burns.

7. Transport — Tor, clearnet, or both

The protocol doesn't require Tor. Replicas can run on:

- **Tor onion only** — privacy-conscious operators; replicas hidden from network observers; operators' IPs hidden from replicas. Requires both sides to have Tor.
- **Clearnet HTTPS only** — easier to deploy; mobile and consumer hosting friendly; but operator + replica IPs visible to network observers and to each other.
- **Both (dual-stack)** — the recommended default for server-class operators. Privacy-conscious users choose Tor; everyone else uses clearnet.

The manifest documents what's available per replica. Operators choose based on their privacy posture and operational constraints.

Server-class operators (active replicas + shadow attestors) SHOULD be dual-stack to give all operator types a path. **Mobile attestors are clearnet-only by necessity** — most phone OSes make Tor integration impractical, and mobile users typically already accept some network-level exposure.

8. A unified user journey: how someone becomes an active replica

This describes the bonded-pool → active-set promotion path:

1. **Day 0:** Alice decides she wants to run a Nightshade replica. She acquires 100,000 token (the active replica minimum bond) on the token's settlement chain.
2. **Day 1:** Alice runs the `nightshade-replica` binary on her server, registers with Foundation, gets included in the next manifest update as a **server-class shadow attestor** (lower bond requirement is satisfied; she runs the same software in shadow mode).
3. **Day 1 onward:** Alice's replica runs in shadow mode. It accepts submissions, runs gossip, builds snapshots, signs them — but her signature doesn't count toward the active 3-of-N quorum. She earns 10% of an active replica's reward rate + has slashing-bounty eligibility.
4. **Months later:** Foundation publishes an expansion proposal because (e.g.) registration volume grew >50% YoY. Alice's track record as a shadow attestor (uptime, consistent attestation history, geographic diversity contribution, bond age) ranks her highly in the selection.
5. **Manifest update:** Alice is promoted from shadow to active. Her signature now counts toward quorum. Her reward rate goes from 10% × R_n to 100% × R_n + XMR fee share.
6. **Steady state:** Alice operates as an active replica. Her replica binary is the same as before — only the manifest's classification changed.

The journey for a mobile user is much simpler: 1. **Day 0:** Bob downloads the Nightshade attestor app, connects his wallet, bonds 1,000 token. 2. **Day 1 onward:** Bob opens the app every few hours when

convenient, taps "Attest." Each successful attestation earns him a small reward. 3. **Many months later:** Bob has accumulated a reasonable amount of token in rewards + has shown consistent participation. If he wants, he can graduate to server-class shadow attestor (acquire more bond, run the replica binary on a server he controls) — but he doesn't have to.

Most mobile users will stay mobile. That's fine. The protocol is designed to support both casual and professional participation.

Part 2 — Technical detail

9. The signed manifest

```
pub struct ReplicaSetManifestBody {
   pub version: u32,                              // Monotonic; increments per update
   pub published_at_unix_seconds: u64,
   pub valid_through_unix_seconds: u64,           // Manifest expiry
   pub active_replicas: Vec<ActiveReplicaEntry>,         // sorted by replica_id
   pub server_shadow_attestors: Vec<ServerShadowEntry>,  // sorted by attestor_id
   pub mobile_attestor_root: [u8; 32],                   // Merkle root of mobile attestors' pubkeys
   pub quorum_threshold: u32,                     // = floor(active_replicas.len() / 2) + 1
   pub min_active_bond: u64,                      // 100_000 * 10^DECIMALS atomic
   pub min_shadow_bond: u64,                      // 10_000 * 10^DECIMALS atomic
   pub min_mobile_bond: u64,                      // 1_000 * 10^DECIMALS atomic
   pub current_halving_epoch: u32,                // Halving regime indicator
   pub base_reward_per_epoch: u64,                // R_n in atomic units
}

pub struct ActiveReplicaEntry {
   pub replica_id: [u8; 32],
   pub pubkey: [u8; 32],
   pub onion_address: Option<String>,
   pub clearnet_endpoint: Option<String>,
   pub jurisdiction_code: Option<String>,         // ISO 3166-1 alpha-2
   pub bonded_at_unix_seconds: u64,
}

pub struct ServerShadowEntry {
   pub attestor_id: [u8; 32],
   pub pubkey: [u8; 32],
   pub onion_address: Option<String>,
   pub clearnet_endpoint: Option<String>,
   pub bonded_at_unix_seconds: u64,
}

// Mobile attestors are too many to list inline; the manifest commits to a
// Merkle root of their pubkeys + bond states. Individual mobile registrations
// happen out-of-band and are committed periodically. Manifest update includes
// a Merkle proof that any specific mobile attestor is in the set.

pub struct SignedReplicaSetManifest {
   pub body_bytes: Vec<u8>,                       // canonical encoding of body
   pub authority_signature: [u8; 64],             // Ed25519 sig on manifest_hash by the signing auth
}
```

```
manifest_hash = SHA-256("nightshade-replica-manifest-v1" || 0x00 || body_bytes)
```

The **signing authority** is the Foundation key in the early operational phase, with a planned transition to an on-chain registry contract later. The protocol uses the same data structure regardless — only the source of the signature changes. Clients verify the signature against whatever authority is currently designated; the designation itself is signed by the prior authority (a single chain of signing key transitions, similar to Tor's directory authority rotation).

9.1 Manifest fetching + caching

Clients fetch the manifest from a well-known Foundation endpoint (HTTPS + Tor mirror) on first run + on cache expiry. Cached manifests are valid until `valid_through_unix_seconds`; clients refuse expired manifests and refresh.

Mobile attestor apps fetch the manifest on every tap (it's small) to ensure they're signing against the current set. Server-class operators fetch periodically (every hour or two) and verify on every snapshot they're signing.

9.2 Mobile-attestor registration via Merkle root

Listing every mobile attestor in the manifest would balloon manifest size unmanageably. Instead, the manifest commits to a **Merkle root** of the mobile-attestor set. When Foundation accepts a new mobile registration, it updates the Merkle tree, signs the new root, publishes the proof.

To verify that a specific mobile attestor's signature is authorized: 1. Client checks manifest's `mobile_attestor_root`. 2. Client checks Foundation-published Merkle proof for this specific attestor. 3. Client verifies the proof links the attestor's pubkey to the manifest's root.

This scales: 100k mobile attestors add 32-byte root + 256-byte proof per attestor (which the attestor's app provides with each signature).

10. Expansion-trigger rubric

10.1 Quantitative thresholds

(Reproduced from Doc 5 §7.1 for completeness; one canonical place to find the rubric.)

Signal	Computed	Threshold	Action
Hosting-provider concentration	`max(count by provider) / N`	≥ 0.40	Propose expansion adding under-represented providers
Geographic concentration	`max(count by jurisdiction) / N`	≥ 0.60	Propose expansion adding under-represented jurisdictions
Quorum margin	`N - quorum_threshold`	< 2	Propose expansion to increase headroom
Bonded pool depth	`(shadow + mobile bonded) / N`	≥ 2.0 (relative to active set)	Propose expansion to absorb committed standby capacity
Marketplace growth	YoY registration volume change	> 50%	Propose capacity-driven expansion
Mandatory review	Time since last expansion	> 12 months	Mandatory regardless of other signals
Security incident	Any single occurrence of attestation divergence, attack attempt, or compromise	any	Emergency review (24h Foundation publication; 7d retroactive ratification)

10.2 Process

When a trigger fires: 1. Foundation publishes an expansion proposal within 30 days. 2. Token-holder vote runs for 14 days; quorum requires >50% of cast votes + >20% of circulating supply participating. 3. If approved, new manifest published within 7 days. 4. If rejected, Foundation must re-propose within 60 days with adjusted reasoning.

Emergency path (security incident): - Foundation publishes emergency manifest update within 24 hours. - Retroactive ratification vote within 7 days. - If ratification fails, manifest rolls back to pre-emergency state at next client refresh.

11. Active replica role

11.1 Protocol responsibilities

(Same as `replicas-detail-2026-05-25.md` §7; consolidated here.)

Responsibility	Spec section	Implementation
Accept operator submissions	§17.6 + §23.1	`nightshade-replica` slice 2
Validate canonical encoding + signatures	§5–§9 + §14.3	slice 2
Build `SnapshotBody` per epoch	§18.4	slice 3
Sign with replica key	§19.1	slice 3
Anti-entropy gossip	§19 (with Q2 resolution = anti-entropy + epoch-batched deterministic)	slice 4
Receive + sign `EmergencyOverlay`	§24.3	slice 5
Maintain transparency log	§19.3 (audit fix H3)	slice 6
Persist state (sled per Q1)	(impl)	slice 7

11.2 Reward economics

- **Token reward:** R_n per epoch, where R_n = R■ / 2^n and n is the halving epoch (12-month period since launch). R■ = 1000 token. At year 1, R_1 = 500. Year 2, R_2 = 250. Asymptotes to zero.
- **XMR fee share:** 1% of each registration's 0.027182 XMR mint fee → reward pool. Split per-registration among the replicas that signed the snapshot containing that registration. Per-registration share: 0.000272 XMR ÷ (typically 3-5 signers).

- **Bond:** `min_active_bond` from manifest (100,000 token).
- **Slashing:** see §6 above.
- **Unbond cooling:** 90 days.

12. Server-class shadow attestor role

12.1 Protocol responsibilities

Identical to active replicas at the protocol level: - Accept operator submissions - Validate canonical encoding + signatures - Build `SnapshotBody` per epoch (deterministic; matches active replicas' snapshot) - Sign with own key - Run anti-entropy gossip with active + other shadow attestors - Receive + sign emergency overlay (their signature doesn't count toward quorum but is recorded for divergence detection) - Maintain transparency log (locally; published for auditing) - Persist state

The only difference: their signature **does not count toward 3-of-N quorum**. The protocol verifies their signatures independently and uses them for divergence detection.

12.2 Divergence detection

For each epoch E and snapshot S:

```
active_signatures = collect_active_replica_signatures(S, manifest)
shadow_signatures = collect_shadow_signatures(S, manifest)

# All signatures should be on the same snapshot_hash if everyone's honest
expected_hash = snapshot_hash(deterministic_construct(input_submissions_at_epoch_E))

# Active replicas claim a certain hash
active_claimed_hash = consensus_hash(active_signatures)

# Shadow attestors independently produce a hash
shadow_claimed_hash = consensus_hash(shadow_signatures)

if active_claimed_hash != shadow_claimed_hash:
   # Divergence! One side is wrong.
   # Replicas in the smaller cluster are slashable.
   file_divergence_claim(epoch=E, active=active_claimed_hash, shadow=shadow_claimed_hash)
```

When a divergence claim is filed: - Foundation governance (or eventually an on-chain registry contract) re-runs the deterministic snapshot construction from the input submission set + identifies the wrong party. - The wrong party's bond is slashed per §6 of this doc. - The shadow attestor who filed a confirmed claim earns 20% of the slashed bond as slashing bounty.

12.3 Reward economics

- **Token reward:** 0.10 × R_n per epoch. Halves on the same schedule as active rate.
- **XMR fee share:** none. (Active replicas are the ones doing the registration-processing work; shadow attestors are doing parallel attestation.)
- **Slashing bounty:** 20% of slashed active-replica bond for confirmed divergence claims.
- **Bond:** `min_shadow_bond` from manifest (10,000 token, 1/10 of active).
- **Unbond cooling:** 90 days.

13. Mobile attestor role

13.1 The tap-to-attest flow (technical)

When a user taps "Attest Now" in the app:

```
1. App fetches the current manifest from Foundation endpoint (cached for ~1 hour).
2. App fetches the current snapshot from a random active replica (caches for the epoch).
3. App verifies snapshot's quorum signatures against manifest's active_replicas.
4. App fetches the submission set for the snapshot's epoch (via gossip protocol).
5. App deterministically constructs its own SnapshotBody from the submission set.
6. App computes snapshot_hash and compares to active replicas' claimed hash.
7. If match: app signs the hash with user's bonded key + uploads to Foundation reward endpoint.
8. If mismatch: app files divergence claim (similar to server-class shadow attestor flow).
9. App displays: "Attestation submitted for epoch N. Reward: X token. Confirmation in next reward ba
```

The whole flow takes ~5 seconds on a typical mobile + wifi. The app shows a progress indicator.

13.2 Rate limiting

A mobile attestor can submit at most **one attestation per epoch**. The app enforces this client-side; Foundation also enforces server-side (only the first signature from a given mobile_attestor_id for a given epoch is rewarded).

This means the maximum daily reward for a mobile attestor is bounded by the epoch count per day: - At 1 epoch = 1 hour: max 24 attestations/day per user - Per attestation reward: 0.0001 × R_n - Max daily reward: 0.0024 × R_n token

The intent isn't for users to tap 24 times a day — it's that the cap exists so no user can extract disproportionate reward. A user tapping 4–8 times per day (every 3–6 hours, opportunistically) earns 0.0004–0.0008 × R_n token per day, which is the design target.

13.3 Anti-Sybil

Mobile attestor bond requirement (1,000 token) is the primary Sybil resistance. To register 1,000 fake mobile attestors, an attacker needs 1,000,000 token in bond — and each fake attestor's bond is slashable if it misbehaves.

Additional anti-Sybil measures: - App requires wallet signature for bond transaction (not anonymous account creation). - Foundation can rate-limit new mobile registrations (e.g., max 1,000 new attestors per day). - Optional KYC-light for mobile attestors who want higher bond limits (provisional: bond up to 10,000 token; small additional reward tier).

13.4 Reward economics

- **Token reward:** 0.0001 × R_n per attestation. Rate-limited to 1 attestation per epoch per attestor. Halves on the same schedule as active rate.
- **XMR fee share:** none.
- **Slashing bounty:** none — divergence detection requires continuous attestation, and mobile is intermittent. (Future: if a mobile attestor's tap happens to coincide with detecting an active replica's misbehavior, they could file a divergence claim and earn the bounty. Designed for; not load-bearing.)
- **Bond:** `min_mobile_bond` from manifest (1,000 token).
- **Slashing:** for unfounded divergence claims (5–15%); for double-signing within an epoch (100% — but the app's rate limiting makes this very unlikely from honest users).
- **Unbond cooling:** 90 days.

14. Roadmap — what ships when

This is the only place this document discusses sequencing. The design above is the complete design; what follows is deployment ordering.

14.1 First release (mainnet launch)

- Active replicas: 5 operators (2 team-operated + 3 external).
- Server-class shadow attestors: any operators willing to bond + run the binary in shadow mode. Bonded pool seeded from interested operators at launch.
- Mobile attestors: app launches in the same window. iOS + Android. Wallet integration depends on settlement chain choice (Hyperliquid / Solana / EVM — pending the strategic-opportunity discussion).
- Manifest signing authority: Foundation key.
- All seven expansion triggers active from day 1.
- Halving schedule begins at first epoch.

14.2 Second release (3–6 months after first)

- Expansion of server-class shadow attestor pool (more operators).
- First scheduled review of active-set expansion (likely triggered by bonded pool depth signal).
- Mobile app updates: improved UX, additional wallets supported.

14.3 Subsequent releases

- Active set expansions per the trigger rubric.
- Mobile attestor base grows organically with app marketing.
- Manifest signing authority transitions from Foundation key to on-chain registry contract once the token's settlement chain has matured. This is a procedure, not a "v2 redesign" — the protocol shape doesn't change.

14.4 What doesn't have a fixed release date

- Active set size at any given moment (depends on expansion triggers).
- Mobile attestor count (depends on app adoption).
- Specific operator identities for expansion candidates (depends on bonded pool composition at the time).

---

15. Summary

Question	Plain English	Technical
Who is in the system?	Three tiers — active replicas (full quorum), server-class shadow attestors (parallel attestation), mobile attestors (tap-to-attest).	All three tracked in the signed manifest; same `nightshade-replica` binary in different modes.
Why three tiers?	To match different operator profiles — dedicated servers for active, dedicated servers for shadow, mobile devices for casual participation.	Different bond requirements + reward rates + slashing exposures per tier; same underlying protocol.
How does the protocol know who counts?	The manifest — a signed document listing current operators per tier.	`SignedReplicaSetManifest` with Foundation signature (transitioning to on-chain registry contract later); Merkle root for mobile attestors at scale.
When does the active set grow?	When any of 7 trigger conditions fires + governance vote approves.	Quantitative thresholds in §10.1; Foundation proposal + 14-day vote + manifest update.
What do mobile attestors do?	Open app every few hours, tap "Attest Now," earn small reward per attestation.	Pull snapshot → verify → deterministically reconstruct → sign → upload. Rate-limited to 1 attestation per epoch per user.
What stops mobile attestors from gaming the system?	Rate limiting + minimum bond + slashing for misbehavior.	1 attestation per epoch per attestor; 1000 token bond per attestor; standard slashing applies.
How much do operators earn?	Active: full reward + XMR share. Shadow: 10% of active rate. Mobile: 0.01% of active rate per attestation, capped at 24/day.	All halve every 12 months on the same schedule.
Do operators have to run Tor?	No. Tor recommended for server operators; clearnet works for everyone including mobile.	Manifest documents per-replica transport; protocol is transport-agnostic.
How does mobile graduate to server-class?	Acquire more bond + run the binary on a server you control + register with Foundation.	Manifest update reclassifies the operator; same binary, different mode.
What's the roadmap?	All three tiers ship at first release. Active set grows organically per the triggers. Mobile attestor base grows with app marketing.	§14 — first release / second release / subsequent releases, all without V1/V2 labels.

16. Open design questions

Carried forward from prior docs + new ones surfaced by this iteration:

1. Halving period (provisional 12 months; could be 6, 18, 24).
2. R■ — initial per-epoch active-replica reward (provisional 1000 token; depends on launch token price + target operator USD economics).
3. Active-set growth path (provisional 5 → 7 at year 2 → 9 at year 3 → 13 at year 5).
4. Bonded-pool reward rate (shadow at 10% of active; mobile at 0.01% per attestation; could be tuned).
5. Slashing bounty % (provisional 20%).
6. Mandatory expansion review cadence (provisional 12 months).
7. Manifest cache lifetime (provisional 7 days).
8. Foundation governance vote quorum + threshold.
9. Mobile attestor onboarding KYC requirements (probably minimal for low bond; tighter for higher bond tier).
10. Mobile app: which platform first (iOS + Android together, or staged?).
11. Manifest signing-authority transition: Foundation key → on-chain contract. Specific milestone.
12. Settlement chain choice — Hyperliquid / Solana / EVM / Cosmos — pending strategic opportunity.
13. Mobile attestor wallet integration: WalletConnect, native SDKs, custom? Depends on chain choice.
14. Maximum bonded pool size — should there be a cap?
15. Multi-bond per operator: can one entity run multiple replicas at different tiers? Affects Sybil-resistance design.

Consolidated design as of 2026-05-25. Companions: `tokenomics-research-2026-05-25.md` *(wide research),* `tokenomics-option-e-detail-2026-05-25.md` *(Option E first-pass). Supersedes the framing of* `replicas-detail-2026-05-25.md`*,* `replicas-open-set-and-halving-2026-05-25.md`*, and* `replicas-v1-manifest-shadow-attestation-2026-05-25.md` *— those documents remain in the docs/ folder as the iteration history but the design described in this document is the current authoritative shape. None of these documents is a binding decision; together they describe the design in sufficient depth for counsel review + strategic-opportunity-driven final calibration.*