MAIA Request Security and Privacy Design

- **Date:** 2026-08-07 (revised 2026-08-09: NPI removed from the signature vocabulary, v1.5.173; vocabulary editions + card stamping added, §15.6, v1.5.174)
- **Status:** Comprehensive review — implemented behavior through **v1.5.169** (PRs #264–#301), plus the approved roadmap (VC/UCAN artifacts → federation → GNAP/MCP → external resource servers).
- **Baseline:** `Groups_Design.md` (2026-07-05/06), the verbatim design conversation this document traces against.
- **Audience:** MAIA maintainers; prospective **group administrators** evaluating whether to sponsor a group; **privacy and security experts** validating the design and its implementation.

> §§2–11 describe shipped, running code in the public repository (`HIEofOne/self`), with file references for independent verification. §12 states residual risks honestly. §§13–14 are the validation guides. §15 is roadmap — its first phase (patient vouch) shipped in v1.5.169; the remaining phases are designed and sequenced, not built.

Contents

1. Original intent, and what became of it

The baseline design fixed six commitments. Each is traced here to its implemented form, including where the implementation deliberately deviated.

#	Baseline commitment (Groups_Design.md)	Implemented form (v1.5.169)	Status
1	**No honeypot.** The group database holds only what membership control and mediated communication require — no clinical data, no interest profiles	The registry (`maia_groups`) stores pairwise pseudonymous ids, chosen aliases, public keys, membership status, the group's *suggested* policies, and (since v1.5.169) vouch key-bindings. Member emails are **deleted at join**; invite tokens and vouch codes are stored only as **hashes** (SHA-256). Clinical data never touches the registry	✓ Kept, strengthened
2	**Each MAIA operates its own Authorization Server** processing external requests: respond autonomously, notify the patient, or treat as spam	Implemented as delivery-time evaluation (registry host) + ingest-time evaluation (member's own host), driven by the member's own policy cards. The three-outcome dispatch is exactly: allow → autonomous response, ask/default → notify patient, deny-silent → drop	✓ Kept

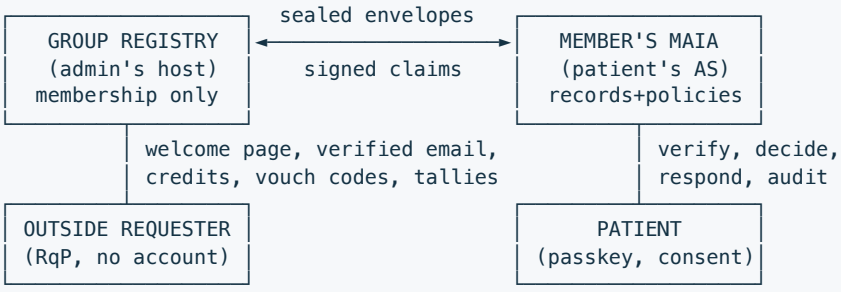
#	Baseline commitment (Groups_Design.md)	Implemented form (v1.5.169)	Status
3	**Deterministic policy control**, published by the group admin, modified by the patient. AI assists but never grants	Plain-language **policy cards** with a fixed vocabulary; group *suggested* policies are adopted as editable cards on the member's own record (patient sovereignty preserved: adopted cards can be edited or disabled). Two AI advisors exist and are structurally incapable of granting access (§9)	✓ Kept
4	**Cedar** as the policy language rather than a proprietary one	*Deliberate deviation, sequenced not abandoned*: cards are the canonical policy representation; Cedar arrives at the next vocabulary expansion as a compiled projection (§3.2). Cedar's `forbid`-overrides-`permit` semantics are already honored: the evaluation order is deny → explicit ask → allow → ask-by-default	~ Deferred by design
5	**GNAP (RFC 9635)** standardization "in a later phase"	Request envelopes were shaped for the mapping from day one (action/resource/purpose/signature/payment/nonce/created + reserved computation-class slot). GNAP triggers at the first machine-to-machine requester (§15.4)	~ On trigger, as planned
6	**Escalate-everything as the v1 policy**; first-contact handshake; autonomy earned per-sender	"MAIA asks you about everything unless you've told it otherwise" is the default outcome; accepting a sender writes an `acceptedSenders` fact (the baseline's open question 5, resolved as proposed)	✓ Kept

Baseline open questions now resolved: relay retention (Q3) = 30-day TTL with sweeps; revocation latency (Q4) = 24-hour membership credentials; autonomous resource ceiling (Q6) = resolved as a *privacy-filter* ceiling rather than a category ban — only privacy-filtered artifacts ever leave autonomously (§7.2), which permits useful autonomy while keeping raw records human-gated. The reserved shapes from Refinement 2 (computation-class field, machine attribution, issuer-role metadata) are present in the envelope and the response wording.

What v1.5.169 adds to this picture: the first *patient-issued credential*. The `verified-by-me` signature level — in the vocabulary since the policy matrix shipped, deliberately unsatisfiable until now — has a real verification path (§4.1, §15.1). It is the first identity in MAIA whose issuer is the patient rather than a platform.

One deviation to flag for reviewers: the baseline proposed **RFC 9421 HTTP Message Signatures** as the request-proofing mechanism. The implementation uses bespoke Ed25519 signed member claims (same key-binding intent, simpler surface). RFC 9421 is the natural upgrade at the GNAP phase, since it is GNAP's key-proofing mechanism; nothing in the current shape blocks that mapping.

2. Actors and trust domains



- **The patient's host is the only trust root for the patient's data.** The registry mediates; it never decides, and (for member-to-member traffic) cannot read.
- **The registry is the trust root for three attestations only**: that an outsider's email was code-verified; that an outsider proved possession of a passkey bound to a patient's vouch (`verified-by-me`, v1.5.169); and that credits were escrowed or charged for a paid request. Member ASes honor these because the registry is the sole writer of

outsider envelopes — the reserved `outsider:` sender prefix is unforgeable by members, whose relay pushes require signed member claims.

- A **member's AS may live on a different host than the registry** (federation is inherent, not special-cased — baseline §3). All flows below are verified in a two-host simulation driving the real route handlers (§14.2).

---

3. The layered architecture and its standards

3.1 Layer stack, bottom to top

Layer	Implemented with	Standard involved
Transport confidentiality	HTTPS (DO App Platform), secure cookies behind `PUBLIC_APP_URL`	TLS
Message confidentiality (relay)	Per-recipient sealed boxes: ephemeral **X25519** ECDH → **HKDF-SHA256** → **AES-256-GCM** (`sealTo` / `openFrom` , `server/routes/groups.js`)	NIST-standard primitives via Node crypto; ECIES-style construction
Message authenticity	**Ed25519** signed member claims (join, relay push, directory, vouch mint/revoke); group signing keys published at a public well-known endpoint	EdDSA
Account identity	**Passkeys / WebAuthn** (`@simplewebauthn`), temporary no-passkey fallback for new users; admin bootstrap via secret then passkey	W3C WebAuthn
Requester identity	Verified email (one-time code); **patient vouch bound to a passkey** (§4.1); signature-strength ladder (§4)	W3C WebAuthn; VC/UCAN projections pending (§15)
Policy	Plain-language policy cards, deterministic twin evaluators	— (Cedar pending as projection)
Economics	Credits ledger with escrow; Stripe Payment Link + signed webhook (**HMAC-SHA256**, timing-safe, replay-tolerant)	Stripe webhook signature scheme
AI assistance	DO GenAI private agents; two advisors outside the grant path	— (MCP touchpoint pending)

3.2 The governing principle: canonical facts, projected standards

Every standards decision follows one rule:

> **MAIA's own records are canonical; standards are projections generated at the boundary.**

- **Policy cards → Cedar.** Cards are the stored, user-visible truth; Cedar becomes the evaluation engine when the vocabulary next expands, collapsing today's twin evaluators into one compiled artifact. Cards survive; the engine is swappable.
- **Trust records → W3C VC.** The vouch shipped in v1.5.169 is already this shape: issuer (the patient), subject key (the requester's passkey), revocation state — stored as a MAIA record. The signed VC artifact appears in Phase 2 for portability, a serialization of facts that already exist.
- **Authority grants → UCAN.** When delegation artifacts appear (Phase 2), the *authorization* artifact is UCAN-shaped from the start (`iss` / `aud` /capability/caveats/ `prf`) — chosen over ZCAP-LD because UCAN needs no JSON-LD canonicalization, its `did:key` principals are just encoded public keys (every key MAIA holds maps directly), and its ⊆-attenuation rule is literally MAIA's existing `SCOPE_COVERS` subsumption check.
- **Request envelopes → GNAP.** The envelope fields were named for the mapping: access rights ≈ scope, client keys ≈ signature, interaction ≈ ask-outcome, continuation ≈ pending→decision.
- **Requester-side agents → MCP/A2A.** Neither protocol has native delegation semantics; both outsource authorization to OAuth-shaped token slots. A serialized UCAN rides in that slot; MCP's known weaknesses (bearer-token passthrough, confused deputy) are precisely what audience-bound attenuated tokens eliminate. The expected first machine requester is an MCP client, making the GNAP phase and the MCP touchpoint one build.

- **External resource servers → conventional JWTs.** At the hospital FHIR boundary (§15.5) the AS *flattens* any internal chain into a boring structured token. Complexity stays on MAIA's side of the seam.

Two disciplines applied **now** so the future layers cost no rework: (1) every credential or capability is **audience-bound to a specific key, never bearer** — already true of the shipped vouch record, and an agent hand-off is therefore always an explicit new chain link, never a token copy; (2) delegation caveats (`delegable: no / once / agents-allowed`) enter the card vocabulary at the Cedar phase, not before.

3.3 Why this layering is reviewable

A reviewer can audit each layer independently: the sealing construction without reading policy code; the evaluator without reading crypto; the credits escrow without either. The twin-evaluator parity tests (§14.1) pin client and server policy semantics to each other, and the two-host simulation pins the federation seams. The projection principle means no standard adoption can silently change stored user facts.

---

4. Identity: the signature-strength ladder

A policy card states the **minimum** identity proof it requires; stronger always qualifies. The ladder, with each level's *proof mechanism*:

Level	Proof today	Honest-evaluation rule
`unverified`	none	floor
`verified-email`	one-time 6-digit code (10-min TTL, 5 attempts, 30 s resend gap, in-memory token)	attested by the host that ran the code flow
`group-member`	Ed25519-signed membership claim against a 24 h credential	proven cryptographically at the relay
`doximity`	**claim only** — no live verification exists	**evaluates as `unverified`** until real verification ships. (An `npi` level existed through v1.5.172; it was removed from the authorable vocabulary as obsolete — legacy stored cards requiring it keep their strict rank and remain unsatisfiable)
`verified-by-me`	patient-minted one-time code, redeemed into a **passkey binding**; a WebAuthn assertion proves possession at send time (v1.5.169)	attested **per member**: elevated only toward the patient who vouched; revocation re-checked on every send

The honest-evaluation rule is a standing invariant (I-6): **an identity claim never evaluates above what was actually proven.** The UI says so explicitly in the policy matrix tooltips, so patients authoring cards are not misled about what a "Doximity verified" claim currently buys.

4.1 The vouch flow (`verified-by-me`)

The patient is the issuer, and — decisively for the architecture — also the only evaluator that will ever honor the credential. That collapses issuer discovery, trust registries, and revocation propagation into a local lookup.

1. **Match, out-of-band.** The patient recognizes the person on a live voice or video call. **Biometrics never enter MAIA**: the code hand-off *is* the trust event, and no voice or face data is captured, transmitted, or stored.
2. **Mint** (patient's own host): Workbook → Sharing Policies → *People I vouch for*. The patient picks the group, labels the person for their own list, and receives a 6-character code — displayed once, never stored in plaintext. Their MAIA registers only the code's **SHA-256** at the registry via a signed member claim. The registry never learns the code or the label.

3. **Redeem** (registry's welcome page): the requester enters the code and creates a **passkey** (discoverable credential, user verification required). The vouch record binds *credential public key ↔ vouching member's pairwise id*. Codes are single-use with a 24-hour TTL; ceremonies are rate-limited and challenges expire in 5 minutes.
4. **Present:** a WebAuthn assertion mints a 10-minute possession token. The request then carries `verified-by-me` **only in the envelope sealed to the vouching member** — per-member sealing makes per-member signature levels free, and every other member still sees at most `verified-email` (I-19).
5. **Revoke:** one click on the patient's list flips the registry record. Every send re-reads it, so revocation beats an unexpired possession token, and later assertions with that credential are refused outright (I-20).

Two design notes for reviewers. The credential is **host-locked by construction** — a WebAuthn credential is scoped to the registry's rpID — which is why Phase 1 chose theft-proof over portable (§12.2-10). And the record binds a **subject key, never a bearer secret**: that key is the principal a future UCAN delegation chain, or a GNAP client-key binding, terminates in.

---

5. The policy layer

One vocabulary, everywhere. `POLICY_MATRIX` (`src/utils/policyCards.ts`) is the single source for the five axes — scope, purpose, signature, payment, action — rendered by one component (`PolicyMatrix.vue`) in every context (welcome page, request builder, card authoring, simulation), with inapplicable cells *disabled with teaching tooltips* rather than hidden. Visitors learn the whole model before holding an account. Where a live page disproves a static disable — a visitor who has just proven a vouch — an `unlock` prop lifts exactly that cell.

Deterministic evaluation, fixed order: `deny → explicit ask → allow → default ask`. A deny card always wins; an explicit "Ask me first" card carves a human-approval requirement out of any broader allow; anything no card covers escalates to the patient. Deny splits into **silent** (the spam answer — requester learns nothing, not even that the request was seen) and **respond** (a decline notice that never names the deciding card).

Twin evaluators, pinned by tests. The matcher exists in `server/routes/policies.js` and `src/utils/policyCards.ts`; `tests/backend/policy-ask.test.js` holds them to identical semantics. Scope subsumption (`SCOPE_COVERS`: everything ⊃ not-sensitive, patient-summary ⊃ meds-allergies, …) is shared. This duplication is the standing argument *for* the Cedar migration; until then the parity suite is the guard.

Write-path guard. Every card save — user-authored, group-suggested-then-adopted, or AI-proposed — passes `normalizeCard`, which enforces vocabulary membership and shape. There is no policy write path that bypasses it.

Sovereignty. Group-suggested policies arrive as *editable cards on the patient's own record*, provenance-tagged `group:<id>`. The admin's authority is membership and suggestions; the patient's evaluator only ever reads the patient's own cards. This implements the baseline's two-layer/one-sovereignty rule without a live dependency on the registry at decision time.

---

6. Request lifecycles

6.1 Outside request (W3): anyone → every group member

1. Requester composes on the group's public welcome page: scope, purpose, message, optional payment; verifies email by code; optionally redeems or asserts a vouch (§4.1).
2. Registry validates (vocabulary, email regex, verified token). If a payment is attached it **moves credits before any delivery** (§8); if a vouch token is presented it **re-reads the vouch record**, so a revoked vouch cannot be spent.

3. Per active member, **delivery-time evaluation** runs when the member's record is resolvable on this host — at that member's own attested signature level, which is `verified-by-me` for the voucher and the base level for everyone else. The outcome (autonomous accept with privacy-filtered artifact by email / autonomous decline / escalate) is embedded as `autoDecision`.
4. Each member receives an individually **sealed envelope** (X25519/HKDF/AES-GCM) under the reserved `outsider:` sender prefix — members cannot forge outsider traffic because member pushes require signed claims.
5. Cross-host members decide at **ingest** on their own host (client refresh, or the hourly mail pull that bounds offline latency to ~1 h), answer requesters directly, and bump the registry's public tally through a capability-URL endpoint (knowing the unguessable `reqId` *is* the authorization).
6. The requester watches a **counts-only tally** (delivered/responded/accepted/declined — no identities, no content).

6.2 Member → member

Sender's host seals to the recipient's pairwise key and pushes with a signed member claim; the registry relays ciphertext it cannot read. The recipient's own AS evaluates at ingest (party = group, signature = group-member). Autonomous replies travel as sealed relay messages (reaching any host) plus direct email when the recipient's host knows the requester. First-contact requests escalate unless a card decides; accepting writes the `acceptedSenders` fact.

6.3 Reliability properties that carry security weight

- **Idempotent ingest:** request docs use deterministic ids derived from the relay message id, so racing refreshes cannot duplicate a request or *reset a human decision already made* (an upsert store makes this a genuine hazard; the id scheme neutralizes it).
- **Nudge debouncing** (6 h quiet period) is time-based, not queue-count-based — a member who is away still gets exactly one nudge, and bulk requests cannot weaponize notifications.
- **Delivered ≠ failed:** post-delivery bookkeeping is conflict-retried and best-effort, so a message that reached its recipient never reports as a send failure (misreporting outcomes is a trust bug, not just a UX bug).

---

7. Privacy architecture

7.1 Registry minimalism (the no-honeypot commitment, mechanized)

What the registry knows: group metadata, pairwise ids, aliases, public keys, invite-token *hashes*, suggested policies, sealed ciphertext in transit (30-day TTL), counts-only tallies, vouch records (the code's SHA-256 as the document id, the redeemed credential's public key, the vouching member's pairwise id, lifecycle timestamps), and — for paid requests — the payer's email and settlement state (disclosed in §12 as a known metadata cost).

What it structurally cannot know: member emails after join (deleted, by design — which is *why* cross-host notification requires the member host's hourly pull), member-to-member message content (sealed end-to-end), members' policies, records, or request outcomes beyond counts, the vouch code itself, or who a vouched person is (the patient's label for them never leaves the patient's own record).

Different groups see different pseudonyms for the same patient — no cross-group correlation by identifier, exactly as the baseline required.

7.2 The privacy filter: the ceiling on autonomy

Only two artifacts can ever leave autonomously: the **privacy-filtered Patient Summary** and **privacy-filtered Current Medications**. The filter (`server/privacyFilter.js`) replaces person names with obviously-fake pseudonyms via a layered extractor: credential-anchored detection (MD/DO/NP/RN/…), structural exclusions (parenthesized names

after credentials), a stoplist and all-caps/acronym organization rules, file-name and header formats, plus client-side masking for legends and tooltips. Three properties matter more than the heuristics themselves:

1. **Fail-closed:** if no filtered artifact exists (summary never verified), an allow card does **not** fire — the request escalates to the human rather than sharing nothing-or-raw.
2. **User-correctable:** the pseudonym mapping is visible and editable; deletions become *tombstones* that healing passes never resurrect.
3. **Self-healing:** every summary verify/edit refreshes the filtered artifact, so the shareable copy tracks the record without manual steps.

The filter is heuristic and is honestly characterized as such in §12. Note that this ceiling holds regardless of identity strength: even a `verified-by-me` requester receives only the filtered artifact autonomously — a stronger identity buys *automation*, never *more data*.

7.3 Disclosure rules (what requesters and members may learn)

- **Requesters never learn which policy card decided** — outcome and artifact only. The deciding card's sentence is recorded solely on the member's own request log.
- Silent deny is *silent*: indistinguishable from absence — the baseline's match-silence principle applied to requests.
- Autonomous responses carry **machine attribution** ("…their MAIA responded autonomously"), never impersonating the human (Refinement 2's non-negotiable, implemented in every autonomous email template).
- Tallies expose counts only; the status endpoint requires the unguessable request id.

7.4 Retention

Verified emails of non-members purge after 72 h; relay messages and tallies expire at 30 days (sweep settles any open escrow first, §8); invites expire at 14 days; membership credentials at 24 h; vouch codes at 24 h (the redeemed key-binding persists until revoked, by design); inboxes and ledgers are size-capped. Nothing in the request path accumulates unboundedly.

---

8. Economic controls: credits

Money as a spam filter and fairness tool (the baseline's §7 posture, made concrete):

Instrument	Price	Mechanics
Spam evaluation deposit	5 credits	escrowed at send; **returned** on any accept or decline-with-reason; **forfeited** if every member silently ignores it until the 30-day expiry
Request evaluation payment	2 credits	charged at delivery
Sharing payment	25 credits	escrowed; **captured** on first accept; returned if none

Design properties reviewers should check (all tested, §14.1): credits are **non-refundable and non-transferable** (prepaid service fees, not stored value — a deliberate money-transmitter-avoidance posture); accounts are keyed by verified email on the host of purchase; **one payment covers the whole request** regardless of member count; every escrow settlement is **idempotent by request id**, so conflict retries and racing sweeps cannot double-settle; the envelope's payment slot is **registry-attested**, so a member card *requiring* payment matches only genuinely paid requests, cross-host included; card `payment: none` matches any request (a payment can never *reduce* access).

Purchases run through a Stripe Payment Link — **MAIA never touches card data**. The webhook that automates granting verifies Stripe's signature (HMAC-SHA256 over the exact bytes, timing-safe compare, 5-minute replay

tolerance, no SDK), is idempotent per event id *and* per ledger ref, and its signing secret is write-only (no endpoint returns it). Captured and forfeited credits fund the host; the admin declares a surplus charity, disclosed to buyers at purchase.

9. AI boundaries

The standing rule from the baseline, kept absolute: **AI output never grants access.** Four AI surfaces exist; none is in the grant path.

1. **Autonomous responses** contain no AI at decision time — the deterministic evaluator decides; AI is not consulted (baseline §4's "on the permit path there is no AI", implemented literally).
2. **Patient-side policy advisor**: a private AI that reviews the patient's own cards, request log, and record profile, and *drafts* cards. Proposals travel as fenced JSON, are re-checked by the deterministic evaluator in front of the user, and are saved only by explicit user action through `normalizeCard`. Context is server-assembled and **never injected for deep-link or shared-chat sessions**.
3. **Group request advisor**: a public assistant on the join page fed **only** what the join page already publishes (description, posting policy, suggested cards, mechanics). It cannot see any member's policies, records, or history — and its system prompt instructs it to say so. Gated by verified email; rate-limited; metered by credits past the free tier.
4. **Summary drafting** (single-player substrate) is consent-gated by the CM/PS invariants: a verified Patient Summary is **never overwritten without explicit confirmation**, and list edits clear verification stamps honestly.

Prompt-injection exposure is bounded by construction: advisor outputs are parsed as data (fenced JSON → vocabulary validation → deterministic re-evaluation → human confirmation), never executed as instructions.

10. Consent invariants

- Ask-by-default: anything no card covers reaches the patient as a question.
- Explicit "Ask me first" cards pin human approval even inside a broader allow.
- Accept/decline of an outside request is a per-request human act; *block* is silent (spam gets no reply).
- Issuing a vouch is a deliberate patient act with a named subject and one-click revocation; it grants standing *identity*, never standing access — the cards still decide, every time.
- Verified stamps are honest: only a human Verify sets them; any edit clears them; the server records the caller's real flag.
- The Patient Summary update after a medications verify is offered, diffed, and applied only on confirmation — patching the meds section while preserving the verified stamp, never regenerating by AI without consent.

11. Data residency

Data	Registry host	Member's host	Client (browser/device)
Clinical records, KB, summaries	—	✓ (CouchDB + Spaces; keys per user)	✓ (File System Access folder — primary custody)
Policy cards	suggested only	✓ canonical	rendered
Membership (pairwise id, alias, pubkeys)	✓	✓ (own memberships)	—

Data	Registry host	Member's host	Client (browser/device)
Member email	**deleted at join**	✓	—
Relay messages	ciphertext, ≤30 d	plaintext inbox (capped) after pull	rendered
Request outcomes	counts only	full log incl. deciding card	rendered
Vouch credentials	code hash (as id), credential public key, voucher pairwise id	the patient's labelled list (`vouchedParties`)	passkey private key in the requester's authenticator
Credits	✓ (payer email + ledger)	—	balance display
Verify tokens / codes	in-memory, ≤10 min (+72 h email retention rule)	same mechanism on its own host	token in page state
Passkeys	vouch credential public keys	account credential public keys	private keys in authenticator
Audit log	✓ own events	✓ own events	—

12. Threat model and residual risks

12.1 What the design defends against

Threat	Defense
Bulk/spam requests	ask-by-default; silent deny; spam deposits with forfeiture; rate limits on advisor and verification; nudge debouncing
Identity inflation ("I'm a doctor")	honest-evaluation rule: claims score as `unverified` until proven; the ladder is proof-based
Vouch code interception or guessing	single-use, 24 h TTL, out-of-band hand-off; 6 characters over a 31-symbol alphabet ($\approx 8.9 \times 10^8$ combinations); redemption rate-limited per IP; once redeemed the code is spent, so a later leak is worthless
Stolen or replayed vouch credential	key-bound, never bearer: each session needs a fresh WebAuthn assertion; possession tokens last 10 minutes; every send re-reads the revocation state
Over-broad vouch elevation	elevation applies only in the envelope sealed to the vouching member; all other members see the base level (I-19)
Member forging outsider traffic	relay pushes require signed member claims; `outsider:` prefix writable only by the registry
Registry reading member traffic	end-to-end sealing to pairwise keys; registry holds ciphertext only
Cross-group correlation	pairwise pseudonyms per group
Replay	nonces on envelopes; idempotent ingest ids; Stripe timestamp tolerance; single-use hashed invite tokens; expiring WebAuthn challenges
Race/double-spend	conflict-retried read-modify-write everywhere money or decisions move; idempotent settlement and grants (tested)
Requester learning policy internals	outcome-only disclosure; counts-only tallies; card sentences stay on the member's log
Token theft (email verify)	short TTL, attempt caps, host-locality; tokens never in query strings (POST bodies only)
Payment card exposure	Stripe-hosted checkout; MAIA sees signed events only; write-only webhook secret
Unconsented AI overwrite	CM/PS confirmation dialogs; stamp honesty; advisor save-path validation

12.2 Residual risks, stated plainly

1. **The registry reads outsider request plaintext.** It must (it seals per member). A compromised registry could also *falsely attest* signature strength, vouch status, or payment on outsider envelopes — member ASes trust those attestations. Mitigations today: the registry never holds clinical data, and member-to-member traffic is sealed past it. The Phase-2/3 roadmap (patient-signed and group-signed artifacts verified at ingest) progressively removes the registry from the trust path.
2. **The privacy filter is heuristic.** Regex-and-rules extraction has no formal guarantee; a novel document format can leak a name. Mitigations: layered extraction with healing passes, the user-visible mapping with per-row deletion and tombstones, fail-closed autonomy when no filtered artifact exists. Reviewers should treat filter bypass as the highest-value red-team target (§14.3).
3. **Email is the requester-response transport.** TLS in transit, but delivered artifacts rest in requesters' mailboxes outside MAIA's control. This is a deliberate reachability trade-off; capability-gated pickup pages are a possible future hardening.
4. **Verified-email state is in-memory and host-local.** A process restart drops pending verifications (re-verify; no security loss, some friction), and a token proves control of an address only to the host that ran the code flow.
5. **Payment metadata at the registry.** A paid request links a payer email to a group and settlement state on the registry host. Disclosed in §11; minimal but nonzero.
6. **Root-secret concentration.** The DO token derives the CouchDB password, session secret, and admin bootstrap passphrase. Rotation is documented but manual; compromise of the host environment is compromise of the deployment. (Single-admin-per-host is the current model.)
7. **Silent-deny forfeiture can burn an honest requester's deposit** when every member simply ignores a legitimate request. Accepted deliberately (silence is what the deposit prices); the amounts are small by design.
8. **Doximity remains a claim.** Cards requiring it are currently unsatisfiable in practice; the UI says so, but a reviewer should confirm no code path scores it higher. (The former `npi` level was removed from the authorable vocabulary in v1.5.173; legacy stored cards requiring it stay ranked — a deliberate choice, since dropping a level from the rank map would make such cards match *everything*.)
9. **A vouch is only as strong as the patient's out-of-band recognition.** Voice cloning and video deepfakes are practical today; a patient who vouches for a convincing impersonator has issued a genuine credential to the wrong person. No cryptography fixes this — the mitigations are procedural (the UI directs the patient to recognize the person on a *live* call, not from a recording or a forwarded message), plus one-click revocation, per-member confinement, and the standing rule that even a vouched requester receives only privacy-filtered artifacts (§7.2). This is the honest weak point of patient-issued credentials and should be stated to any group considering them.
10. **Vouch passkeys are host-locked.** A WebAuthn credential is scoped to the registry's rpID, so a vouch proven at one host cannot be presented at another, and a lost device means re-vouching. Deliberate for Phase 1 (theft-proof beats portable); Phase 2's signed artifact adds portability without giving up key binding.
11. **The registry learns the vouch graph.** It knows which member vouched, when the credential was redeemed, and each time it is asserted — not who the person is (no name, no email required) but the existence and use pattern of the relationship. This is the metadata cost of the registry-attested model; Phase 2/3 artifacts verified at ingest move it to the member's host.
12. **Vouch-code guessing scales with outstanding codes.** The per-IP rate limit is the practical control; a distributed attacker facing many simultaneously-outstanding codes has proportionally better odds than the raw $\approx 8.9 \times 10^8$ space suggests. Codes are single-use and short-lived, so the exposure window is small — but a deployment expecting large-scale vouching should add a global attempt budget (§14.3-7).

---

13. Validation guide for group administrators

What you can verify **before** sponsoring a group, from the outside:

1. **The no-honeypot claim is inspectable.** Fetch the public group info endpoint and the join page: everything the registry publishes is visible there. Ask your counsel to compare §11 against your liability posture — the design goal is that *the sponsor holds membership, never PHI*.
2. **Suggested policies are suggestions.** Adopt one as a test member, edit it, disable it — confirm your edit governs and the group copy does not overwrite it.
3. **Run the outsider path yourself.** Send a request to your own group from the welcome page (verified email; optionally attach a deposit). Confirm: the tally shows counts only; the response email never names a policy card; a silent-deny card produces indistinguishable silence.
4. **Exercise a vouch end-to-end.** Vouch a colleague from a test member account, redeem the code in a private window, and confirm three things: their request reaches *your* MAIA as `verified-by-me`, a *second* test member sees them at no more than `verified-email`, and revoking downgrades the very next request.
5. **Check the audit trail.** Every delivery, autonomous decision, human decision, vouch creation/redemption/revocation, and broadcast lands in the audit log with policy ids on the member side and counts on the registry side.
6. **Money handling.** Confirm the purchase flow never leaves Stripe-hosted pages, the charity disclosure appears at purchase, and the admin panel's earned/held/outstanding totals reconcile with the ledger after your test request settles.
7. **Exit is real.** Backup/restore to another host is a shipped path; membership revocation and the 24 h credential expiry bound how long a removed member's claims verify.

---

14. Validation guide for privacy and security experts

14.1 The executable specification

The invariants in Appendix A are pinned by test suites a reviewer can run in minutes (`npm run test:backend` — **135 tests**, no CouchDB required; in-memory fakes drive the **real route handlers**):

Suite	Pins
`tests/backend/crosshost-autonomous.test.js` (17)	two-host federation: sealed delivery, ingest decisions, tally capability endpoint, idempotent re-refresh, nudge debounce, hourly pull, paid-request escrow end-to-end (hold → cross-host release; capture on paid card match; unpaid escalates; 402 before delivery; expiry forfeiture), and the vouch lifecycle (hash-only registration, single-use redemption, per-member elevation with an unvouched control, revocation beating a live token)
`tests/backend/credits.test.js` (12)	ledger semantics, hold/release/capture/forfeit idempotency, webhook signature verification incl. tamper/stale/replay, grant idempotency under redelivery
`tests/backend/privacy-seed.test.js` (13)	extraction formats, structural exclusions, tombstone suppression
`tests/backend/policy-ask.test.js` (6)	client/server evaluator parity; deny → ask → allow → default order

Suite	Pins
`tests/backend/db-prefix.test.js` (4)	test/prod isolation on shared infrastructure

14.2 Methodology worth reusing

The two-host simulation (`FakeCloudant` + fake `fetch` routing between two in-memory apps running the production handlers) is the reference way to interrogate federation claims: any "the registry cannot X" statement in this document can be turned into an assertion there. WebAuthn ceremonies are injected through the route module's dependency object, so credential flows are exercised end-to-end without a browser authenticator — and a reviewer can substitute a hostile implementation to test failure paths.

14.3 Suggested red-team exercises, in value order

1. **Privacy filter bypass**: craft summary/med formats that smuggle person names past extraction (§12.2-2). The mapping UI and tombstones are in scope as mitigations.
2. **Attestation forgery**: attempt to make a member-pushed relay message evaluate as `outsider:` or carry an inflated signature/vouch/payment attestation without registry cooperation.
3. **Vouch confinement**: try to get a vouched request evaluated as `verified-by-me` by a member who did *not* vouch (I-19), or to elevate without a valid assertion.
4. **Revocation lag**: spend a possession token minted *before* revocation; it must fail, because the send re-reads the record (I-20).
5. **Card-disclosure leakage**: diff requester-visible outputs (emails, tallies, status endpoints) across policy configurations that produce the same outcome — they should be indistinguishable.
6. **Settlement abuse**: race decision/responded/sweep paths for double-release or double-capture; replay Stripe events; attempt grants via unsigned webhook posts (must 503/400).
7. **Code-space pressure**: measure real redemption-attempt throughput against many outstanding codes (§12.2-12) and judge whether a global attempt budget is warranted for your deployment size.
8. **Correlation attacks**: attempt cross-group linkage of one patient from registry-visible data alone, including the vouch graph.
9. **Advisor injection**: adversarial group descriptions/posting policies attempting to steer the public advisor into disclosing non-public facts (it holds none — verify) or emitting cards that survive `normalizeCard` with out-of-vocabulary content.

---

15. Roadmap: credentials, delegation, and external resource servers

Approved sequencing, with the triggers that start each phase. Phase 1 has shipped; the rest are designed and sequenced, not built.

15.1 Phase 1 — patient vouch (`verified-by-me`) — SHIPPED in v1.5.169 (PR #301)

Out-of-band recognition by the patient; one-time code; passkey binding at redemption; registry attestation confined to the vouching member; one-click revocation re-checked on every send. Full mechanics in §4.1. The structural facts that made this the right first phase: issuer == evaluator (no PKI, no revocation propagation), per-member sealing already permitted per-member signature levels, and no biometric data ever enters the system. What it does *not* yet do — portability beyond the minting host, and delegation — is exactly Phase 2.

15.2 Phase 2 — signed artifacts, split by role

Identity claims (the vouch, group-admin credentials) project as **VCs**; authority grants become **UCANs** (audience-bound, attenuable, `SCOPE_COVERS` as the ⊆ rule; card vocabulary as caveats). The split matters: VCs answer *who is this*, capabilities answer *what may this key do*. Trigger: a portability need (presenting a vouch at a registry other than the minting one) or the first group-admin-issued credential.

15.3 Phase 3 — federation as delegation chains

Registry → admin → member → RqP chains, cross-group trust as cross-signed delegations verified against already-published group keys; enforcement stays at the patient's AS (chain validation + local revocation + cards as backstop). Cedar lands at this vocabulary expansion, collapsing the twin evaluators.

15.4 GNAP / MCP phase

Trigger: the first machine-to-machine requester — expected to arrive as an MCP client. GNAP access token = serialized UCAN; RFC 9421 message signatures; an MCP-facing facade for requester-side agents; offline attenuated re-delegation with enforcement (and the patient's cards) still authoritative at invocation. Neither MCP nor A2A has native delegation semantics today; both leave an OAuth-shaped token slot that an audience-bound UCAN fills, which is what removes their bearer-token and confused-deputy exposure.

The standing invariant across all phases: **capability issuance is consent-gated by cards; invocation re-checks cards and revocation at the patient's AS.** Delegation decentralizes; enforcement does not.

15.5 External resource servers: the hospital FHIR boundary

The GNAP phase raises the complementary question: when the patient's records also live behind a hospital's standard FHIR API, what must the hospital do to become a **Resource Server honoring the patient's AS tokens**? The answer splits into two modes, because the technical delta is small but the trust inversion (hospital keeps the RS role, cedes the AS role per patient) is organizationally large. This is the UMA 2.0 / HEART federated-authorization pattern; GNAP is its successor, and the RS side is specified in RFC 9767 (GNAP Resource Server Connections).

	Mode 1 — Bridge	Mode 2 — Native RS
Hospital changes	**None**	Gateway-level token validator + per-patient AS registration
Who is the AS for hospital data	Hospital (its own OAuth/SMART AS)	The patient's MAIA AS
How MAIA participates	Registers as a SMART on FHIR app on the certified §170.315(g)(10) API; patient approves once; MAIA holds the token and **re-gates every downstream use with the patient's cards** (requesters never touch the hospital token; only privacy-filtered artifacts leave)	Requester's client presents a MAIA-issued, key-bound GNAP token directly to the FHIR API
Deployable	Today, unilaterally	Per willing partner

Mode 2 checklist (what the hospital literally does):

1. **Patient↔AS registration** — bind MRN / FHIR `Patient` id to the patient's AS URI + published keys; carried by a signed HIPAA right-of-access direction (UMA's "resource owner introduces the AS to the RS"). Per-patient federation; no global trust framework needed to start. Identity proofing can ride TEFCA IAS practice.
2. **Token validation at the FHIR front door** (API-gateway plugin, not an EHR rewrite): structured tokens verified against the AS's JWKS (audience, expiry, access rights) or opaque tokens introspected per RFC 9767. Recommended posture for patient-scale ASes: **structured + short TTL** — no runtime dependency on the AS's uptime; TTL bounds revocation lag.

3. **Proof-of-possession** — verify the client's HTTP Message Signature (RFC 9421) against the token's bound key on every call. The genuinely new piece for hospital infrastructure (SMART tokens are bearer) and the piece that makes standing patient-granted access defensible. The bound key is the same kind of subject-key principal the shipped vouch record anchors (§4.1) and a UCAN chain terminates in.
4. **Access-rights → FHIR enforcement** — GNAP `access` entries map near-mechanically to SMART scope semantics pinned to one patient compartment. The RS enforces **compartment confinement from its own registration record regardless of token contents** (defense in depth against a buggy or compromised AS). MAIA's scope lattice projects onto FHIR types (meds-allergies → MedicationRequest + AllergyIntolerance; not-sensitive is bounded by FHIR's own weak sensitivity labeling — flagged honestly).
5. **Audit + revocation** — AS-token accesses logged as a distinct class; revocation via short TTL or introspection. Both ends keep independent audit trails (the AS's request log + the RS's access log) — stronger accountability than portal-click consent.

What the hospital does NOT need: MAIA software; any group/registry awareness (this boundary is pure AS↔RS); a policy engine (the AS decides, the RS enforces token contents); UCAN validation — per the projection principle (§3.2), the AS flattens any internal delegation chain into a conventional structured token at this seam. Hospitals consume boring JWTs; the chain stays on MAIA's side.

Regulatory rails: the information-blocking rule and the right of access make refusing patient-directed automation progressively harder to defend; TEFCA IAS normalizes individual-mediated exchange; §170.315(g)(10) certification already mandates the API Mode 1 rides on. The standards for Mode 2 are complete (RFC 9635, RFC 9767, RFC 9421, FHIR) — what remains per hospital is governance, which is why Mode 1 exists as the unilateral wedge.

15.6 Vocabulary evolution: versioned cards, group namespaces, and Cedar

The problem. A stored card references vocabulary values (`signature: "group-member"`) whose *meaning* — rank in the identity ladder, position in the scope lattice — lives in the current build's tables. When the vocabulary evolves, stored cards can silently change meaning: removing a level makes its rank lookup undefined and a strict card **fails open** (the v1.5.173 `npi` removal surfaced exactly this — see §12.2-8 for the mitigation); adding a level inside an ordered ladder silently enters every old card's "or stronger"; editing the scope lattice changes what an old "everything not sensitive" card releases; splitting a purpose leaves no record of which meaning the patient consented to. Today cards are few and easily re-authored. As the matrix expands and groups differentiate, vocabulary evolution becomes a recurring **consent** event, not a code event.

The governing rule. *A card never silently changes meaning. Every vocabulary change either proves it preserves each affected card's semantics, or the card comes back to the patient — and until reviewed, it fails closed to ask.* This is the standing "deterministic evaluator + explicit consent decide" invariant extended over time, with the safe failure direction built in: an unmigratable card is disabled, and everything it covered escalates as questions. Nothing leaks; the worst case is extra asks.

Shipped now (P0, v1.5.174): the vocabulary is an *edition* — `POLICY_VOCAB_VERSION`, identical client and server (parity-tested) — and every card saved through `normalizeCard` stamps (a) the edition its elements are expressed in (absent stamp = edition 1; a card still using `npi` is stamped 1, the only edition that expresses it) and (b) its `authoredSentence`, the exact consent language rendered at save time. `Documentation/Policy_Vocab_Changelog.md` records every edition with each change classified **compatible** / **narrowing** / **consent-affecting** and a stated per-card migration rule. Stamping is deliberately first: no card authored from here on is unmigratable.

Planned (with the Cedar phase, §15.3): Cedar policies validate against a *schema*, and the schema is the natural carrier of editions. Cards stay canonical; each compiles against its authored edition; a vocabulary bump becomes a recompilation pass that either **proves equivalence** (or strict narrowing) and re-stamps silently, or queues the card in the Sharing Policies tab — old sentence beside new, keep / edit / retire — reusing the existing confirm-a-card UI. The

twin-evaluator collapse and the migration gate are one construction project, which is why versioning ships years of cards *before* Cedar rather than after.

Group namespaces (as groups differentiate). One small, curated **core** vocabulary (the cross-deployment contract) plus per-group **extensions** — scopes and purposes only, namespaced `grp:<groupId>:<value>` and published as part of the group's signed public info, so any member's AS on any host can evaluate them. The parameterized Apple Health category cell is the in-repo precedent. Three deliberate rules: (1) **signature ladders are core-only** — groups extend *what* can be asked for, never *who counts as verified*, keeping the rank lattice small and centrally versioned; (2) **cross-namespace subsumption defaults to none** — a core "everything" card does not silently absorb a scope a group invents later; the group's suggested cards request it explicitly; (3) a patient in several groups sees core + their groups' sections in the one matrix, and namespacing makes collisions impossible.

Appendix A. Numbered invariants

Testable claims. Suites that pin them are named in §14.1.

- **I-1** Evaluation order is deny → explicit ask → allow → default ask; a deny card always wins; the uncovered case always escalates to the human.
- **I-2** Client and server evaluators are semantically identical (parity suite).
- **I-3** Only privacy-filtered artifacts leave autonomously; absent a filtered artifact, allow degrades to ask — never to sharing raw or empty content.
- **I-4** Requester-visible outputs never disclose which policy card decided, in any outcome.
- **I-5** Silent deny is observationally indistinguishable from absence to the requester.
- **I-6** No identity claim evaluates above its actual proof (Doximity claims score as unverified today; removing a signature level from the vocabulary never weakens a legacy card that required it).
- **I-7** Members cannot forge outsider-attributed traffic; only the registry writes `outsider:` envelopes.
- **I-8** Member-to-member content is end-to-end sealed; the registry relays ciphertext only.
- **I-9** Member emails do not exist at the registry after join; invite tokens are stored only as hashes.
- **I-10** Ingest is idempotent: re-delivery can neither duplicate a request nor reset a decided one.
- **I-11** A delivered message is never reported as failed.
- **I-12** Payments move before delivery; every escrow settlement and purchase grant is idempotent; one payment covers one request; a card requiring payment matches only attested-paid requests; `payment: none` cards match everything.
- **I-13** MAIA never handles card data; unsigned or unconfigured webhook calls can never grant credits.
- **I-14** AI output never grants access: autonomous decisions are AI-free, and advisor proposals reach storage only through deterministic validation plus explicit user confirmation.
- **I-15** A verified Patient Summary is never overwritten without explicit confirmation; verification stamps are set only by human Verify and cleared by any edit.
- **I-16** Group-suggested policies bind only after adoption onto the patient's record and remain patient-editable; the evaluator reads only the patient's own cards.
- **I-17** Autonomous responses always carry machine attribution.
- **I-18** Credentials and capabilities are audience-bound to a key, never bearer — already true of the shipped vouch record, and required of every future delegation artifact.
- **I-19** A vouch elevates the signature to `verified-by-me` only in the envelope sealed to the vouching member; every other recipient of the same request sees at most the base level.

- **I-20** Revocation beats possession: every send re-reads the vouch record, so a revoked credential fails even with an unexpired session token, and later assertions are refused.
- **I-21** The registry never stores a vouch code (only its SHA-256, as the document id) nor the patient's label for the vouched person.
- **I-22** A stronger identity buys automation, never more data: every autonomous release is privacy-filtered regardless of signature level.
- **I-23** Every card records the vocabulary edition it was authored under and its consent sentence as rendered at save (absent stamp = edition 1); a vocabulary change either provably preserves a card's semantics or returns it to the patient, failing closed to ask in between.

Appendix B. Glossary

AS — the patient's Authorization Server (their MAIA host acting on their policies). **RS** — Resource Server: any holder of the patient's data that enforces the AS's tokens (§15.5: a hospital FHIR API). **RqP** — requesting party. **Registry** — the group admin's host: membership, relay, welcome page. **Card** — one plain-language policy statement over scope/purpose/signature/payment/action. **Envelope** — the sealed, structured request delivered per member. **Tally** — counts-only public record of an outside request. **Vouch** — patient-issued `verified-by-me` credential, bound to the requester's passkey (shipped v1.5.169). **Credits** — host-scoped prepaid service fees (100 = $2). **Projection** — a standards-format serialization of MAIA-canonical facts (Cedar, VC, UCAN, GNAP, JWT). **SMART on FHIR / §170.315(g)(10)** — the certified OAuth2 app-access surface every US hospital FHIR API must expose; Mode 1's on-ramp. **TEFCA IAS** — Individual Access Services under the US national exchange framework; the identity-proofing practice §15.5's registration step can ride.