POSTERN LABS

Postern
Technical Whitepaper

The Postern Technology Stack — architecture, primitives, and honest non-guarantees

> Reference prototype · **UNAUDITED** · for adversarial senior review.
> Every "planned / scaffold / stub / not-built / not-booted" marker in this document is deliberate and load-bearing. If a claim reads as more finished than the code, treat it as a documentation bug.

AUTHOR	Postern Labs
DATE	2026-07-12
AUDIENCE	Ultra-senior engineers & security auditors
STATUS	No external audit completed · contracted, not delivered

The Postern Technology Stack — Technical Whitepaper

Audience: principal / staff engineers evaluating the architecture adversarially. **Posture:** this document is written to be *checked*. Where a property is designed but not achieved, built but not booted, or implemented but not audited, it says so in the same sentence as the claim. A senior reader who finds an overclaim here should treat it as a documentation bug, not a feature.

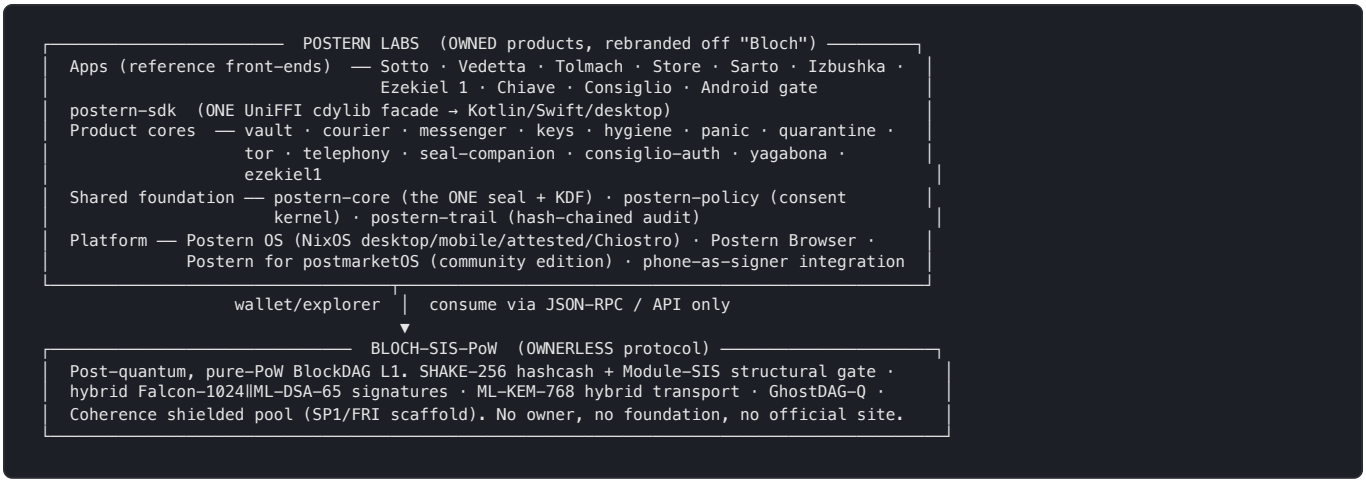
One-line summary. Postern Labs builds an *owned, permissively-licensed* privacy/security product suite (Rust cores + reference front-ends + a reproducible NixOS family) that sits on top of an *ownerless* post-quantum proof-of-work L1 it does not control. The product cores are built and test-verified; the OS images, cloud/enterprise editions, and mobile shells are designs and reference prototypes at varying maturity; **nothing in the stack has completed an external security audit**, and the base chain is a nascent, 51%-attackable, trivially-forgeable testnet whose coin is explicitly worthless.

Table of contents

---

1. Architecture overview — the two-layer model

The stack is deliberately partitioned into two ownership domains that meet only at a narrow, documented seam.



1.1 Why the split matters technically

- **The base is a git dependency, not a submodule of the product.** `bloch` / `bloch-crypto` are pulled from `gitlab.com/blochsispow-group/BlochSISPoW-project` and pinned in `Cargo.lock` (currently `rev = dd8df3526518ba792057708a6aa0efd5a91d8e40`). Only three product members actually need protocol crypto (`postern-seal-companion`, `mobile/core`, `desktop/src-tauri`); the rest of the suite compiles without touching the chain at all. This keeps the product attack surface disjoint from consensus code.
- **Most products never touch the chain.** The vault, courier, messenger, hygiene, panic, quarantine, tor, keys, and telephony cores are self-contained privacy primitives. The chain is relevant only to the *wallet* and the *wallet-as-login* identity story.
- **License hygiene is enforced mechanically.** The workspace is MIT/Apache-2.0 only; `cargo deny` (`deny.toml`) fails the build on copyleft, on external git deps other than the pinned protocol, and on RUSTSEC advisories. The PQ-crypto fork (`pqcrypto-internals`, seeded-RNG override) is vendored so the workspace is self-contained.

1.2 The four shared invariants

Everything above the base is held together by four "extracted-once" components, each existing specifically so two products cannot drift into incompatible behaviour:

Component	Guarantees one canonical…	Consumed by
`postern-core`	AEAD **seal** (XChaCha20-Poly1305, v2 key-committing) + **KDF** (Argon2id)	vault, courier, panic, telephony, pairing, seal-companion
`postern-policy`	**consent kernel** — fail-closed `Decision`, RFC-3339 `Expiry`, `EnforcementMode` honesty type	yagabona, ezekiel1, enterprise shield
`postern-trail`	append-only **SHAKE-256 hash-chained** audit log (canonical JSON)	yagabona, ezekiel1
`postern-sdk`	ONE UniFFI `cdylib` surface (one `setup_scaffolding!()`)	Android / iOS / desktop

The rest of this document is, in effect, a walk down from these invariants to the leaves.

---

2. The honest security architecture

This section answers "why is the OS secure?" the only honest way: by stating what each layer proves, and then the specific thing it does **not** prove. A senior reader should be able to reconstruct an accurate threat model from this section alone.

2.1 What "reproducible" means here — a strict claim ladder

The base repo's `REPRO.md` defines a claim ladder that the whole stack adheres to:

1. **"Not reproducible"** — before inputs are pinned (moving Nix channels, un-pinned upstreams).
2. **"Reproducible-by-design"** — after `flake.lock` / `Cargo.lock` are committed and inputs pinned. *This is where Postern OS is today*. Same inputs → same output *by construction* of Nix, but no independent party has yet produced a byte-identical rebuild.
3. **"Reproduced"** — earned *only* after `repro-compare.sh` shows a bit-for-bit identical narHash from **two independent builders** (diffoscope-clean). **Not yet achieved.** RocksDB 8.10 (C++) is flagged as the #1 nondeterminism risk.
4. **"Attested"** — earned *only* when a verity-sealed image boots on real SEV-SNP/TDX hardware. **Not yet achieved** — the confidential-VM path has been vector-tested but never run on a real CVM.

The load-bearing honesty rule, repeated across every surface: **a signature proves who published an artifact, not what it was built from; reproducible-by-design ≠ reproduced; built ≠ booted ≠ verified ≠ audited.**

2.2 The canonical seal — one thing to audit

Every at-rest secret in the suite is sealed by exactly one construction, in `postern-core::aead`:

- **Cipher:** XChaCha20-Poly1305 (24-byte nonce, 16-byte Poly1305 tag).
- **Wire format v2:** `0x02 ‖ nonce(24) ‖ commit(32) ‖ AEAD-ciphertext`.
- **Per-seal subkeys via HKDF-SHA-256** salted by the fresh nonce: `enc_key = HKDF(salt=nonce, ikm=key, info="postern-seal/v2/enc" ‖ context)` and `commit = HKDF(salt=nonce, ikm=key, info="postern-seal/v2/commit" ‖ context)`.
- **Key commitment:** the stored `commit` is checked in constant time (`subtle::ConstantTimeEq`) *before* decryption — excludes "invisible-salamander"/partitioning cross-key ciphertext substitution. This is the AWS-Encryption-SDK derive-and-store technique, chosen deliberately over bleeding-edge committing-AEAD modes.
- **Context binding:** a domain string (e.g. `b"postern-vault/v1"`) enters both HKDF `info` strings *and* the AEAD associated data, so a blob sealed for one product cannot be opened as another even under the same 32-byte key.
- **Zeroization is compile-time-checked:** a `const _` assertion fails the build if a dependency bump ever drops `ZeroizeOnDrop` on the cipher.

The KDF (`postern-core::kdf`) is Argon2id (v0x13) with **explicitly pinned** parameters in two tiers: **at-rest** `m = 64 MiB, t = 3, p = 1` (matches Bitwarden's default; RFC 9106 tier) and a **floor** `m = 19 MiB, t = 2, p = 1` (OWASP interactive minimum) for constrained targets. The parameters are part of the wire format — the same

passphrase derives a different key per tier.

Non-guarantee: symmetric crypto is not the post-quantum weak point (Grover only square-roots a 256-bit key), but the vault key is protected *only* by this software seal today. **TEE key-wrapping (StrongBox / Secure Enclave) and hybrid PQ key-wrapping (ML-KEM alongside the wallet's Falcon‖ML-DSA) are planned app-layer integrations for which no code exists in the vault.** Do not read "software keystore" as "hardware-backed."

2.3 The two attestation models — deliberately distinct

The stack has **two** attestation constructions that are complementary and must not be conflated:

1. **Device-signature attestation** (`postern-core::seal` + `postern-seal-companion`, default build). The transcript `DOMAIN("postern-seal/attestation/v1") ‖ os_roothash(32) ‖ len(measurement) ‖ measurement ‖ nonce(32)` is signed by the device's **hybrid Falcon-1024‖ML-DSA-65** key (the same verifier the node runs). The verifier **fails closed without a trust anchor**: a report checked only against its own embedded key authenticates nothing (the "F1 attack," which the test suite exercises). Trust root = a *pinned* public key, out of band.
2. **Hardware TEE attestation** (`postern-seal-companion` `sev-snp` feature; `tdx` module). For SEV-SNP, `virtee/sev` (pure-Rust, no OpenSSL — `crypto_nossl`) chains **VCEK → ASK → ARK** to a *pinned, vendored* AMD root (Milan/Genoa/Turin), verifies the VCEK signs the report body, then checks authenticated fields (measurement, host_data = L1 policy hash, report_data = freshness+channel binding, a component-wise TCB anti-rollback floor). **Intel TDX is an honest fail-closed stub** — `verify_td_quote` never returns `Genuine`.

What TEE attestation proves: the running rootfs is the exact immutable image (via the dm-verity `os_roothash`, which is the *TEE-independent* rung — even bare metal can prove it), and it ran inside a genuine SEV-SNP/TDX guest with the expected measurement.

What it does NOT prove (stated in the CVM MOTD and `ATTESTATION.md`): you are trusting the CPU vendor's TEE and its attestation CA chain, and the provider firmware baked into the launch measurement; side channels remain open research; the provider still sees traffic shape and controls availability; and attestation says nothing about a kernel 0-day or a coerced user. There is **no revocation check** in the SNP path (pinned 2020-era AMD keys — a revoked VCEK would still verify; CRLs must be refreshed out of band), and the whole path has **never been exercised against a quote from hardware Postern launched** — only offline `virtee/sev` fixtures.

2.4 The threat-model bottom line

Layer	Real guarantee today	Honest non-guarantee
Seal / KDF	XChaCha20-Poly1305 + Argon2id(64 MiB); key-committing, context-bound; zeroized	Software keystore; no TEE wrap; no PQ key-wrap shipped
Audit trail	SHAKE-256 hash-chained, tamper-**evident**	Tamper-**evident, not tamper-proof** — machine owner can rewrite the whole file
Policy/consent	Fail-closed `Decision`, RFC-3339 expiry, revocation	`EnforcementMode::ByConstruction` = **userland check, not a kernel/OS sandbox**
Nix OS image	Reproducible-**by-design**, hardened, dm-verity capable	**Not** reproduced by two builders; not built in the dev sandbox
Confidential VM (Chiostro)	Seal-gate handshake implemented + vector-tested	**Never run on real SEV-SNP/TDX**; TDX stubbed; no CRL check
Base chain	Post-quantum primitives, memory-safe Rust, zero `unsafe` in-tree	**k=4 trivially forgeable; 51%-attackable; single-operator; unaudited**

Governing rule (from `SECURITY.md`): no security or privacy property is claimed before its own external-audit gate clears. No "100% private," no "secure," ever, before an audit — and the audit is **contracted but not delivered.**

---

3. Shared foundation crates

All crates are `0.1.0`, edition 2021, `MIT OR Apache-2.0`, RustCrypto-only. A deliberate finding: **BLAKE3 is used nowhere**; the audit chain is SHAKE-256, channel binding is SHA-256, login transcripts are SHA3-256.

3.1 postern-core — the seal + KDF + attestation transcript

Covered in §2.2 and §2.3. Three modules: `aead` (the seal), `seal` (the attestation transcript byte string), `kdf` (Argon2id). Legacy pre-v2 bare `nonce‖ct` blobs remain **read-only** for migration; both paths are AEAD-authenticated so the version-byte fallback can fix a misparse but never admit a forgery.

3.2 postern-policy — the consent kernel (`#![forbid(unsafe_code)]`)

No cryptography — authorization logic extracted from yagabona and ezekiel1 so the "userland check, not a kernel boundary" caveat is **one shared type** rather than prose each product rounds up.

- `Decision { Allow, Deny{reason} }` — fail-closed; default/`silent()` = Deny; `and()` is a first-Deny-wins no-widening conjunction (enforces the "wheels-within-wheels" narrowing invariant).
- `Expiry` — RFC-3339/UTC, must be strictly future; compares by instant, not wall clock.
- `Revocation` — monotonic; `gate()` returns Deny ("never given") when revoked or expired.
- `EnforcementMode { KernelBoundary, ByConstruction, BestEffort }` — the anti-round-up honesty type. `honest_label()` returns pinned caveat strings (ByConstruction = *"integrity by construction: a userland, in-process check, not an OS sandbox or kernel boundary"*). Deliberately **not** `derive(Ord)` (declaration order would invert strength). Package + module docs: **"UNAUDITED draft."**

3.3 postern-trail — hash-chained audit (`#![forbid(unsafe_code)]`)

- **Hash:** SHAKE-256 (`Shake256` XOF, 32 bytes → 64 hex). `hash = SHAKE-256(prev_hash_bytes ‖ canonical_json)`.
- **Canonical JSON:** recursively key-sorted (`BTreeMap`), compact separators, timestamps-as-strings — RFC-8785/JCS in spirit, independent of serde_json feature flags. `SCHEMA_VERSION = 1` is part of the preimage; `hash` is excluded from its own preimage; `RESERVED_KEYS = [v, prev_hash, hash]` rejected in content (fail-closed).
- **API:** `Trail<C>` with `append()` that `fsync`s (`sync_all()`) before returning; `verify_chain()`; `verify_chain_to_head(expected)` catches truncation/tail-rollback that plain verification structurally cannot.
- **Honesty: "tamper-EVIDENT, not tamper-proof."** The machine owner can rewrite the entire file and recompute every hash — only a *partial* edit is detectable, and dropped tails are caught only with an out-of-band pinned head.

3.4 postern-sdk — the UniFFI facade

One `cdylib` with exactly one `setup_scaffolding!()`, folding in `bloch-mobile::WalletCore` (its own FFI compiled out via `default-features=false`), vault, courier, seal-companion, hygiene, panic, deniable, keys. A pure-grep CI gate (`mobile/ci/check-one-setup-scaffolding.sh`) asserts exactly one scaffolding — two in one cdylib collide on FFI symbols. **Secret-handling policy (load-bearing):** secrets are never fields of a `uniffi::Record` (Kotlin/Swift auto-`toString()` would log them); every secret lives behind an opaque `Arc<Object>` handle, Rust-side intermediates are `Zeroizing`, and mutex poisoning is *recovered* rather than panicked (a panic crossing the FFI during a `PanicSession` duress scenario would crash the host app at the worst possible time). Skipped in v0.1: tor, quarantine, messenger.

---

4. The product cores

4.1 postern-vault — sealed secrets store

`BTreeMap<String, Vec<u8>>` sealed with the canonical seal, context `b"postern-vault/v1"`, Argon2id at the 64-MiB at-rest tier. Custom length-prefixed serialization into one exactly-sized `Zeroizing<Vec>` (pre-computed size so a reallocation can't strand un-wiped secret fragments on the heap); all offset arithmetic is checked. Values (not names) zeroized on drop/remove/replace and on partial parse failure. **Non-guarantee:** TEE and PQ key-wrapping are planned, not shipped (§2.2).

4.2 postern-courier — hybrid PQ file/message drop

Today this is **only the crypto seal** — no transport code exists; a `SealedPayload` is never transmitted.

- **Hybrid KEM:** ML-KEM-1024 (FIPS-203 final, RustCrypto `ml-kem` 0.2) ‖ X25519, X-Wing-style combiner.
- **Combiner:** `HKDF-SHA-256(ikm = ss_MLKEM ‖ ss_X25519, info = "postern-courier/hybrid" ‖ WIRE_VERSION ‖ SHA256(kem_ct) ‖ SHA256(recipient_public))` → 32-byte key → XChaCha20-Poly1305 via `postern-core`.

- Hardening beyond the upstream crate: enforces FIPS-203 §7.2 encapsulation-key canonicality (byte-exact re-encode round-trip, which `ml-kem` 0.2 skips), rejects non-contributory (low-order) X25519 points, uses implicit-rejection decapsulation, zeroizes raw KEM secrets before any `?`.
- **Trust model / honesty:** no sender authentication — `open` succeeding proves nothing about who sealed (PQ sender-signatures planned, not implemented); key authenticity is out-of-band via an 80-bit `key_fingerprint`; payload length leaks; **Tor onion transport is NOT YET IMPLEMENTED**; `ml-kem` is pre-1.0 and unaudited — "exactly why the classical X25519 leg exists."

4.3 postern-messenger — Megolm messaging core (UNAUDITED)

Data model = `SecurityTier { InternalE2ee, BridgedExternal }` enforcing a fail-closed no-silent-downgrade wall + mandatory disclosure copy for bridged conversations. Crypto = **Megolm via vodozemac 0.10** (Matrix, Apache-2.0, Least-Authority-audited 2022); optional live path via `matrix-rust-sdk 0.18` (Olm per-device room-key distribution). A `megolm_reference` module is a legacy hand-rolled group session (`SessionConfig::version_1`, in-memory replay `HashSet`), explicitly "legacy/reference only — no new callers." Deliberately **never** Element (AGPL) or libsignal (AGPL). **Honesty:** Megolm is classical Curve25519 — *not* post-quantum (harvest-now-decrypt-later on transport); the homeserver sees metadata and the social graph; no persistence yet; no Tor wiring; no cross-signing/SAS (Phase 2).

4.4 postern-keys — passkey / FIDO2 core

WebAuthn **ES256 = ECDSA over NIST P-256** (`p256` crate), one keypair per relying party, random 16-byte `credential_id`, signs `authenticator_data ‖ client_data_hash`, DER signatures. `to_vault_record()` exports the private key **only to be sealed in postern-vault**. Compile-time `ZeroizeOnDrop` assertion. **Honesty:** classical ES256, **no PQ passkeys** (relying parties don't accept them); most sites ignore attestation; *"a software authenticator is weaker than a hardware key unless the key lives in the TEE"* — and the vault's TEE wrapping is itself unshipped. Full CTAP2/WebAuthn is left to the app layer.

4.5 postern-hygiene — send-side metadata scrubber

Not a crypto crate. Scrubs OOXML/ODF (`zip`), PDF (`lopdf`), JPEG (drops APP1–APP15 + COM, truncates post-EOI), PNG (drops text/eXIf/tIME chunks). Zip-bomb guards (≤8192 entries, ≤64 MiB/entry, ≤256 MiB total). Fails closed on malformed input. Opt-in provenance: `digest_shake256()` (SHAKE-256) and a `seal()` whose signature is a caller-supplied hybrid Falcon‖ML-DSA signature (closure keeps keys out of the crate). **Honesty:** the provenance seal is opt-in *because it can de-anonymize*; embedded non-JPEG/PNG media, OLE `.doc/.xls`, and PDF `/EmbeddedFiles` metadata are out of scope (reported in `passed_through`). No mat2 (LGPL) / ExifTool (GPL).

4.6 postern-quarantine — receive-side untrusted-file neutralizer

`quarantine_image()` decodes an untrusted image to raw pixels (`image` crate, PNG/JPEG only) and re-encodes clean PNG — structure/metadata/payload never survive the pixel round-trip. Decode-bomb guards (≤64 MiB input, ≤16384 px/axis, ≤512 MiB decode alloc). **Honesty (critical): the decode runs in-process, unsandboxed** — the decoder *is* the attack surface; pure Rust removes the C memory-unsafety class but not logic bugs; no wall-clock timeout yet; the PDF/Office path (PDFium + LibreOffice-headless + Landlock/seccomp) is design-only. "Do not describe this as exploit-proof." Concept from Dangerzone (AGPL), reimplemented with no code taken.

4.7 postern-tor — in-process Tor (Arti)

Wraps `arti-client` (no external tor daemon / Orbot / SOCKS). `PosternTor` with `bootstrap()`, per-identity circuit isolation (`isolated_client()` / `IsolationToken`), and a hand-rolled HTTP/1.1 `json_rpc()` (CRLF-injection validation, 60 s timeout, 2 MiB capped read, rejects non-200 and chunked). **Honesty: onion-service *client* only — hosting is NOT YET IMPLEMENTED** (this is what blocks the courier drop); live bootstrap is compile-verified only (CI test `#[ignore]`d); distinct identities *must* use isolated circuits or timing/exit correlation links them; on clearnet HTTP the exit relay is an active MITM (anonymity preserved, integrity not).

4.8 postern-telephony — carrier SMS/voice for Postern OS Mobile

A fresh MIT/Apache **zbus D-Bus client** to GPL system daemons (ModemManager, mmsd-tng, callaudiod) over IPC — no fork/embed. `SealedTelephonyStore` uses the canonical seal (XChaCha20-Poly1305 + Argon2id from the device vault) as a length-prefixed encrypted record log; plaintext types are `zeroize`-derived. **Honesty:** carrier SMS/voice is **NOT private on the wire** (plaintext to carrier, lawful intercept, SS7, tower knows location/IMEI/IMSI); the value is a hardened, encrypted-at-rest, no-telemetry local client that steers sensitive traffic to the E2EE messenger. MMS is a Phase-2 placeholder; an honest `Unimplemented` error variant exists.

4.9 postern-seal-companion — the attestation verifier

The user-facing "trust but verify" tool. Default build does device-signature peer-verify (§2.3), fails closed without a pinned anchor. The `sev-snp` feature adds the AMD SEV-SNP chain (VCEK→ASK→ARK to a pinned root, `virtee/sev`, p384 ECDSA + RSA-PSS). `binding.rs` computes the channel binding `report_data = SHA-256(nonce ‖ guest_pubkey) ‖ 0×32`. `gate.rs` is the transport-agnostic wire codec (`PSG1` challenge / `PSE1` evidence, hard length caps enforced before allocation). `tdx.rs` is the honest never-`Genuine` stub. Ships `seal_gate` + `seal_verify` binaries. **Honesty:** no revocation check; SNP never run on self-launched hardware; the `Cargo.toml` dep-pin comment still narrates an `<A_MERGED_SHA>` placeholder even though a concrete `rev` is now present — a human should confirm the pinned SHA is the intended canonical consensus-bundle commit (it must match `mobile/core` and `postern-consiglio-auth`).

4.10 postern-consiglio-auth — enterprise wallet-as-login (UNAUDITED)

Passwordless org auth where the Postern PQ wallet *is* the identity. Signature verification reuses the node's **hybrid Falcon-1024‖ML-DSA-65** (`bloch_crypto::crypto`) — no bespoke crypto. Login transcript = domain-separated SHA3-256 (`"consiglio:wallet-login:v1" ‖ len‖org ‖ len‖user ‖ nonce(32) ‖ issued_at`), 120 s TTL. `verify_response` is a 5-step fail-closed order (enrolled → pubkey-binding → freshness → single-use nonce → PQ verify); a bad signature does **not** burn the nonce. RBAC over 5 Consiglio roles. **Honesty ledger:** REAL = PQ verify, SHA3-256 transcript, anti-replay, pubkey→user binding, RBAC gate; **stubbed** = enrollment store (in-memory), nonce store (no TTL sweep), admin dual-control, Brewer-Nash SOD walls (declared, not runtime-enforced). "A compiling MVP scaffold, NOT a hardened IAM."

4.11 yagabona — privacy-first scheduled-task agent daemon (UNAUDITED)

(Trademark note: ships as "Yagabona," never "Baba Yaga.") A local-only, zero-telemetry task daemon. A task runs only if the user declared an explicit scoped, expiring capability grant in an **Izbushka TOML manifest**. Pipeline: `manifest` parse → `threshold::validate` (fail-closed; canonicalizes every path inside `workspace_root`, rejects `..`/symlink escapes; network must be `None` or `api.anthropic.com`; expiry via the shared

`postern_policy::Expiry`; heuristic inline-secret scan) → `ScopedFs` runtime path check → `postern-trail` SHAKE-256 audit → scheduler tick loop. `Task` is a **trait, never exec of arbitrary commands** (default `NoopTask`). `#![forbid(unsafe_code)]`. **Honesty:** `fsboundary` is a **userland boundary, not an OS sandbox** (`EnforcementMode::ByConstruction`) — does not stop raw syscalls or TOCTOU symlink races; Landlock/seccomp is unimplemented follow-up.

4.12 ezekiel1 — access-layer governance engine (UNAUDITED)

Decides "who may reach what," per access, default-deny. Two editions gated first: **Personal** (parental control only, parent→child, no tenants/walls) and **Business** (full lattice + Chinese walls + tenant isolation). Ezekiel *declares* (emits Izbushka manifests); Yagabona *runs* them. Mechanisms: strict TOML (`deny_unknown_fields`; a grant without `expires` fails to parse); a "wheels-within-wheels" no-widening lattice forest with **Bell-LaPadula** read/write rules; **Brewer-Nash Chinese walls** (first touch binds a COI class permanently); "firmament" tenant isolation; a 6-step fail-closed evaluator; same shared SHAKE-256 trail (genesis anchors a `policy_hash`). Re-exports `postern_policy::Decision` so it and yagabona cannot drift on "silence = deny." **Honesty:** firmament isolation is enforced **by construction inside a single process** — same address space/allocator/CPU, **not** an OS/kernel/hardware boundary; true per-tenant isolation is the deployment's job.

---

5. The applications and the browser

The `apps/*` are **self-contained reference front-ends** — almost all single-file, dependency-free, strict-CSP HTML with the inline script pinned by a CSP sha256. They share one deliberate honesty pattern: **browser primitives stand in for the product's real primitives, the *flow* is real while the *primitive* differs, and each app carries a stand-in-vs-product table.** Typical stand-in mapping: browser Ed25519/ECDSA-P256 → product hybrid Falcon-1024‖ML-DSA-65 in a device TEE; PBKDF2-SHA-256 → Argon2id; SHA-256 domain tags → SHA3-256; demo wordlists → BIP39-2048. None claims "100% private."

- **Sotto** — privacy-first voice assistant (Web Speech API STT — audio leaves to the browser vendor, disclosed; `speechSynthesis` TTS local; BYO Anthropic key, only dest `api.anthropic.com`). At-rest key lock: AES-256-GCM under PBKDF2-SHA-256 (310k iters). Emits the Sotto→Izbushka intent.
- **Vedetta** — a "quiet markets lookout" (Sotto's sibling): watchlist/alerts/holdings, voice+type. Adds market-data APIs to its CSP `connect-src`; seals both Anthropic and Finnhub keys; holdings stay in `localStorage`, "never leave the device."
- **Tolmach** — private interpreter + meeting minutes. Sessions encrypted at rest in IndexedDB with AES-256-GCM, key via PBKDF2-SHA-256 (600k iters, versioned header for a future Argon2id migration). A "Seal minutes" (PQ-sign + anchor) button is **disabled/planned** — "nothing is anchored anywhere today."
- **Tolmach-subscription** — a reference metered-billing tier: a **zero-dependency Node.js** proxy (two tiers: *Sovereign* = BYO key browser→Anthropic; *Subscription* = browser→proxy→Anthropic on the operator key). Bearer tokens stored only as SHA-256; Stripe webhook verified via HMAC-SHA-256; meters token counts only. **Its README leads with the tradeoff:** the proxy sees plaintext prompts in memory ("policy, not proof"); Sovereign is the recommended tier and must never be removed; the identity provider is deliberately stubbed.

- **Postern Store** — natural-language app store: you state the need, an optional concierge parameterizes a *deterministic local search* over a signed catalog. In-page **Ed25519 via SubtleCrypto** verifies `catalog.json` (trust states UNSIGNED/VERIFIED/TAMPERED/UNVERIFIED). **Loud caveat:** the catalog is a **SCAFFOLD** — no apk built/published; live signing state is UNSIGNED (`CATALOG_PUBKEY_B64 = null`); the future "Depot" plans TUF + hybrid Falcon‖ML-DSA PQ signing.
- **Sarto** (`postern-onboard`) — onboarding configurator *and* the transparent parental/corporate policy desk. Signed policies (Ed25519, fingerprint-pinned admin key), passphrase-sealed blobs with a **KDF-downgrade clamp**, and a cryptographically-enforced **transparency floor** — a policy requesting covert (`monitoring="full"`) monitoring is *refused even if otherwise valid* ("covert monitoring cannot be signed"). "Shown = enforced" invariant proven by node tests.
- **Izbushka** — the task portal, "the only door into Yagabona." CSP `connect-src 'none'` — the portal opens no network; a task manifest *may declare* `api.anthropic.com` as a capability the daemon honors. The Sotto↔Izbushka bridge treats an intent as **DATA, never an action** (same-origin only; the literal ritual string must match; it only pre-fills a form the human confirms).
- **Ezekiel 1** — the governance console over `crates/ezekiel1`. `connect-src 'none'`. Honest boundary: "the Rust engine is the authority; this page only DECLARES and DISPLAYS" — Web Crypto lacks SHAKE-256, so in-browser trail verification is out of scope.
- **Chiave** (`postern-login`) — Sign-In With Postern Wallet (no email, no account, `connect-src 'none'`). The richest client: wallet sealed AES-256-GCM/PBKDF2-310k at rest; login derives a domain-separated `m/login` key (never touches `m/spend`); single-use, 2-min, origin-bound nonce challenge; signature-gated session. Tests pin nonce-single-use (burned even on failure — no retry oracle), verifier-clock expiry, origin binding, phishing-proxy rejection, KDF-downgrade clamp. Optional WebAuthn+PRF biometric ("a gate, not a key"); Panic Lock disables the biometric path (coerced-finger defense).
- **Consiglio** (`postern-enterprise`) — enterprise admin console: dual-control escrow (M-of-N quorum, not a super-admin), RBAC+ABAC, Brewer-Nash walls, provision/offboard ceremonies, tamper-evident audit chain. Includes **real Shamir 2-of-3 over GF(256)** for the recovery key (shares ECIES-sealed to officer keys — "not theater"), Ed25519/P-256 signing, ECDH ECIES. `console.html` is the thin admin surface over the `enterprise/features` layer. Loud: **NOT LEGAL ADVICE**; the audit chain "detects, does not prevent."
- **Postern Android** (`postern-android`) + **Android gate** (`postern-android-gate`) — the **non-attestable escape-hatch tier**: a POSIX-shell wrapper around a Waydroid LXC Android container plus a deny-by-default consent gate. Entering Android means data leaves the sealed base; **not covered by the Postern Seal** (the `os_roothash` says nothing about a mutable Waydroid image); Play Integrity STRONG fails by design. The gate normalizes any malformed/partial state to **full-deny + not-consented** and always labels the tier `"non-attestable"`. Full honesty header: **"UNTESTED — nothing here has ever booted an Android container"** (validated only by shellcheck + a mock trace).

5.1 Postern Browser — hardened Firefox ESR

A hardened *distribution* of Firefox ESR (Gecko), in the LibreWolf/Mullvad-Browser class — **not** Chromium, not a from-scratch engine, not a code fork. Tagline: "hardened, not anonymous · built on Firefox ESR · unaudited public beta." Hardening lives in two locked layers:

- **`user.js` (about:config spine):** `resistFingerprinting` + letterboxing + `fingerprintingProtection`; `contentblocking.category=strict`; `cookieBehavior=5` (Total Cookie Protection) + network-state

partitioning; `https_only_mode` (incl. private windows); WebRTC kept on but leak-safe (`ice.no_host`, `default_address_only`); prefetch/predictor/speculative/beacon/geo off; DRM (EME/Widevine) off; telemetry killed with dead endpoints as belt-and-suspenders; FxAccounts off; `xpinstall.signatures.required=true`. Deliberately does **not** set `network.trr.*` — DoH is owned *solely* by `policies.json` so pref/policy/wire cannot drift.

- **`policies.json` (enterprise, locked):** telemetry/studies/Pocket/form-history/FxA disabled; tracking protection locked on; **DoH locked to `https://dns.mullvad.net/dns-query` with `Fallback:false`** (fail-closed — a DoH outage hard-fails resolution rather than leaking to ISP DNS); extensions blocked by default except uBlock Origin.

Built via `wrapFirefox` on `firefox-esr-unwrapped` (policies read from `policies.json` via `fromJSON` so derivation and artifact can't drift). **Ship-blocker honesty:** `wrapFirefox` de-brands only externally — a *published, signed* release requires a full source rebuild (`--with-branding=browser/branding/unofficial`, stripped logos/strings, a branding-audit grep gate) because Mozilla's trademark policy forbids redistributing under the Firefox name. `nixpkgsPin` is currently `TODO(pin)`, so the build falls back to the ambient channel with a loud trace warning that it is not rebuild-verifiable. The build manifest carries enforced honesty fields `reproducible=false, attestable=false, audited=false` (the R2 publisher refuses a manifest omitting them). Firefox builds are not trivially reproducible — "do not claim they are."

---

6. The NixOS solutions — Postern OS family

The `os/` directory is validated, idiomatic Nix that (per its own README) **was not built in the dev sandbox** (no Nix on the host); it must be built with `nix build` / `nix flake check` on a Linux host. Nix gives "same input → same image" *by design*; there is no claim of an achieved bit-for-bit reproduction.

- **`package.nix` / `bloch-node.nix`:** `rustPlatform.buildRustPackage` for `bloch`, source `lib.cleanSource` (drops `.git` / `target`), `cargoLock.lockFile` for self-containment (vendored `pqcrypto-internals`, no network fetch); RocksDB linked from the system package rather than recompiled; `doCheck=false`. The `services.bloch` module ships the "L2 hardening spine" as a systemd sandbox: `NoNewPrivileges`, `ProtectSystem=strict`, `PrivateTmp/PrivateDevices`, `MemoryDenyWriteExecute`, a `SystemCallFilter` allowlist, `RestrictAddressFamilies`, `LockPersonality`, `UMask=0077`, `LimitCORE=0`, dedicated `bloch` user.
- **`desktop.nix` (daily-driver):** Wayland GNOME (GDM), telemetry packages stripped; LUKS + TPM2 machinery via systemd-in-initrd (the module enables the machinery; the disk is formatted at install time); firewall deny-inbound; Tor client (SOCKS `127.0.0.1:9050`); `systemd-resolved` with DNSSEC + DNS-over-TLS; AppArmor; hardening sysctls; coredumps off; node off by default on laptops. Imports the hardened Firefox seed.
- **`attested.nix` (immutable + measured):** read-only erofs rootfs sealed with **dm-verity** via `systemd-repart` (esp/root + paired root-verity hash partition), UKI + `systemd-boot`, TPM2, `/` mounted `ro`. The verity roothash is passed as `roothash=` on the kernel cmdline; the node reads it and reports it as `os_roothash` via `getattestation`. Mutable state on a separate writable partition.
- **`cloud.nix` — Chiostro (confidential-VM guest):** imports `attested.nix`; a headless CVM guest (AMD SEV-SNP primary, Intel TDX secondary) with `sev-guest` / `tdx_guest` modules exposing `/dev/sev-guest`.

Pre-attestation the firewall opens **only** TCP 16510 (seal-gate) + UDP 51820 (WireGuard); everything else binds `wg0`. **Seal-gate flow:** per connection reads a fresh 32-byte nonce, requests an SNP report with `report_data = SHA-256(nonce ‖ WG_guest_pub)`, replies `report ‖ VCEK ‖ WG_guest_pub`; the client runs `postern-seal-verify` and opens the WireGuard tunnel **only** after `Verdict::Trusted`. The WG identity is generated *in-guest* on first boot (TEE-random) on the encrypted `/persist` volume — the host sees ciphertext only. **Honesty:** written on a non-Nix host, not built there; per-provider firmware/measurement details are **unverified until run on actual SEV-SNP/TDX instances**; the handshake math and AMD-root chain *are* implemented and vector-tested in `postern-seal-companion`, but the `/dev/sev-guest` path has **never been validated on hardware**. No QEMU-vs-hardware claim is made — the honest state is "implemented + unit/vector-tested, never run on a real CVM."

- **`seal-gate.nix`:** builds `postern-seal-gate` + `postern-seal-verify` with `buildFeatures=["sev-snp"]`. The `bloch-crypto` git dep's `cargoLock.outputHashes` must be pinned on first `nix build` (currently a `lib.fakeHash` placeholder).
- **`mobile.nix` (Mobile NixOS):** a **wallet-only** phone profile (`services.bloch.enable = mkForce false` — phones can't do SIS PoW), ships the `bloch-wallet` CLI. Not built in the sandbox; Mobile-NixOS revision must be pinned.

Attestation layering (`ATTESTATION.md`): L1 reproducible build → `image_digest`; verity → `os_roothash`; L2 hardening → posture only; L3 TEE → `tee` / `measurement` / `hostdata`. `os_roothash` is the TEE-independent rung. No attestation claim is adopted until the whole chain is audited.

---

7. Postern for postmarketOS — community edition

This is the **explicitly non-attestable** mobile tier. Naming rule enforced in `HONESTY.md`: it is always **"Postern for postmarketOS — community edition," never "Postern OS."** Only Nix-built, verity-sealed Mobile-NixOS images may be called "Postern OS." A mandatory disclaimer string ships in the distribution: *"Postern software on postmarketOS. No reproducibility or attestation guarantee. The attestable images are 'Postern OS' (Mobile NixOS) only."*

The three apk packages (abuild):

1. `postern` — an empty meta-package (`depends="postern-wallet postern-apps tor postmarketos-ui-phosh"`) whose post-install enables the Alpine Tor OpenRC service (pmOS uses OpenRC, not systemd) and prints the honesty notice. Ships no Postern daemon/user.
2. `postern-wallet` — the offline PQ wallet CLI (`bloch-crypto` `wallet-cli` feature), built `--no-default-features` to prune node deps (rocksdb/libp2p/ tokio) — a *light* wallet, RocksDB intentionally not packaged. Musl-specific fix: 1 MiB thread stack (`-z stack-size=0x100000`) because Falcon's deep C stack SIGSEGVs on musl's default 128 KiB. **Software keystore (Argon2id)** — the PQ secret is decrypted into normal RAM to sign; no StrongBox/TEE.
3. `postern-apps` — the web-app suite as static assets (`noarch`), with a build-time guard that fails if any `index.html` references `http(s)://` (offline invariant) and a generated freedesktop launcher per app. (`tolmach-subscription` is excluded — it needs an operator proxy.)

Why non-attestable (the load-bearing distinction): Alpine/postmarketOS packages are not bit-for-bit reproducible as a distro property (pmOS's own reproducible-repo attempt was abandoned because Alpine itself wasn't reproducible); there is no Nix closure; **signing (abuild-sign RSA / minisign Ed25519) proves *who published*, not *what it was built from*.** Bootloaders are unlocked on essentially every pmOS device, so even hypothetical verity would be tamper-evidence, not device attestation. Two independent signing systems: System A (apk repo, RSA, `abuild-sign`, verified against `/etc/apk/keys/<name>`) and System B (disk image, minisign Ed25519).

Status nuance to flag: the APKBUILD banners and `HONESTY.md` (2026-07-10) say "nothing has ever been built or installed," but `PUBLISHED.md` (2026-07-11) supersedes that and states the three packages are now built/signed/published for aarch64 on a live Cloudflare R2 repo — while still carrying every caveat: **never booted on a device**, signed with an *alpha* build key (not a stable release key), non-attestable, unaudited, and the **bootable image is NOT built** ("three apks and an index, not an operating system").

---

8. The mobile↔desktop phone-as-signer integration

Three independently-buildable crates (`integration/pairing`, `integration/signer-protocol`, `integration/pairing-transport-lan`), each with its own empty `[workspace]` so they build and test in isolation, offline. **Trust model — deliberate asymmetry:** the **phone is the custodian** (holds the wallet key, verifies, strong device); the **desktop is a terminal** (weak, unattested, software-only, holds *nothing*). The private key never crosses the wire — this is enforced *structurally*, not by policy.

8.1 Pairing — QR + KEM-bound SAS

- Desktop generates an ephemeral **ML-KEM-1024 ‖ X25519** keypair and shows a QR anchor (`anchor_pubkey` = 1568 B ML-KEM ek ‖ 32 B X25519 = 1600 B). Phone scans off-glass, encapsulates → session key, derives a **6-digit SAS**; desktop decapsulates → same key; humans compare SAS; `confirm_sas(matched)` is **fail-closed** (no timeout auto-accept).
- **Channel binding:** `SHA-256(nonce ‖ anchor_pubkey)`. Session key = X-Wing-style `HKDF-SHA-256(ikm = ss_MLKEM ‖ ss_X25519, info = "postern-pairing/session/hybrid" ‖ version ‖ SHA256(kem_ct) ‖ SHA256(anchor_pubkey) ‖ nonce)`. SAS is a *separate* HKDF branch (showing the SAS leaks nothing about the traffic key). Under active MITM the two sides derive different keys → different SAS → the human aborts. **No PAKE** — the typed value is never key entropy.
- **Channel frames:** `counter(8,BE) ‖ nonce(24) ‖ XChaCha20-Poly1305_ct`, AAD binds a per-role direction label (anti-reflection) + strictly-increasing counter (anti-replay). Forward-secret (fresh ephemerals per pairing, never derived from the wallet seed). `PairingRecord` sealed at rest with the canonical v2 seal; `Resume` does a fresh forward-secret re-handshake so the human SAS is not repeated. 48 tests.

8.2 Sign-request protocol — structural key-denylist + WYSIWYS

- Digests are domain-separated **SHA3-256 over decoded fields** (JSON framing is not trust-bearing): `login_digest` under `"consiglio:wallet-login:v1"`, `transfer_digest` under `"postern:signer-protocol:v1"` — domain separation guarantees a login signature "cannot move funds."

- **Structural key-denylist:** the seed and the Falcon‖ML-DSA secret live in types with **no `Serialize`** and can never be a field of a wire message. A compile-time census (`_WIRE_CENSUS`) plus a runtime golden test fail the build if any key byte (raw/hex/serde-array) reaches a serialized message. Only signatures + public keys travel back.
- **WYSIWYS:** there is deliberately **no human-text field on the wire** — the phone decodes the payload itself and renders the prompt from the actual signed bytes, so a compromised desktop cannot narrate a lie; an undecodable payload fails closed. Three consequence tiers: `SilentReadOnly` (no key use), `SingleTap` (login, can't move funds), `ExplicitReview` (moves funds). A 6-hex confirmation fragment is shown on both screens. Anti-replay = three independent fail-closed checks (session binding + monotonic counter + single-use nonce); a rollback-resistant watermark MACs the last counter. 19 tests.

8.3 Implemented vs planned

Implemented + tested: full QR/SAS pairing with a MITM-diverges-SAS test; PQ-hybrid session + XChaCha20-Poly1305 frames; SignRequest/Approval + replay guard + WYSIWYS + tiers + grants + structural denylist; the *real* phone side (`WalletCoreSigner`, hybrid Falcon‖ML-DSA) whose login signature verifies with the node's `bloch_crypto::crypto::verify` (the `e2e-demo` drives all 10 steps over loopback ending in a real hybrid verify); canonical conformance vectors + drift guard. **Planned:** real network transport as a plug-in (a real LAN TCP+mDNS transport crate exists but pairing ships loopback-only by default; **Tor onion out** — hosting unimplemented; **BLE out** — widens MITM); end-to-end value-transfer signing (the Transfer shape is proven; tx decode lives in `mobile/signer::txwysiwys`); hardware attestation of either endpoint (there is none — only a software SHA-256 pin, i.e. tamper-evidence, not attestation); the UniFFI app surface is wired but the **mobile image is UNBOOTED**. **Honest defeat conditions:** a rooted/malware endpoint (ring-0 reads the RAM key and forges the approval UI); it is not a hardware wallet (the PQ secret is in RAM to sign, zeroized on drop); coercion is out of scope; `ml-kem` is pre-1.0 and unaudited (X25519 is the classical hedge). The whole thing is labeled a scaffold.

---

9. The ownerless base — Bloch-SIS-PoW

A post-quantum, pure-PoW BlockDAG L1 (`bloch` `0.1.0-genesis`). GhostDAG-Q / PHANTOM consensus, libp2p gossipsub, RocksDB storage, ASERT-Lattice difficulty (~30 s target). No BFT finality, no validator set, no treasury — finality is PoW depth, à la Bitcoin/Kaspa. The products *sit on* this substrate via JSON-RPC/API only; **revenue never comes from the coin.**

9.1 What "SIS-PoW" actually is (and is not)

Given a header `H_block` and nonce `ν`, the miner finds a short vector `s ∈ {-B,…,B}^N` with (1) `‖s‖_∞ ≤ B`, (2) `‖A·s − t‖_∞ < β` — a Module-SIS instance where `A` (512×256) and `t` (512) are SHAKE-256-expanded from the seed — and (3) `SHAKE256("BLOCH-POW-AUX-V1" ‖ s ‖ ν ‖ H_block) < target`. Shipped v0.1 params: `n=256, m=512, q = 2²³−2¹³+1 = 8,380,417` (the ML-DSA-65 prime), `B=2, β = q/16 = 523,776`.

The load-bearing honesty: the PoW's security is **hashcash cumulative work on the aux SHAKE-256 target — NOT lattice hardness.** Three independent analyses converge on the proof that a *trapdoorless* PoW cannot be simultaneously lattice-hard and mineable (the honest miner and the attacker face the same instance;

every ≥100-bit-secure point is unmineable and every mineable point is trivial q-ary — the regimes are disjoint). The Module-SIS residual is only a **structural gate** (`√k·β < q`, compile-time-enforced) binding work to a lattice form; **no lattice bit-security number attaches.** The post-quantum property comes solely from SHAKE-256 (Grover is only quadratic).

9.2 The k parameter — "trivially forgeable right now"

The residual bound (condition 2) is checked on a small `k` of the 512 rows. **The chain currently runs the relaxed `k=4` regime → work is trivially forgeable** (checked on a handful of coefficients). The `k=8` hardening was activated at **block 213,000** as a soft-fork then **reverted** — it multiplied difficulty ~4096× and the low/solo hashrate stalled the chain; it re-activates only *together with a matched difficulty reduction* (so block time stays ~30 s). Until then, **no security is claimed.** The outstanding research deliverable is the no-shortcut / attacker-asymmetry proof for the chosen k, plus an IACR ePrint — not a hardness reduction.

9.3 Primitives

Signatures = **hybrid Falcon-1024 ‖ ML-DSA-65** (both must verify — two independent lattice families), seed-deterministic. Transport = **ML-KEM-768 (Kyber) hybrid + ChaCha20-Poly1305** with hybrid PQ peer identity. Hashing = SHAKE-256 throughout the consensus path (no classical primitive). The lean crypto surface is extracted to `crates/bloch-crypto` so products depend on it without the node.

9.4 The Coherence shielded-tx scaffold (SP1/FRI)

`coherence-core` = notes / SHAKE-256 commitments / nullifiers / Merkle accumulator + `check_spend` (one source of truth shared by node, guest, mobile). `coherence-prover` = an **SP1 (Plonky3) zkVM scaffold**: a RISC-V guest runs `check_spend`; a host proves/verifies. **Post-quantum rule: use the raw `.core()` STARK/FRI proof and FRI verification — NEVER a Groth16/PLONK elliptic-curve wrap (that would be Shor-breakable).** The node verifies the FRI proof locally and trustlessly — never calls the prover service for consensus. **Status: scaffold only** — `check_spend` + `ShieldedState::validate` are implemented/unit-tested, but turning shielded tx *on* (deploy the prover, wire the node-side FRI verifier, replace the reject-all `verify=false` stub, and the U.4 live reorg wiring) is open. No "private" claim until the C3/C4 audit clears.

9.5 The coin, stated plainly

21,000,000,000 BLCH nominal, 100% miner emission, ~30 s blocks, with a **disclosed 17% founder premine (10-year cliff, 40-year monthly on-chain vesting, structurally passive)**. Per `PRINCIPLES.md`: **"The token is not an asset… no listing will be pursued… on the zero-security testnet it is worth nothing, by design."** Being ownerless is precisely *why* it is not a security — no promoter, no expectation of profit. The tokenomics exist only to make the network function.

9.6 The maintainers' own comparison to Bitcoin (from `SECURITY_SELF_ASSESSMENT.md`)

This is the single most useful honesty artifact in the base repo, deliberately structured to make the chain look bad where it is worse:

- **Age:** < 1 year vs Bitcoin's 16. **External audits:** 0 vs 1 (Quarkslab). **Implementations:** 1 (Rust) vs 5+.

- **Decentralization:** ~1 seed + ~5 workers, **all operated by the founder**, zero independent third-party nodes; ~1–40 MH/s. "Bloch-SIS is **not decentralized** — it is a single-operator network." **Cost of a 51% attack: dollars** (a laptop CPU, or one second-hand ASIC). This gap "dwarfs every technical advantage."
- **Where it is genuinely better:** post-quantum signatures, PQ transport confidentiality, memory-safe Rust (zero in-tree `unsafe`), smaller consensus-critical codebase.

(Note: that self-assessment is an older, rebranded document describing an SHA-256/GhostDAG configuration; the current canonical PoW is the SHAKE-256 hashcash + Module-SIS gate described in §9.1. Its decentralization/audit/51% framing is unchanged and current.)

---

10. Per-product reference table

Product	What it is	Core mechanism / primitive	Builds on	Trust boundary / status
postern-core	The one canonical seal + KDF	XChaCha20-Poly1305 v2 (key-committing, HKDF-SHA-256 subkeys, context+AAD bound); Argon2id 64 MiB/t3	—	Software crypto; foundation everything audits once
postern-policy	Consent kernel	Fail-closed `Decision`, RFC-3339 `Expiry`, `EnforcementMode` honesty type	—	UNAUDITED draft; userland, not a kernel boundary
postern-trail	Hash-chained audit	SHAKE-256 over canonical JSON, fsync-per-append	—	Tamper-EVIDENT, not tamper-proof; UNAUDITED
postern-sdk	UniFFI facade	One `cdylib`, one scaffolding; secrets behind opaque handles	vault/courier/seal/hygiene/panic/keys + wallet	v0.1; tor/quarantine/messenger deferred
postern-vault	Sealed secrets store	Canonical seal, ctx `postern-vault/v1`, Argon2id	core	TEE + PQ key-wrap **planned, not shipped**
postern-courier	PQ file/message drop	ML-KEM-1024‖X25519 (X-Wing HKDF) → XChaCha20-Poly1305	core	Seal only; **no transport**; no sender auth; ml-kem unaudited
postern-messenger	Megolm messaging	vodozemac 0.10 / matrix-rust-sdk; tier wall	— (never Element/libsignal)	UNAUDITED; classical (not PQ); metadata visible
postern-keys	Passkeys	ES256 / ECDSA-P-256	core (vault export)	Classical only; software authenticator weaker than HW
postern-hygiene	Metadata scrubber (send)	OOXML/PDF/JPEG/PNG scrub; SHAKE-256 provenance (opt-in)	—	Embedded media / OLE / PDF-embedded out of scope
postern-quarantine	Untrusted-file neutralizer (recv)	Pixel round-trip re-encode (PNG/JPEG)	—	**Decoder runs unsandboxed**; PDF/Office path design-only
postern-tor	In-process Tor	arti-client; circuit isolation	—	**Onion hosting NOT implemented**; client only
postern-telephony	Carrier SMS/voice	zbus→ModemManager; sealed store (core)	core	**Not private on the wire**; MMS Phase-2 stub
postern-seal-companion	Attestation verifier	Device Falcon‖ML-DSA verify; SEV-SNP VCEK→ASK→ARK	core, bloch-crypto	Fails closed w/o anchor; **never run on real HW**; TDX stub; no CRL
postern-consiglio-auth	Wallet-as-login	Falcon‖ML-DSA challenge; SHA3-256 transcript; RBAC	bloch-crypto	UNAUDITED; enrollment/nonce store/SOD stubbed
yagabona	Scheduled-task daemon	Izbushka manifest → threshold → ScopedFs → SHAKE-256 trail	trail, policy	UNAUDITED; userland boundary, not OS sandbox
ezekiel1	Access governance	No-widening lattice + Bell-LaPadula + Brewer-	trail, policy	UNAUDITED; firmament = in-process, not kernel

Product	What it is	Core mechanism / primitive	Builds on	Trust boundary / status
		Nash; SHAKE-256		isolation
Postern Browser	Hardened Firefox ESR	user.js + locked policies; DoH fail-closed to Mullvad	Nix wrapFirefox	`reproducible/attestable/audited = false`; trademark rebuild pending
Sotto/Vedetta/Tolmach	Voice/markets/interpreter	AES-256-GCM + PBKDF2 (310k/600k) at rest; BYO Claude key	—	Reference prototypes; classical stand-ins
Postern Store	NL app store	In-page Ed25519 catalog verify	—	Catalog is a SCAFFOLD; live state UNSIGNED
Sarto (onboard)	Configurator + policy desk	Ed25519-signed policies; transparency floor un-signable if covert	—	Reference prototype
Chiave (login)	Sign-In With Wallet	m/login key; single-use origin-bound nonce; WebAuthn+PRF gate	consiglio-auth	Zero-network reference; classical stand-ins
Consiglio (enterprise)	Admin console	Shamir 2-of-3 GF(256); Brewer-Nash; SHA-256 audit chain	consiglio-auth	Reference prototype; NOT LEGAL ADVICE
Enterprise licensing	Offline license lib	Ed25519 over canonical JSON; Vendor→Master→Sub chain	node:crypto	UNAUDITED; DUNS format-only ("honor system")
Enterprise cloud	Hosted offering	Ed25519 license svc; keyless relay (Megolm ciphertext)	—	**"Hosted is less private than self-hosting"**; in-process tenant namespacing
Postern Wallet (mobile)	Light client / wallet	Falcon‖ML-DSA signing; StrongBox/SecureEnclave *wrap+attest*	mobile/core, sdk	Core tested; **app shells unbuilt; image UNBOOTED**
Postern Desktop	Tauri node companion	Node control + wallet + vault + panic + signer-requester	bloch, WalletCore, tor, vault	Build-from-source only; CSP is a pin, not an audit
Bloch-SIS-PoW	Ownerless PoW L1	SHAKE-256 hashcash + Module-SIS gate; Falcon‖ML-DSA; ML-KEM-768	—	**k=4 forgeable; 51%-attackable; single-operator; unaudited**

11. Honest status & threat model

11.1 The five load-bearing non-claims

1. **Reproducible-by-design ≠ reproduced.** Nothing in the stack has a two-builder bit-for-bit reproduction. A matching hash from your own rebuild is meaningful, but until an *independent* party publishes a matching build it is not reproduction, and never an audit.

2. **Built ≠ booted.** The confidential-VM (Chiostro) path, the mobile OS images, the Android/Waydroid tier, and the mobile app shells have never been run on real hardware/devices. Vector tests and unit tests are not a booted system.
3. **QEMU / offline vectors ≠ hardware.** The SEV-SNP verifier has only been exercised against `virtee/sev` fixtures — never against a quote from hardware Postern launched. Intel TDX is a fail-closed stub.
4. **Software keystore ≠ hardware-backed.** Vault, wallet, and pmOS keys are Argon2id-protected software keystores. TEE key-wrapping and PQ key-wrapping are planned app-layer integrations with no shipped code. Mobile TEEs (StrongBox/Secure Enclave) are classical P-256 and only *wrap at rest / attest* — they do not hold the post-quantum key; PQ signing happens in RAM (zeroized on drop).
5. **Signature ≠ attestation.** Across pmOS, the browser, the store, and enterprise deployment, a signature proves *who published*, never *what was built from*. Only the Nix/verity/TEE path claims attestability, and that claim is itself audit-gated and hardware-unproven.

11.2 Audit status

Zero external security audits have been completed — on any product core, any OS image, or the base protocol. An audit is *contracted but not delivered*. Internal review, an in-house "Fable-5" adversarial pass, and passing test suites are not an audit. Several cores are explicitly labeled "UNAUDITED draft" (policy, trail, messenger, consiglio-auth, yagabona, ezekiel1); the integration and cloud/enterprise layers are labeled reference prototypes.

11.3 The base protocol's real threat model

- **PoW is trivially forgeable** in the current k=4 regime; k=8 is reverted pending a matched difficulty cut.
- **51%-attackable for the cost of a laptop or one used ASIC.** ~1 seed + ~5 workers, **all operated by the founder** — the network is not decentralized.
- **The coin is worthless and not a security** — no sale, no listing, no price, 17% premine disclosed. Do not attach value.
- The Coherence privacy layer is a scaffold (reject-all stub); no privacy claim until its audit gate.

11.4 What is genuinely solid today

To be fair to the reader in both directions: the `postern-core` seal is a careful, modern, key-committing/context-bound AEAD with pinned Argon2id and compile-time zeroization checks; the shared consent/audit crates encode their own honest limits as *types* rather than prose; the phone-as-signer's structural key-denylist (no-`Serialize` secrets + compile-time census + golden test) and WYSIWYS (prompt rendered from signed bytes, no wire text field) are strong, well-tested designs; the seal-companion correctly fails closed without a trust anchor and pins the AMD root out of band; the wallet core is byte-compatible with the node (golden test) and memory-safe (zero in-tree `unsafe`). The engineering discipline is real — the gap is *maturity, hardware validation, decentralization, and an external audit*, and the documentation is unusually candid about exactly that gap.

Appendix A — cryptographic primitive inventory

Concern	Exact primitive	Location
AEAD seal	XChaCha20-Poly1305, v2 key-committing, HKDF-SHA-256 subkeys, context+AAD bound	postern-core `aead`
Passphrase KDF	Argon2id v0x13, 64 MiB/t3/p1 (at-rest) or 19 MiB/t2/p1 (floor)	postern-core `kdf`
Attestation transcript	Domain-tagged, length-prefixed byte string (integrity, not secrecy)	postern-core `seal`
Audit chain hash	**SHAKE-256** over canonical JSON (not BLAKE3)	postern-trail
Channel binding	SHA-256(nonce ‖ pubkey) ‖ 0×32	seal-companion `binding`, pairing
Device attestation signature	Hybrid **Falcon-1024 ‖ ML-DSA-65** (via bloch-crypto)	seal-companion, wallet, consiglio-auth
Cloud TEE root	AMD SEV-SNP VCEK→ASK→ARK (p384 ECDSA + RSA-PSS, virtee/sev), pinned root	seal-companion `snp` (feature)
Intel TDX	Fail-closed stub — never `Genuine`	seal-companion `tdx`
Hybrid PQ KEM (courier / pairing)	**ML-KEM-1024 ‖ X25519**, X-Wing-style HKDF-SHA-256 combiner	courier, integration/pairing
Base-chain transport KEM	**ML-KEM-768 (Kyber) hybrid** + ChaCha20-Poly1305	bloch node transport
Base-chain PoW	SHAKE-256 hashcash + Module-SIS structural gate (n256/m512/q8380417/B2/β=q16, k=4 today)	bloch-sis-pow
Shielded proofs	SP1 / Plonky3 **raw FRI/STARK** (never Groth16/PLONK)	coherence-prover (scaffold)
Passkeys	ES256 / ECDSA-P-256 (classical only)	postern-keys
Login transcript	Domain-separated **SHA3-256**	consiglio-auth, signer-protocol, Chiave
Enterprise licensing	Ed25519 over canonical JSON	enterprise/licensing
Enterprise recovery	Shamir 2-of-3 over GF(256) + ECDH ECIES	Consiglio console
Browser DoH	Locked to Mullvad, fail-closed (no ISP fallback)	Postern Browser policies.json

---

Document status: technical description derived from the source at `bloch-protocol`, `bp-wt-browser`, and `bloch-blockchain` as of 2026-07. Every "planned," "scaffold," "stub," "UNAUDITED," and "not built/booted" marker in this document is transcribed from the code, comments, or repo docs — not softened. If any statement here reads as more finished than the code, treat it as an error to correct against the source.