SECURITY & CLAIMS AUDIT · REMEDIATION REVIEW

Aphanite v1
Security & Claims Audit — v1.1

A source-level security assessment of the Aphanite protocol, application, and smart contract, with a line-by-line verification of the public claims at aphanite.io — updated to confirm the remediation of findings from the initial review.

Remediation status of the 12 substantive findings from v1.0

9 fixed · 1 partially addressed · 2 open (by design / low priority)

All three High-severity findings verified fixed in code.

Subject
Aphanite v1 — aphanite.io

Scope
Cryptography · Backend · Smart contract · Claims

Method
Manual source re-review · fix verification · live on-chain check

Report
July 25, 2026 · Version 1.1 (re-audit)

A point-in-time review; not a substitute for a formal third-party audit or machine-checked verification — a step the Aphanite specification itself recommends and which this document again endorses.

Contents

1. Remediation summary (what changed since v1.0)
2. Executive summary
3. Scope, method, and limitations
4. System overview and trust model
5. Claims verification matrix (updated)
6. Findings & remediation status
7. Cryptography review — X3DH & Double Ratchet
8. Smart contract review — AnchorLog
9. What Aphanite does well
10. Remaining work

A. Appendix — severity model, ciphersuite, on-chain data

1 · Remediation summary

Following the initial review (v1.0), the Aphanite team applied fixes across the codebase. This version re-reads the current source, verifies each fix against the actual code, and re-checks the on-chain state. The headline result: **every High-severity finding is resolved**, and the great majority of Medium and Low items with it. The two items that remain open are a smart-contract hardening item that requires a redeploy and a low-severity storage quota — both reasonable to defer, and neither affects message confidentiality or custody of funds.

ID	Severity	Status	Finding & fix
H-1	HIGH	FIXED	SIWE now enforces a domain allowlist and passes `domain` + `time` into verification.
H-2	HIGH	FIXED	Session-signing secret fails closed in production (throws if unset / dev default).
H-3	HIGH	FIXED	X3DH is bound to the transparency-anchored device key; a swapped bundle can no longer establish a readable session.
M-1	MEDIUM	FIXED	New key-vault encrypts all private key material at rest under a non-extractable WebCrypto key.
M-2	MEDIUM	FIXED	Auto-backup passphrase now wrapped by the same non-extractable key; legacy plaintext migrated and purged.
M-3	MEDIUM	PARTIAL	Marketing now scopes forward secrecy to “direct messages”; a group ratchet remains a documented roadmap item.
M-4	MEDIUM	OPEN	AnchorLog consistency proofs / multi-publisher / regular checkpointing — requires a contract redeploy and cadence; deferred.

L-1	LOW	FIXED	Payment-verify bypass now hard-fails in production.
L-2	LOW	FIXED	Admin authority bound to an address allowlist, not a claimable handle.
L-3	LOW	FIXED	Anonymous rate-limit re-keyed via a trusted client-IP helper.
L-4	LOW	OPEN	Backup/blob storage still lacks a size cap — low-priority DoS hardening.
L-5	LOW	FIXED	QR link-login now shows the requesting device's fingerprint before approval.
I-2 / I-3	INFO	FIXED	FAQ and privacy policy now explicitly caveat metadata and scope the "cannot read" / freeze claims.

Remaining informational items (I-1 unencrypted ratchet headers, I-4 8-character backup passphrase minimum, I-5 proof-of-sealing heuristic) are unchanged and remain low-priority hardening suggestions.

2 · Executive summary

Aphanite is a wallet-native, end-to-end encrypted messenger that lets people send funds wallet-to-wallet inside an encrypted conversation. Its promises — *Unseizable, Unseen, Unspoofable* — reduce to concrete technical claims: the server can never read message content, funds are self-custodied and cannot be frozen, and identities are verifiable through an on-chain key-transparency log. This review tested those claims against the actual source, the deployed smart contract, and the published whitepapers, and — in this version — confirmed the remediation of the issues found the first time.

Overall assessment: the core claims are supported by real engineering, and the account-layer weaknesses identified in the first pass have been closed. Message content is encrypted on the client with standard audited primitives; X3DH and the Double Ratchet are implemented correctly for direct messages; the relay stores only ciphertext and rejects plaintext; and the on-chain key transparency is genuinely verified in-browser against Ethereum. With the current fixes, the three items that most affected the security posture — domain-bound login, fail-closed session signing, and a handshake anchored to the transparency-logged identity key — are resolved, and private key material is now encrypted at rest.

Headline verdict (v1.1)

Content confidentiality: **Supported.** Client-side E2EE is real; the server routes ciphertext it cannot decrypt.

Metadata privacy: **Supported with caveats — now disclosed.** The FAQ and privacy policy state plainly what routing metadata the relay sees.

Account security: **Materially improved.** Login is domain-bound, sessions fail closed, and the handshake is anchored to the transparency log.

Self-custody of funds: **Supported.** The server never holds keys or funds.

On-chain key transparency: **Supported with caveats.** Genuinely verified in-browser; still bounded by a single publisher and a sparse checkpoint history (M-4, open).

Findings posture

Severity	v1.0	Fixed	Remaining

CRITICAL	0	—	0
HIGH	3	3	0
MEDIUM	4	2 (+1 partial)	1 (M-4)
LOW	5	4	1 (L-4)
INFO	5	2	3 (minor)

3 · Scope, method, and limitations

3.1 What was reviewed

Both the original and this follow-up covered the Aphanite v1 application repository (a single Next.js/TypeScript codebase, internally named `sealed`): the client cryptography (`src/lib/crypto/`), the server and API routes (`src/server/`, `src/app/api/`), the sole smart contract (`contracts/AnchorLog.sol`) with live on-chain verification, and every public claim surface (landing page, in-app whitepaper/protocol pages, FAQ, privacy policy, and the three downloadable PDFs).

3.2 Method for this re-audit

Each prior finding was re-checked by reading the current source of the specific file(s) involved and confirming the fix does what it claims — for example, that the SIWE domain is actually enforced before verification, that the session secret genuinely throws in production, and that the handshake uses the anchored device key rather than the server-asserted bundle field. The smart contract's on-chain state was re-read from a public Ethereum RPC.

3.3 Limitations

> This remains a **point-in-time source review**. It is not a formal third-party audit, machine-checked proof, live penetration test, or dependency/supply-chain audit, and it does not exhaustively cover the two largest UI files. Verifying that a fix is present and correct in source is not the same as proving the absence of all related issues; the remaining assurance step is an independent audit, which this document continues to recommend before high-risk use.

4 · System overview and trust model

Aphanite ships as one Next.js application. The client holds all keys and performs all encryption; the server is a "blind" directory-and-relay that stores and forwards ciphertext envelopes plus the account, handle, and transparency records needed to route them. There is no phone number and no email — the account root is a wallet, and login is a wallet signature (SIWE / EIP-4361).

Layer	What it does	Operator trust (post-fix)
Messaging		

	1:1 uses X3DH → Double Ratchet (`dr-v1`); groups and rich payloads use X25519 sealed boxes (`box-v1`).	Low for content. Key distribution is now anchored to the transparency log (H-3 fixed).
Identity	Wallet is the root; `@handle` → device-key bindings enter an append-only Merkle log.	Directory anchored on-chain and client-verified (M-4 caveats).
Relay / storage	A JSON-backed store holds ciphertext envelopes, handles, sessions, prekeys, and the transparency log; envelopes pruned after 7/30 days.	Can withhold or delete delivery/ accounts — cannot read content.
Payments	Transfers are on-chain transactions signed by the user's wallet; the server only verifies payments.	None for custody.
Anchor	`AnchorLog` on Ethereum mainnet stores checkpoints published by a single immutable operator wallet.	Single publisher (M-4, open).

> **The honest one-line trust model**
>
> Aphanite is a **single-operator service with client-side end-to-end encryption and on-chain transparency checkpoints.** A malicious or compelled operator cannot read message content and cannot touch funds; it can see communication metadata and deny service. This is a strong, defensible model — and, notably, the product now says so plainly in its own FAQ and privacy policy.

5 · Claims verification matrix (updated)

Verdict key: **Supported** · **Supported w/ caveats** · **Partially supported** · **Overstated**. Rows changed since v1.0 are marked **updated**.

5.1 Cryptography & confidentiality

Claim	Verdict	Basis
"X3DH + the Double Ratchet… so Aphanite can never read them."	**Supported w/ caveats**	DM crypto correct; content ciphertext-only. Now additionally anchored so an active operator can't silently swap keys (H-3).
"The server routes ciphertext it structurally cannot read."	**Supported**	Ciphertext-only envelopes; plaintext rejected server-side; keys never leave the device.
"We cannot read your messages… to anyone." (FAQ / Privacy) **updated**	**Supported (now scoped)**	The copy now explicitly adds: "this covers content, not metadata," and names the routing data the relay sees. This resolves the prior overstatement (I-2).
"Forward secrecy and post-compromise security." **updated**	**Supported (now scoped to DMs)**	Landing now reads "…on direct messages." Groups/ `box-v1` forward secrecy remains a documented roadmap limitation (M-3).

5.2 Custody & funds

Claim	Verdict	Basis
“Non-custodial by construction… no Aphanite in the flow of funds.”	**Supported**	Server verifies on-chain payments only; never holds keys or funds.
“No balance here anyone can hold or freeze.”	**Supported w/ caveats**	True for on-chain funds; account/message delivery is still operator-controllable. The site keeps this scoped to funds.

5.3 Identity, authenticity & transparency

Claim	Verdict	Basis
“Your browser proves who you’re paying against the blockchain.”	**Supported w/ caveats**	Client reads the checkpoint from a public RPC and compares roots locally. Entries newer than the last checkpoint are server-trusted until anchored (M-4).
“Proving control of the wallet is the login (SIWE)… a captured transcript cannot re-authenticate.” updated	**Supported**	Now accurate: the server enforces domain binding, single-use nonce, and validity window (H-1). Phishing-origin replay is closed.
“The directory can’t swap a key… without publicly provable evidence.” (FAQ)	**Partially supported**	Holds for the anchored prefix; the whitepaper’s full caveat (server-trusted tail, no in-band consistency proof) still applies (M-4).

5.4 Architecture & product

Claim	Verdict	Basis
“No phone number or email.”	**Supported**	Auth is purely wallet-signature; no PII fields.
Backups “encrypted with Argon2id + AES-256-GCM… unrecoverable without the passphrase.”	**Supported w/ caveats**	Scheme correct; the auto-backup passphrase is no longer stored in plaintext (M-2). Guarantee still bounded by passphrase strength (I-4).
Marketing avoids “military-grade / zero-knowledge / unbreakable / anonymous.”	**Confirmed — positive**	Still absent on every public surface.
A native token / airdrop.	**No such claim**	No token promised anywhere public.

6 · Findings & remediation status

H-1 · SIWE signature verified without binding the site domain

HIGH FIXED

Component: `src/server/auth.ts` · `verifySiwe()`

Original issue. SIWE was verified with only the signature, leaving the message `domain` unenforced — a phishing page on another origin could solicit a signature that the server would then accept, enabling session takeover.

Fix verified. `verifySiwe` now reads `siwe.domain`, rejects anything not in an `allowedSiweDomains()` allowlist (the app's own hosts, extendable via `SEALED_SIWE_DOMAINS`), and passes `{ domain, time }` into `siwe.verify(...)` so the library also enforces the validity window. A signature solicited on `attacker.com` is now rejected before the nonce is even consulted.

H-2 · Fallback session-signing secret with no fail-closed guard

HIGH FIXED

Component: `src/server/auth.ts` · `secretKey()`

Original issue. If `SEALED_SESSION_SECRET` was unset in production, session JWTs were signed with a hardcoded public string, allowing cookie forgery for any account.

Fix verified. `secretKey()` now throws when `NODE_ENV==="production"` and the secret is missing or equals the dev default — "refusing to sign sessions with a public key." The service fails closed rather than silently using a forgeable key.

H-3 · Signed-prekey check authenticated nothing to a trust anchor

HIGH FIXED

Component: `src/lib/crypto/signal/dm.ts`

Original issue. The X3DH handshake ran against the server-supplied `identityKey`, and the signed-prekey signature was checked against another server-supplied key — so an active operator could substitute a whole prekey bundle and machine-in-the-middle new conversations.

Fix verified. The sender now requires the bundle's `identityKey` to equal `peerDevicePublicKey` — the device key resolved from the directory whose bindings are recorded in the on-chain-anchored transparency log — and runs X3DH against that *anchored* key rather than the bundle field. A substituted bundle now fails to establish a session (an availability event) instead of yielding a readable one: the `DH(EK_A, IK_B)` term uses the victim's real identity key, which the attacker cannot compute. **Residual:** the strength of this now rests on the resolved device key being the anchored one, which ties back to the transparency-log guarantees in M-4.

M-1 · Private keys stored unencrypted in the browser

MEDIUM FIXED

Component: new `src/lib/keyvault.ts`; used by `crypto/device.ts`, `signal/keys.ts`, `signal/session-store.ts`

Original issue. Device key, signing key, prekey privates, and ratchet-session state sat in `localStorage` as plaintext.

Fix verified. A new key-vault encrypts each blob with AES-256-GCM under an in-memory master key, and stores the master key only encrypted under a *non-extractable* WebCrypto AES-GCM key held in IndexedDB. Reads and writes go through `secureGetItem/secureSetItem`; legacy plaintext is re-encrypted on next read. Plaintext-at-rest exposure is closed. **Residual (inherent):** script executing in the origin (XSS) can still ask the key to decrypt — the ceiling for any browser-resident identity — and the vault fails open to legacy behavior if IndexedDB is unavailable.

M-2 · Auto-backup passphrase persisted in plaintext

MEDIUM FIXED

Component: `src/lib/backup.ts`

Original issue. The hourly-backup passphrase was written to `localStorage` in cleartext, defeating the backup's own encryption for a local attacker.

Fix verified. The passphrase is now stored as `{iv,ct}` encrypted under a non-extractable IndexedDB AES-GCM key (`sealed.backup.pass.v2`); the legacy plaintext key is migrated then purged, and both are excluded from portable backups. Same in-origin ceiling caveat as M-1 applies.

M-3 · Groups & rich payloads (box-v1) lack forward secrecy

MEDIUM PARTIAL

Component: `lib/crypto/device.ts` / `lib/wire.ts`; marketing copy

Original issue. Group and non-DM payloads use an ephemeral-static sealed box to the recipient's long-term key — no forward secrecy or cryptographic sender authentication — while the hero copy implied global forward secrecy.

Status. The *marketing-scope* half is addressed: the landing page now scopes forward secrecy and post-compromise security to "direct messages." The *technical* half — a forward-secret group ratchet (sender keys / MLS) — remains a documented roadmap item, which is a reasonable v1 position. No regression; recommendation stands as a future enhancement.

M-4 · AnchorLog integrity gaps & single-publisher centralization

MEDIUM OPEN

Component: `contracts/AnchorLog.sol`, `server/merkle.ts`

Issue. `publish()` enforces only a monotonic tree size — not root uniqueness, `root != 0`, or an RFC-6962 consistency proof — and the publisher is a single immutable wallet (a liveness/centralization single point of failure).

Status: open (deferred by design). The contract and Merkle module are unchanged, and the on-chain state still shows a single checkpoint, so entries beyond it remain server-trusted until the next anchor. This is the one substantive item not addressed this round; it legitimately requires a contract redeploy plus a regular anchoring cadence. Recommended before scaling trust in the directory: publish checkpoints on a schedule, verify inter-checkpoint consistency client-side, require `root != 0`, and move the publisher to a multisig or add rotation.

Low-severity findings

L-1 · Global payment-verification bypass switch

LOW FIXED

The `SEALED_VERIFY_PAYMENTS=off` escape hatch now hard-fails in production: `payments.ts` throws *"Payment verification cannot be disabled in production"* when `NODE_ENV==="production"`, restricting the bypass to local/test builds.

L-2 · Global admin authority tied to a claimable handle

LOW FIXED

`admin.ts` now derives `ADMIN_ADDRESSES` from `OPERATOR_ADDRESSES` plus an env allowlist and gates `isGlobalAdmin()` on the address — the transferable `@eas` handle no longer confers admin.

L-3 · Anonymous rate-limit keyed on a spoofable header

LOW FIXED

The anonymous relay now keys its limiter via a shared `rateKey("anon", req)` helper (`@/server/client-ip`) rather than the raw `X-Forwarded-For` value.

L-4 · Unbounded opaque blob storage

LOW OPEN

`api/backup/route.ts` is unchanged and still accepts an arbitrarily large `blob` with no size cap — a low-severity memory/disk DoS vector. **Recommendation:** add per-blob and per-account size quotas. Low priority.

L-5 · QR link-login could authorize an attacker device

LOW FIXED

Link-login now validates the device key format and exposes a preview (`deviceFingerprint` + metadata) so the approver can compare the requesting device's fingerprint before approving — closing the blind-approval phishing gap.

Informational items

ID	Status	Note
I-2	FIXED	FAQ now states the "content, not metadata" scope and names the routing data the relay handles.
I-3	FIXED	Privacy policy adds the same metadata caveat; freeze framing remains scoped to funds.
I-1	OPEN	Ratchet headers remain unencrypted (established DMs already use rotating sealed-sender mailbox tags, which mitigates the social-graph exposure). Consider header encryption.
I-4	OPEN	Backup passphrase minimum is still 8 characters — raise it / add a strength meter.
I-5	OPEN	Proof-of-sealing remains a printable-ASCII heuristic (defense-in-depth); wording is otherwise fair.

7 · Cryptography review — X3DH & Double Ratchet

The direct-message cryptography was already the strongest part of the system and is now better anchored.

Purpose	Primitive	Source
Handshake & ratchet DH	X25519	`@noble/curves`
Signed-prekey signatures	Ed25519	`@noble/curves`
Message AEAD	AES-256-GCM (96-bit IV, 128-bit tag)	`@noble/ciphers`
Key derivation	HKDF-SHA-256	`@noble/hashes`
Backup KDF	Argon2id (t=3, m=64 MiB, p=1)	`@noble/hashes`
At-rest key wrapping **new**	AES-256-GCM under a non-extractable WebCrypto key	`lib/keyvault.ts`

X3DH performs all three (or four, with a one-time prekey) Diffie–Hellman computations correctly, binds both identity keys into the associated data, and — after the H-3 fix — runs against the transparency-anchored identity key rather than an unauthenticated server field. The Double Ratchet correctly implements the symmetric and DH ratchets, deletes message keys after use, and bounds skipped keys. One test-hardening suggestion from v1.0 still stands: add a test that a validly-signed bundle from the *wrong* key is rejected, so the H-3 protection is covered by a regression test.

> **Bottom line on the crypto**
>
> For direct messages, Aphanite implements the Signal handshake and ratchet correctly on audited primitives, now with key material encrypted at rest and the handshake bound to the on-chain-anchored identity. The remaining crypto-adjacent work is a forward-secret group scheme (M-3) and the transparency hardening in M-4.

8 · Smart contract review — AnchorLog

`AnchorLog.sol` is the only on-chain component: a minimal append-only checkpoint log, unchanged since v1.0. Its deployment and state were re-confirmed live on Ethereum mainnet.

> **On-chain verification (re-confirmed)**
>
> **Contract:** `0x3b0467414B33707d0D925139738e34871f2fF36D` — Ethereum mainnet.
>
> **Publisher:** `0x5aD8Fc47BEa64f5666994D0ABEC16382A6F31e2B` (immutable).
>
> **State:** `count() = 1` — still a single checkpoint anchored. Everything past that checkpoint remains server-trusted until the next anchor is published (M-4).

The contract is real, live, append-only, and genuinely checked by the client (the anchor address is baked into the bundle so the server cannot substitute it). The open items — inter-checkpoint consistency, `root != 0`, a non-single publisher, and a regular anchoring cadence — require a redeploy and are tracked as M-4.

9 · What Aphanite does well

- **Fast, correct remediation.** Every High finding and most Mediums/Lows were fixed at the source level, with in-code comments referencing the finding IDs — a sign of a team that takes review seriously.
- **Correct, standard crypto.** X3DH and the Double Ratchet are implemented faithfully for DMs on audited `@noble` primitives, now anchored to the transparency log and with keys encrypted at rest.
- **Transparency that is actually verified.** The client reads the checkpoint root from Ethereum via a public RPC and compares it to the served log, surfacing mismatches as provable tampering — "on-chain" as a control, not a slogan.
- **Content is genuinely opaque to the relay,** which stores ciphertext only and rejects plaintext; private keys never leave the device.
- **Non-custodial by construction;** the wallet signs, the chain settles.
- **Honest documentation — now including the short-form copy.** The FAQ and privacy policy state the metadata scope plainly, and the marketing avoids high-risk superlatives. Absolutes have been aligned to what the whitepaper always said.

10 · Remaining work

Priority	Items	Action
Medium	M-4	Harden the transparency anchor: publish checkpoints on a cadence, verify inter-checkpoint consistency in the client, require `root != 0`, and move the publisher to a multisig or add rotation. Requires a contract redeploy.
Low	L-4, I-4	Add backup/blob size quotas; raise the backup passphrase minimum and add a strength meter.
Enhancement	M-3, I-1, tests	Add a forward-secret group ratchet; consider ratchet-header encryption; add a regression test for the H-3 key-substitution case.
Assurance	Program	Commission an independent third-party audit and, ideally, machine-checked models of the handshake/ratchet before broad deployment — as the project's own specification recommends. Confirm the open-source relay repository is public and complete.

Appendix A · Reference

A.1 Severity model

Severity	Meaning
CRITICAL	Breaks a core guarantee with no precondition. **None found.**
HIGH	Enables takeover, MITM, or impersonation (often requiring an active operator, phishing, or misconfiguration). **All resolved.**
MEDIUM	Weakens defense-in-depth or a documented property under a plausible secondary condition.

LOW	Limited or conditional impact — abuse, paid-feature integrity, or DoS.
INFO	Hygiene, scope-of-claim, and hardening observations.

A.2 Ciphersuite (as implemented)

```
secp256k1/ECDSA (SIWE) · X25519 · Ed25519 (prekey sigs) · X3DH + Double Ratchet (dr-v1) · X25519 sealed box (box-v1) · HKDF-SHA-256 · AES-256-GCM · SHA-256 Merkle · Argon2id (backups) · AES-256-GCM key-vault under a non-extractable WebCrypto key (at-rest) · JWT HS256 sessions · @noble primitives.
```

A.3 On-chain anchor (re-verified)

Network	Ethereum mainnet (chain id 1)
AnchorLog contract	`0x3b0467414B33707d0D925139738e34871f2fF36D`
Publisher (immutable)	`0x5aD8Fc47BEa64f5666994D0ABEC16382A6F31e2B`
Checkpoints published	1

A.4 Disclaimer

This document reflects a good-faith, point-in-time source-code security review and remediation verification of Aphanite v1 as of the report date. It is a transparency resource, not a warranty, certification, or guarantee of security, nor legal or financial advice. Confirming that a fix is present and correct in source is not equivalent to proving the absence of all related vulnerabilities. Code, configuration, and on-chain state can change after publication. This review does not replace an independent third-party audit, formal verification, or a live penetration test, each of which is recommended before treating Aphanite as suitable for high-risk use. Severity ratings and verdicts are this reviewer’s professional judgment.

— End of report · Aphanite v1 Security & Claims Audit · Version 1.1 (Remediation Review) —