EverstoneBTC

EVERSTONE PROOF ARCHITECTURE

PROVE IT TRANSACTION WORKFLOW

From private document to Bitcoin timestamp.

Everstone Proof hashes the user's content in the browser, stores only a hash plus encrypted bundle metadata, accepts payment, then writes `EVST-PROOF:<hash>` into Bitcoin OP_RETURN.

1

User prepares proof

At `/proof/create`, user enters text and/or selects a file, title, owner name, email, and privacy mode.

2

Client-side SHA-256

Browser computes a 64-hex SHA-256 hash. The original file never leaves the user's device.

3

Register record

`prepare/register` stores hash metadata, slug, `bundleSecret`, status `PENDING`, payment `UNPAID`.

4

Seal bundle

Client builds certificate bundle. Private proofs are encrypted in-browser before commit.

CONTINUE BELOW

5

Commit bundle

`prepare` phase `commit` stores encrypted blob or public JSON and marks `bundleStoredAt`.

6

Checkout

Payment is **$25 + dynamic BTC fee**. Square handles card/Apple Pay; OpenNode handles Bitcoin/Lightning.

7

Webhook verifies

Provider webhook checks signature/status, finds `TimestampRecord`, and atomically flips `UNPAID → PAID`.

8

Anchor lock

`withAnchorLock(proof:hash)` prevents concurrent duplicate anchors for the same content hash.

CONTINUE BELOW

9

Bitcoin OP_RETURN

`anchorProof()` spends treasury UTXO and embeds `EVST-PROOF:<64-char hash>`.

10

Broadcast + recovery

TX broadcasts via `mempool.space/api/tx`. Anchor recovery logs txid before final DB update.

11

Proof anchored

Record becomes `ANCHORED` with txid. If anchoring fails, record stays `PAID` for retry.

12

Receipt + share

User receives proof receipt/bundle. Share link reveals hash, txid, timestamp — not private content.

EverstoneBTC
OPERATIONAL REFERENCE

IMPLEMENTATION NOTES

Proof is hash-first, zero-knowledge by design.

This page captures the code-level ownership: record lifecycle, privacy boundary, payment routes, anchoring route, and receipt behavior.

Proof state machine

PENDING	Record exists with content hash, slug, email, title/owner metadata, and bundle secret.
UNPAID	Payment not completed. Bundle may already be pre-sealed so browser closure does not lose it.
PAID	Square/OpenNode webhook completed and verified. If Bitcoin anchoring fails, this state is retryable without charging again.
ANCHORED	`anchorProof()` succeeded and txid is stored. Current Proof flow stores txid immediately after broadcast.
SEALED	Not a status string; operationally means `bundleStoredAt` is set and encrypted/public bundle data is retrievable.
SHARED	Public share page can prove timestamp/hash/txid without exposing the underlying document contents.

Core components

Frontend	`/proof`, `/proof/create`, `/proof/[slug]`, `/proof/[slug]/share`, `/verify`.
Prepare API	`/api/proof/prepare` register/commit creates record and stores client-encrypted bundle.
Payments	`/api/payments/create-proof-square` and `/api/payments/create-proof`.
Anchoring	`lib/anchoring.ts` creates treasury-funded Bitcoin TX using OP_RETURN payload `EVST-PROOF:<hash>`.
Bundle APIs	`/api/proof/[slug]/bundle`, `/seal`, `/download`; secret-gated unless proof is public.
External APIs	mempool.space for fees, UTXOs, tx broadcast, and BTC/USD price; Square/OpenNode for payment completion.

Privacy boundary

Private Proof is zero-knowledge: the server receives the content hash and encrypted bundle, not the plaintext document. The user's passphrase stays client-side.

Chain payload

Proof uses `EVST-PROOF:<64-char hash>` directly, avoiding the old overflow risk and keeping the OP_RETURN payload inside current limits.

Product promise

The shareable proof link is the point: "I can prove this existed then" without revealing what "this" is unless the owner chooses to open the sealed envelope.