Digital ownership, made possible with Bitcoin

Our Vision

We're building a **universal provenance layer** for digital content—files, posts, AI-generated images and videos—that lets anyone:

- **Verify authenticity:** confirm the content is intact, who claimed it, and when.
- **Prove authorship:** cryptographically anchored to Bitcoin time.
- **Transfer ownership:** keep a clear, tamper-evident chain of custody.

This goes beyond simple timestamping. It's a cheap, open, infinitely scalable **Intellectual Property layer for the internet**. In an era of generative AI, knowing *who created what, and when* is critical.

How it works

Digital fingerprint (file) + Your signature → Anchored in Bitcoin (OpenTimestamps)

1. **Digital fingerprint:** We hash the file (SHA-256).
2. **Your signature:** You sign that hash with your private key
3. **Bitcoin timestamp:** We anchor the event on Bitcoin using OpenTimestamps (OTS).
4. **Ownership log:** A single JSON manifest stores the file's fingerprint and an append-only list of signed events (mint, transfers), each with its own OTS proof.

What you have:

- The **file** (or post) itself
- A single **provenance manifest JSON** (contains the complete history and embedded OTS proofs)

User-friendly interface

Digital signatures – (we may need to develop our own stamps here)

Verified files

Technical Implementation

File Layout (two files only)

- artifact.bin
- artifact.json

Data Model

Manifest (manifest/v1)

- **artifact**: file name, size, sha256_hex
- **events**[]: ordered, append-only list of events

Event (event/v1)

- **index**: 0, 1, 2, …
- **action**: "mint" | "transfer"
- **artifact_sha256_hex**: must match manifest.artifact.sha256_hex
- **prev_event_hash_hex**: null for first, else prior event's hash
- **actors**: keys involved (creator / prev_owner / new_owner)
- **issued_at**: ISO-8601 time
- **event_hash_hex:** SHA-256 of canonical event (exclude signatures, ots_proof_b64, event_hash_hex)
- **signatures**: detached signatures over event_hash_hex
- **ots_proof_b64**: embedded OpenTimestamps proof

Signing rule: compute event_hash_hex, then each required actor signs that hash.

Example: artifact.json

```
{
  "type": "provenance.manifest/v1",
  "artifact": {
    "file_name": "artifact.bin",
    "size_bytes": 482133,
    "sha256_hex":
"7f6b5ee8b3f0d40c28efc4037a0e682ec52c7db58b0530e03b8c55dd9f31c2a9"
  },
  "events": [
    {
      "type": "provenance.event/v1",
      "index": 0,
      "action": "mint",
      "artifact_sha256_hex":
"7f6b5ee8b3f0d40c28efc4037a0e682ec52c7db58b0530e03b8c55dd9f31c2a9",
      "prev_event_hash_hex": null,
      "actors": { "creator_pubkey_hex": "02a1...bc" },
      "issued_at": "2025-09-25T14:12:34Z",
      "event_hash_hex": "ab12...ef",
      "signatures": { "creator_sig_hex": "3045...01" },
      "ots_proof_b64": "AAABASE64ENCODED_OTS_PROOF_FOR_EVENT_0..."
    },
    {
      "type": "provenance.event/v1",
      "index": 1,
      "action": "transfer",
      "artifact_sha256_hex":
"7f6b5ee8b3f0d40c28efc4037a0e682ec52c7db58b0530e03b8c55dd9f31c2a9",
      "prev_event_hash_hex": "ab12...ef",
      "actors": {
        "prev_owner_pubkey_hex": "02a1...bc",
        "new_owner_pubkey_hex": "03de...55"
      },
      "issued_at": "2025-10-01T09:00:00Z",
      "event_hash_hex": "cd34...88",
      "signatures": {
        "prev_owner_sig_hex": "3045...02",
        "new_owner_sig_hex": "3044...03"
      },
      "ots_proof_b64": "AAABASE64ENCODED_OTS_PROOF_FOR_EVENT_1..."
    }
  ]
}
```

Verification Flow

1. Re-hash file → must equal artifact.sha256_hex.

2. For each event:
    - Recompute event_hash_hex from canonical event.
    - Verify prev_event_hash_hex links correctly.
    - Verify all listed signatures over event_hash_hex.
    - Verify ots_proof_b64 → Bitcoin block/time.
3. Current owner = last valid event's actor (e.g., new_owner_pubkey_hex).

Scalability

- OTS batching: millions → billions of events in one Bitcoin tx.
- Proof size: small (KB), logarithmic growth.