Digital Gold Certificates

A Peer-to-Peer Proof-of-Ownership System for Custodied Physical Assets

Abstract. This document proposes a practical design for representing custodied physical gold with digitally signed certificates whose ownership history is recorded on a blockchain. The goal is not to claim that a ledger can prove physical reality, but to make certificate integrity and transfer history tamper-evident, auditable, and programmable. We describe a minimal protocol for issuance, verification, peer-to-peer transfer, and redemption, plus a reference marketplace flow (escrow + settlement) and operational controls (reconciliation, risk scoring, and dispute handling).

Design goal	Outcome
Tamper-evident certificates	Any modification breaks signature and hash proof.
End-to-end ownership history	Every transfer is an on-chain event, traceable from issuer to current owner.
Privacy by design	Public history is pseudonymous; sensitive identity stays off-chain.
P2P trading-ready	Escrow, settlement states, and dispute hooks are first-class.

Status. Portfolio draft. This is a technical concept for prototyping. It intentionally avoids referencing any specific institution and does not constitute legal, financial, or regulatory advice.

Contents

1. Motivation and Problem Statement

Physical gold is a bearer asset in the real world, but modern distribution often relies on custodianship and digital representations. When ownership records live solely inside a traditional database, users must trust that the record cannot be silently altered, backdated, or duplicated. Disputes become slow and expensive because the evidence is controlled by the same system being disputed.

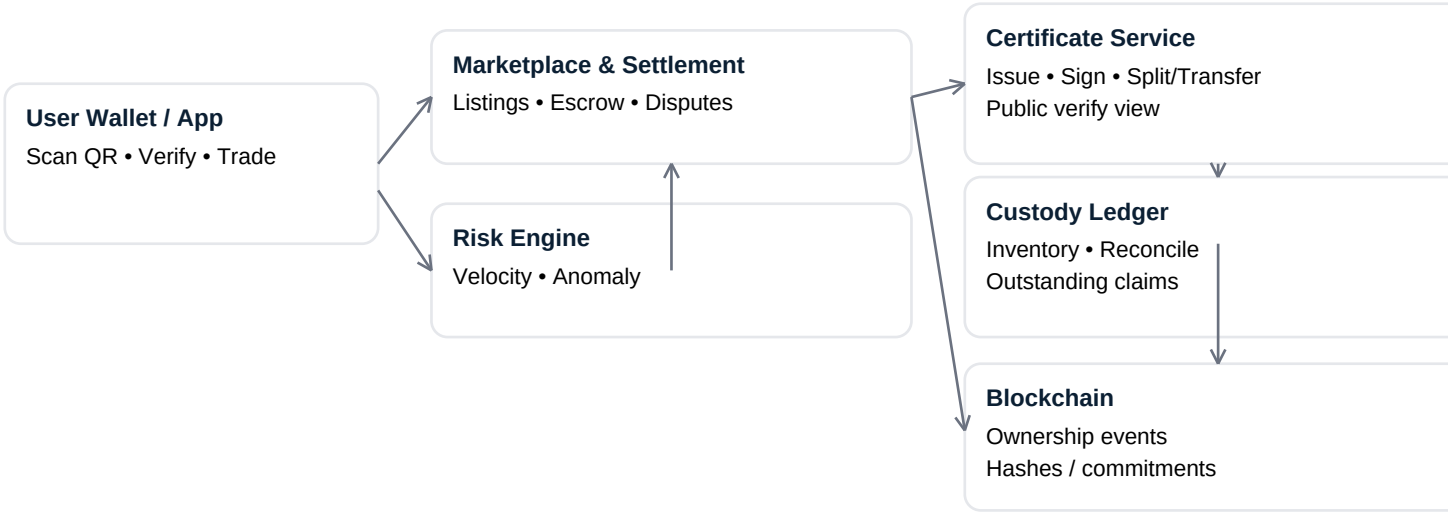
This paper proposes digital gold certificates (DGCs): digitally signed documents that represent a claim on custodied physical gold. Every certificate transfer is recorded as a blockchain event, producing a globally verifiable ownership history from initial issuance to the current holder. The design is intentionally minimal: it aims to be implementable by a small team, auditable by third parties, and usable as a real service rather than a demo.

Why a blockchain at all? Because an append-only ledger with decentralized validation makes it hard to rewrite history. The ledger is not a substitute for vault controls or audits; it is a way to make digital records tamper-evident and independently verifiable.

2. System Overview

The system separates three concerns: (a) custody and inventory accounting, (b) certificate lifecycle management (issue, verify, transfer, redeem), and (c) a public ownership-history ledger. Peer-to-peer (P2P) trading is implemented as an off-chain marketplace with escrow and settlement states, while the ownership transfer itself is committed on-chain so the lineage remains transparent.

Figure 1. High-level components and trust boundaries.



Actors. Users hold certificates in a wallet-like application. A certificate service issues and signs certificates. A custody ledger tracks physical inventory and outstanding claims. A blockchain stores ownership events and proofs. A marketplace service enables listings, escrow, and settlement. A risk engine consumes events to detect anomalous behavior (velocity, wash trading patterns, repeated disputes, etc.).

3. Certificate Format and Cryptographic Proof

A DGC is a canonical document (e.g., canonical JSON) signed by the issuer. The issuer signature prevents forgery; canonicalization ensures that hashing produces a stable digest across systems.

3.1 Canonical fields (illustrative)

```
{ "cert_id": "DGC-2026-02-08-000001", "version": 1, "issued_at":
"2026-02-08T12:00:00Z", "issuer_id": "ISSUER-VAULT-01", "owner_pubkey":
"03ab...91", "gold": { "amount_gram": 1.0000, "purity": "999.9" }, "status":
"ACTIVE", "parent_cert_id": null, "metadata": { "batch_ref": "BAR-LOT-2026-01",
"tx_ref": "SALE-12345" } }
```

The system computes a SHA-256 hash of the canonical document and stores that hash as a proof anchor. The certificate is also signed by the issuer's private key (preferably held in a KMS/HSM).

3.2 Proof record

A proof record contains (cert_id, hash, timestamp, ledger_tx_ref). Proof commits may be batched to reduce fees and latency. Verification is done by re-hashing the retrieved certificate and comparing it with the on-chain record, while also validating the issuer signature.

4. Ownership History on Blockchain

The core requirement is that historical ownership is recorded on-chain so the movement of a certificate can be traced from seller to buyer. This can be implemented in two common models: UTXO-style (spendable outputs) or account-style (balances and transfers). The portfolio prototype may use an account-style ledger for simplicity.

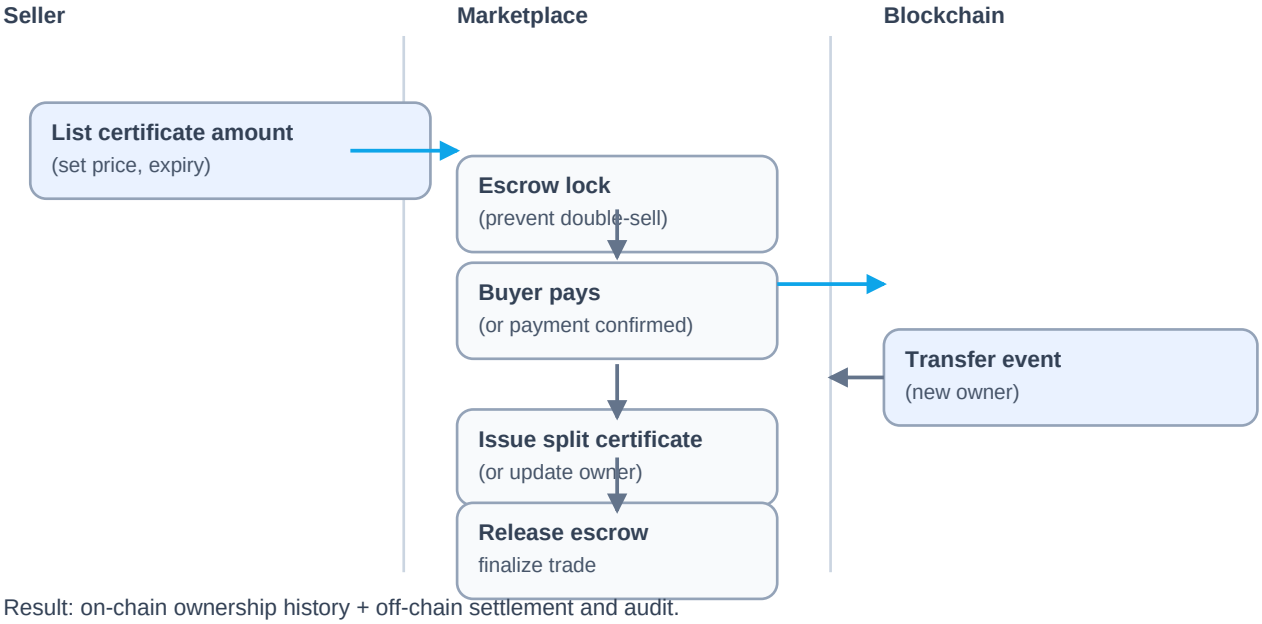
4.1 On-chain events

Each transfer is an on-chain transaction that emits an event such as:

```
Transfer( cert_id, from_address, to_address, amount_gram, price_paid, timestamp, proof_hash )
```

This yields a complete public lineage graph: any observer can follow cert_id (or split lineage) across transfers. Because this history is public, identities should be represented by pseudonymous addresses. If a deployment requires confidentiality, the price or other fields may be encoded as commitments or encrypted memos, while still recording the transfer edges.

Figure 2. Simplified P2P trade flow: off-chain escrow, on-chain transfer.



Result: on-chain ownership history + off-chain settlement and audit.

5. Peer-to-Peer Trading Model

P2P trading is modeled as a marketplace that coordinates discovery and settlement, while the blockchain records the definitive ownership transfer. This division keeps the user experience fast (matching and payment states are off-chain) while ensuring that final ownership is tamper-evident (on-chain).

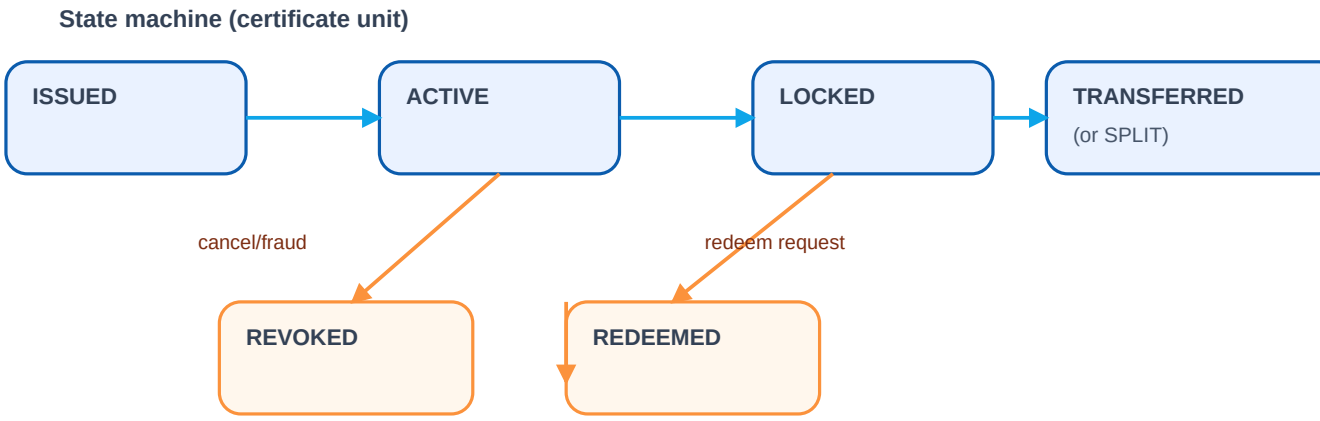
5.1 Listings and escrow

A seller creates a listing by selecting a certificate (or a fraction of its amount). The system places the selected amount into an escrow lock, preventing double-selling. Escrow can be implemented as a database lock plus an on-chain constraint (e.g., the contract refuses to transfer more than unlocked amount).

5.2 Transfer or split

For partial trades, the recommended approach is **split certificates**: the trade creates a new child certificate for the buyer while reducing the seller's amount. Parent-child relationships preserve lineage and enable precise audit trails.

Figure 3. Certificate state machine (unit-level).



6. Privacy and Data Disclosure

Recording detailed transaction history on a public blockchain creates an immediate tension between auditability and privacy. This design chooses traceability of ownership edges while protecting user identity through pseudonymity and optional encrypted disclosures.

Recommended privacy controls:

- Address rotation: wallets generate new addresses per trade to reduce linkability.
- Public vs owner view: public verification shows certificate details without personal identity.
- Encrypted memos: optionally encrypt buyer/seller references or invoices, stored off-chain and referenced by hash.
- Selective disclosure: authorized parties can reveal identity bindings (e.g., via signed attestations) when required.

In a portfolio prototype, it is acceptable to keep price and amount on-chain for demonstrability, while using placeholder identifiers and explicitly documenting the privacy tradeoffs.

7. Security Considerations and Threat Model

The system must defend against both external attackers and internal failures. A blockchain reduces the risk of silent history rewriting, but it does not eliminate operational threats such as key compromise or custody fraud. Below is a minimal threat model for a production-grade implementation.

Threat	Mitigation
Forgery of certificates	Mitigate with issuer signatures; verify signature on every read and transfer.
Key compromise (issuer)	Use KMS/HSM, rotation, and multi-party approval for high-value issuance.
Double-selling / race conditions	Escrow locks, idempotency keys, and contract-level transfer checks.
Marketplace fraud / wash trading	Risk scoring on event streams, limits, and dispute freezes.
Custody mismatch (digital vs physical)	Daily reconciliation; independent audits; separation of duties.
Privacy leakage via on-chain metadata	Keep identity off-chain; rotate addresses; encrypt optional memos.

8. Operational Controls: Inventory and Reconciliation

Because the underlying asset is physical, operational controls are as important as cryptography. The custody ledger tracks total inventory (vault holdings) and outstanding digital claims (sum of ACTIVE and LOCKED amounts). A reconciliation job runs periodically to ensure these values match under defined tolerances, producing a report and alerts on mismatch.

Minimal reconciliation equation:

```
total_physical_gold_gram >= sum(outstanding_certificate_amount_gram)
```

A portfolio prototype can simulate custody inventory and show how mismatches trigger a freeze or require manual review before further transfers are allowed.

9. Implementation Notes (Portfolio Scope)

A strong portfolio implementation should emphasize correctness, auditability, and observability. Suggested modules:

Module	Responsibilities
certificate-service	Issue/sign certificates; owner and public views; split/transfer; OpenAPI spec.
ledger-adapter	Commit and verify proof records; batching; retries; idempotency.
marketplace-service	Listings; escrow; settlement state machine; dispute hooks.
risk-stream	Consume events; compute risk score; trigger holds; simple dashboard.
web-verifier	QR scan/verify view; ownership timeline visualization.

Deliverables that signal senior engineering: a threat model, an architecture diagram, load tests, structured logs, and a clear explanation of privacy tradeoffs.

10. Limitations and Future Work

This design deliberately avoids making absolute claims about physical reality. A blockchain can show that a transfer was recorded and cannot be rewritten easily, but it cannot verify vault contents by itself. Future work includes stronger privacy (commitments or zero-knowledge proofs for price), multi-party issuance approvals, hardware-backed key storage, and standardized attestations from independent auditors.

References (non-exhaustive):

- Nakamoto, S. Bitcoin: A Peer-to-Peer Electronic Cash System (2008).
- RFC 8032: Edwards-Curve Digital Signature Algorithm (EdDSA).
- RFC 2104: HMAC: Keyed-Hashing for Message Authentication.

Appendix A. Business Requirements Document (BRD)

This appendix reframes the proposal as a product/service BRD. It is written to be implementation-friendly: clear scope, functional requirements, non-functional requirements, and success criteria.

A1. Objectives

- Provide a verifiable digital proof of ownership for custodied physical gold.
- Enable peer-to-peer trading of certificates with final ownership recorded on-chain.
- Reduce dispute cost by making certificate history tamper-evident and independently verifiable.
- Offer a public verification experience (QR / link) without exposing user identity.

A2. In-Scope Users and Stakeholders

Role	Responsibilities
End user	Holds certificates, verifies authenticity, sells/buys in P2P market.
Custodian operator	Manages inventory, issuance policy, redemption, and reconciliation.
Marketplace operator	Runs matching, escrow, settlement, and dispute workflow.
Auditor / regulator (optional)	Verifies supply vs outstanding claims; reviews trails under authorization.

A3. Core User Journeys

Journey	Definition of Done
Buy & receive certificate	User completes purchase; certificate is issued; on-chain proof is committed; user sees QR/link.
Verify certificate	Anyone can verify issuer signature + on-chain proof hash + current owner address.
Transfer (gift)	Owner transfers certificate to another address; history updates on-chain.
P2P trade	Seller lists; escrow locks; buyer pays; on-chain transfer finalizes; escrow releases.
Redeem	Owner requests physical redemption; certificate is locked; redemption completes; certificate is marked REDEEMED.

A4. Functional Requirements

ID	Requirement	Acceptance Criteria
FR-01	Issue certificate	Create a signed certificate for a specified amount and bind it to an owner address.
FR-02	Commit proof	Anchor certificate hash (or commitment) on-chain and store tx reference.
FR-03	Public verify	Verify signature + on-chain proof + status (ACTIVE/LOCKED/REDEEMED/REVOKED).
FR-04	Transfer	Transfer ownership on-chain; update ownership timeline in the app.
FR-05	Split	Support partial transfers by splitting parent certificate into child certificates.
FR-06	P2P listing	Create, update, and cancel listings with price and expiry.
FR-07	Escrow lock	Prevent double-sell via lock state during a pending trade.
FR-08	Settlement	Finalize trade by triggering on-chain transfer after payment confirmation.
FR-09	Disputes	Provide a dispute workflow (freeze, investigate, resolve) with audit logs.
FR-10	Reconciliation	Compute outstanding claims vs inventory; alert and optionally freeze transfers on mismatch.

A5. Non-Functional Requirements

Category	Target
Security	Issuer keys in KMS/HSM; least privilege; tamper-evident logs; rate limits.
Availability	99.9% monthly for verifier and wallet APIs (prototype target: graceful degradation).
Latency	Verification page load < 2s; transfer initiation < 500ms (excluding chain finality).
Scalability	Event-stream based ingestion; horizontal scale for marketplace; batching for proof commits.
Auditability	All lifecycle changes are immutable events with correlation IDs.
Privacy	No personal identity on public chain; support address rotation; optional encrypted memos.

Appendix B. Implementation Milestones

This appendix outlines a long-term, iteration-friendly path from a reference implementation to production-grade controls. Milestones are structured by capability rather than by a fixed calendar.

B1. Baseline Capability Milestones

- Certificate issuance + signing + QR verifier page.
- On-chain proof commit + ownership transfer events.
- Wallet UI: portfolio view + ownership timeline (graph).
- Basic P2P market: listings + escrow lock + settlement state machine (mock payment).
- Reconciliation simulator + admin dashboard + freeze switch.

B2. Operational Hardening and Extensions

- Payment integration + idempotent settlement + dispute workflow.
- Risk engine (stream processing) + anti-wash-trading heuristics.
- Confidential price options (commitments/encrypted memos).
- Key management hardening + multi-party issuance approvals.
- External auditor attestation format + public transparency report.