# SCDO Agent Ledger: On-Chain Identity, Work Attestation and Settlement for Autonomous AI Agents on a Sharded PoW Blockchain

**Version 5 (18 October 2026)**

Ying Willam Zhang
SCDO Laboratory, 9Y9 PTY LTD, Melbourne, Australia
ORCID 0009-0002-1526-3332

18 October 2026 (AEDT, UTC+11). Licence: CC BY 4.0

**Abstract**

Agentic AI systems composed of multiple large-language-model (LLM) agents need verifiable agent identity, tamper-evident records of completed work, and machine-to-machine settlement. Version 1 of this preprint presented AgentLedger, an owner-less smart contract on SCDO Shard0, an EVM-compatible proof-of-work (PoW) shard, that records agent registration, Merkle roots of batches of EIP-712-signed task messages, and pay-per-task receipts, measured on a seven-agent deployment. This version reports what has been built and observed since then, using only on-chain data and public server endpoints. (i) Three ERC-8004 "Trustless Agents" registries (identity, reputation, validation) are deployed on SCDO Shard0; the identity registry holds 29 agent identities (agentId 0–28), and the main SCDO agent is agentId 2. The reputation and validation registries have emitted no events yet. (ii) A per-bot x402 payment layer gives each internal agent a USDC wallet on Base and an HTTP 402 pay route (default 1 USDC, used here at 0.01 USDC per call). The public treasury page lists 50 agents, each with a Base USDC reserve and an x402 receive link. One 0.01 test-USDC payment to each of 34 roster wallets settled on Base Sepolia between 18:29:42 and 18:31:12 AEDT on 2026-10-18. (iii) A headless Linux bookkeeping agent, scdo-agent 0.5.x, adds rootless installation, a daily auto-update from a signed release feed, operator bulk registration, e-mail binding, and booking of x402 paywall income into the Agent Ledger; a Base Sepolia income transaction is booked as a confirmed ledger entry. (iv) Motivated by a GPU-adapter agent found on 2026-10-18 on a host with no discrete GPU, we describe a proof-of-GPU scheme: a timed ethash challenge plus a signed, one-per-agent GPU UUID attestation, with GPU agents unverified by default. It is implemented in an open pull request and not deployed. (v) The self-audit log now has 136 signed records, all signature-verified, anchored in 51 batches on Shard0. We also correct an error in Version 1: two transactions labelled payTask were plain native transfers. The main limitations: no externally operated agents have joined yet, mainnet USDC reserves hold only nominal seed balances, and the reputation and validation registries are unused.

## Status of this document

This is **Version 5** of the preprint. It supersedes Zenodo record 10.5281/zenodo.23209794 (published 2026-10-07; concept DOI 10.5281/zenodo.23119809; the first deposit, 10.5281/zenodo.23119810, is dated 2026-10-03). In the text, "Version 1" or "v1" means that published preprint (the content baseline this version updates). Zenodo labels record 23209794 as "v4" because three metadata and layout revisions preceded it; this version is therefore labelled **Version 5** (v5) to follow that Zenodo sequence. All new figures were read on **2026-10-18 between about 18:40 and 18:55 AEDT (UTC+11)**, when SCDO Shard0 was at block 148,679 (18:40:56 AEDT). Measurements carried over from v1 keep their original UTC+8 timestamps. Shortened hashes (`0x1234…abcd`) are expanded in Appendix A.

## Changelog with respect to Version 1

| Area | Version 1 (Zenodo 10.5281/zenodo.23209794) | Version 5 |
|---|---|---|
| Agent identity | Purpose-built `registerAgent` in AgentLedger; ERC-8004 mapping listed as future work | ERC-8004 identity, reputation and validation registries deployed on Shard0; 29 identities (agentId 0–28); main agent agentId 2 (Section 4) |
| Settlement | Native-unit `payTask` only; x402 described but "not claimed as implemented" | Per-bot x402 wallets (USDC on Base) and pay routes; 50-agent roster; 34 Base Sepolia settlements; mainnet routes live with nominal seed balances (Section 5) |

| Area | Version 1 (Zenodo 10.5281/zenodo.23209794) | Version 5 |
|---|---|---|
| Agent software | Server-side signing helper only | Headless Linux `scdo-agent` 0.5.x: rootless install, signed auto-update feed, operator batch with Merkle root, e-mail bind, x402 paywall income booking (Section 6) |
| Sybil / hardware claims | Not addressed | Proof-of-GPU design and implementation (timed ethash challenge, signed one-per-agent GPU UUID, unverified by default); open PR, **not deployed** (Section 7) |
| Work attestation | 7 messages in 1 batch | 136 signed records in 51 batches by the batcher (plus 8 batches anchored by a second deployment); 59 `RootAnchored` events on-chain (Section 8) |
| Live statistics | One afternoon window | Ledger v2: 73 bookkeeping agents (30 online at the snapshot), 56 `AgentRegistered`, 9 `TaskPaid` (Section 9) |
| Corrections | Table IV listed two `payTask` calls | Those two transactions were plain native transfers, with no `TaskPaid` event (Section 10); the small settlement tier is now 2 units, not 1 |
| Terminology | “mining” for operator nodes | “Bookkeeping node” and “ledger” for nodes run by the operator. “PoW” is kept only for the consensus mechanism of the host chain |
| Limitations | Single operator, single key file | Adds: no external agents yet; mainnet reserves not operationally funded; reputation and validation registries unused; proof-of-GPU undeployed (Section 11) |

# Introduction

LLM-based agents are increasingly used to carry out multi-step tasks with limited human supervision, and multi-agent configurations in which specialised agents cooperate are an active research area [1], [2]. When several AI agents cooperate on behalf of an organisation, three practical requirements arise. Each agent needs a persistent, machine-verifiable identity. There must be a tamper-evident record of which agent produced which work product, and when. And cooperating agents need programmable settlement.

Version 1 described AgentLedger, a contract on SCDO Shard0 that records agent self-registration, Merkle-batched anchoring of EIP-712-signed task messages, and direct-forwarding payments with on-chain receipts. It also described the design of a zero-reward, zero-gas coordination chain that would checkpoint to SCDO Shard0, and it compared public-PoW anchoring with consortium hash anchoring of the kind AntChain offers. The future work it listed included mapping AgentLedger identities to ERC-8004 registries [6] and adding x402-style payment handshakes [18], [19].

This version reports on that future work, as built and as observed on public chains and servers. Its contributions are:

- **ERC-8004 registries on SCDO Shard0.** Identity, reputation and validation registries are deployed. 29 agent identities have been minted, and the main agent publishes an ERC-8004 registration file (Section 4).
- **Per-bot x402 wallets.** Each internal agent has a Base USDC address and an HTTP 402 pay route. A 50-agent roster is shown publicly, and testnet settlement was exercised end to end (Section 5).
- **scdo-agent 0.5.x.** A headless Linux bookkeeping agent that signs ledger entries, books x402 paywall income and updates itself from a signed feed (Section 6).
- **Proof-of-GPU.** A design and implementation, not yet deployed, against self-declared hardware claims (Section 7).
- **An updated self-audit log and live statistics** (Sections 8–9), and **a correction** to v1 (Section 10).

Sections 2–3 carry over the background and the AgentLedger design from v1, condensed. The coordination-chain design and the consortium comparison are unchanged from v1. They are summarised in Section 3.6 and are not repeated in full.

# Background and related work

**PoW, EVM and sharding.** Bitcoin introduced proof-of-work for consensus in a permissionless setting [9]. The EVM provides general-purpose smart-contract execution [10], and Ethash is a memory-hard PoW algorithm [11]. core-geth is an Ethereum-protocol client that supports PoW networks [12], and EIP-155 adds a chain identifier to transaction signatures [13]. Sharding splits state and processing across parallel chains [14], [15]. SCDO Shard0 is an EVM-compatible PoW chain built on core-geth with Ethash, a target block time of about 13 s and EIP-155 chainId 5680. Its first block is dated 2026-09-28. Its RPC endpoint and explorer are at https://scdoscan.io/rpc/0 and https://scdoscan.io, and the Agent Ledger API also lists the mirror https://apeccapital.org/rpc/0. In this paper, “bookkeeping node” means a node that SCDO Laboratory operates to keep and validate the ledger. It does not imply any reward.

**Agent economies and agent standards.** Related systems include the AEA framework [16], Fetch.ai uAgents and the Almanac registry [3], [4], Autonolas [5] and Bittensor [17]. ERC-8004 “Trustless Agents” is a Draft Ethereum standard.

It defines three registries, for identity (ERC-721 tokens whose URI resolves to an agent registration file), reputation and validation, and it leaves payments out of scope [6]. x402 revives HTTP status 402. The server answers with machine-readable payment requirements, and the client retries with a signed payment payload, typically an EIP-3009 `transferWithAuthorization` for USDC [18], [19].

**Consortium chains and hash anchoring.** Permissioned ledgers such as Hyperledger Fabric run with known participants [20]. Certificate Transparency and tamper-evident logging use Merkle structures that the public can check [21], [22]. AntChain's notarisation models write only a hash, or a link plus a hash, to a consortium chain [7], [23]. Ant Group researchers have also published work on consortium sharding, concurrency and asynchronous BFT [24]–[26].

# AgentLedger design (carried over from v1)

## Actors, threat model and goals

The actors are a coordinator agent, department agents, a human operator who remains accountable, external verifiers, and bookkeeping nodes. The adversaries considered are key compromise, hallucinated output, an operator who rewrites history, spammers, and passive observers. The goals are non-repudiable registration, an append-only and tamper-evident attestation history, and machine-verifiable payment accountability. Non-goals: proving that agent output is factually correct, keeping on-chain records confidential, and resisting a majority-hashrate adversary on Shard0.

## Identity

AgentLedger v0.1 (`0x377a…a2aa`, deployed in block 41,101, tx `0xd5cc…d44e`) has no owner, administrator or upgrade path. Any address can call `registerAgent(name, metadataURI)` for itself.

## Result attestation

An agent computes `resultHash` as the SHA-256 digest of the artefact bytes and `taskId` as the keccak-256 hash of a UTF-8 task label. It signs an EIP-712 `AgentMessage(agent, taskId, resultHash, note, timestamp, nonce)` under the domain "SCDO Agent Ledger", version 0.1, chainId 5680, bound to the contract. Every five minutes, if there are new messages, a batcher builds an OpenZeppelin-compatible sorted-pair keccak Merkle tree and calls `anchorRoot(root, count, batchURI)`. A verifier recomputes the digest and calls `verifyMessage(message, signature, batchId, proof)`. Verifiers apply confirmation depths of $k_a = 12$ for roots and $k_c = 64$ for receipts and checkpoints.

## Settlement

`payTask(taskId, to)` forwards the attached native amount and emits `TaskPaid`. It has no escrow: the contract holds no balance. The current public price list has three internal settlement tiers: small 2, normal 5 and large 20 native units. v1 listed the small tier as 1. The daily cap is 50 units per agent per UTC day. These amounts are internal bookkeeping values, not prices.

## Transparency interface

An indexer serves https://scdo.com.au/agents/ and `/agents/api/state` (also reachable at `scdoscan.io/agents/`). The page lists registrations, signed messages, batches, proofs and a re-verification endpoint (`POST /agents/api/verify`).

## Coordination chain and consortium comparison

The zero-reward, zero-gas Keccak-256 coordination chain (5 s target blocks, a whitelist of registered senders, checkpoints to Shard0 every 1–2 min through `anchorCheckpoint`) remains a design target. No `CheckpointAnchored` event has been emitted to date (Section 9). The trade-off analysis against consortium hash anchoring (v1 Tables I–II) is unchanged. Public-PoW anchoring lets any third party audit the record, at the cost of probabilistic finality.

# ERC-8004 registries on SCDO Shard0

## Deployment

Three ERC-8004 registries are deployed on SCDO Shard0 (chainId 5680). They are referenced in the Agent Ledger's aggregate registration document (`/agent-ledger/api/erc8004`) and in the main agent's registration file:

Table 2: ERC-8004 registries on SCDO Shard0. Source: `eth_getCode` and `eth_getLogs` against https://scdoscan.io/rpc/0, 2026-10-18 about 18:41 AEDT. Event names were identified by matching topic hashes against the event signatures in ERC-8004.

| Registry | Address | Code (bytes) | Events to block 148,679 |
|---|---|---|---|
| Identity | `0xe1A6728e467ed9Fa2F2d0d7eE425a95c9d941F4d` | 10,547 | 29 `Registered`, 30 `Transfer`, 59 `MetadataSet`, 2 `URIUpdated` |
| Reputation | `0xeB49023E929bb71A4818924DffC46Eb8B46546c4` | 9,030 | 0 |
| Validation | `0x5bc80C209263Fe4F0455c76eC41F663E62D814dF` | 4,340 | 0 |

### Identities

The identity registry has minted agentIds 0 to 28 (29 identities):

- **agentId 0 and 1** were registered in blocks 111,135 and 111,136 (2026-10-13, 03:13 AEDT) by the batcher address `0x9a1e…09be`. Their token URIs were later updated (`URIUpdated` in blocks 147,689 and 147,691).
- **agentId 2–27** (26 identities) were registered in block 147,644 (2026-10-18 14:38:47 AEDT) by `0x5a2c…4d9b`. This is the address that the x402 roster lists as `x402-facilitator`.
- **agentId 28** was registered in block 148,529 (18:07:42 AEDT) by a sponsor address `0x48dc…6357` and transferred to the agent's own address `0x6379…1B7c` in block 148,531 (tx `0x2caf…e5d1`). This is the first sponsored-then-transferred identity.

The Agent Ledger's aggregate document lists the same 29 registrations. It also lists 73 per-agent `registration.json` URLs, one for every agent known to the ledger. Only 29 of those agents hold an on-chain identity.

### Main agent (agentId 2)

`ownerOf(2)` returns `0x5a2c6b8592b140e85ea9dc6185b822b5be1c4d9b`. `tokenURI(2)` returns https://scdo.com.au/.well-known/agent-registration.json. That file is an ERC-8004 `registration-v1` document named "SCDO Wallet" and described as the main SCDO agent. It advertises an A2A agent card, a web discovery page, an x402 manifest (`/.well-known/x402.json`), the Agent Ledger API and an e-mail contact. It declares `x402Support: true` and `supportedTrust: ["reputation","validation"]`, and it points back to registration `eip155:5680:0xe1A6…1F4d` with agentId 2. The registration transaction is `0xcb52…5f63` (status 1, 173,173 gas).

### Relation to AgentLedger

ERC-8004 identities and AgentLedger registrations are separate. AgentLedger has 56 `AgentRegistered` events, and the identity registry has 29 tokens. The ERC-8004 identity supplies discovery, through the registration file and its service endpoints. AgentLedger supplies batch attestation and native settlement. An open pull request (Section 7) adds on-chain `ownerOf()` discovery, so that identities registered by a sponsor and then transferred (such as agentId 28) are recognised without a hand-maintained ID file. **The reputation and validation registries have recorded no feedback and no validation requests.** Declaring `supportedTrust` therefore states an intended capability, not an observed one.

## Per-bot x402 wallets and settlement

### Wallet registry and pay routes

The public registry https://scdo.com.au/x402/wallets (version 1.0.0) lists **55 wallets** on Base mainnet (`eip155:8453`) for the asset USDC (`0x8335…2913`). Each has a distinct `payTo` address. 47 are marked `live` and 8 `receive-only`, and every one has a daily spend cap of 1 USDC. Each wallet has a route `/x402/pay/<id>` that answers an unpaid request with HTTP 402 and a base64 `PAYMENT-REQUIRED` header. The payload is `x402Version: 2`, scheme `exact`, `maxTimeoutSeconds: 300`, and EIP-712 domain `USD Coin`, version `2`. The default price is 1 USDC. The `amount` query parameter sets the price; for example, `?amount=0.01` yields `amount: "10000"` in atomic units. In this paper, "pay-per-call 0.01 USDC" refers to that parameterisation, which was used in all tests below. A companion registry, `/x402/scdo/wallets` (version 1.1.0), lists 20 native-SCDO pay routes on Shard0.

### Internal roster

The treasury page https://scdo.com.au/treasury/ renders the agent roster from the wallet registry. In a headless-Chromium rendering on 2026-10-18 at about 18:52 AEDT it showed **50 agent cards**. Each card had a "USDC reserve

(Base)" address and balance and an x402 receive link. (The remaining five registry entries are not rendered as roster cards.)

### Testnet settlement (Base Sepolia)

To exercise the full x402 flow (402 challenge, EIP-3009 authorisation, facilitator settlement), a throwaway test payer `0x6e46…F1F9` paid **0.01 test USDC** (`0x036C…CF7e`, Base Sepolia, chainId 84532) to the pay route of each of **34 roster agents**, one at a time. All 34 transactions have receipt status 1. They fall in Base Sepolia blocks 48,270,147–48,270,192 (2026-10-18 18:29:42 to 18:31:12 AEDT), go to 34 distinct payees, and each carries exactly one USDC `Transfer` of 0.01. Appendix B lists them. The roster has 35 test slugs; `scdo-ceo` has no settlement in the test log.

### Mainnet status

The mainnet routes answer with valid 402 challenges, but the reserves are **not operationally funded**. Reading `balanceOf` on Base mainnet (`mainnet.base.org`) for all 55 registry addresses on 2026-10-18 at about 18:45 AEDT gave a total of **35.66 USDC**:

- 26 wallets hold a non-zero balance, mostly seed amounts of 1.00 USDC. Five game wallets hold 3.00 USDC each, and the others range from 0.56 to 1.10 USDC.
- The other 29 wallets hold 0.

Two mainnet x402 payments appear in the signed log:

- an internal 0.01 USDC payment from `agent-grokbot` to the `tax-bureau` reserve (tx `0xc703…f42e`, Base block 52,686,563, status 1);
- a 1 USDC game purchase (tx `0xc638…dce0`, Base block 52,665,293, status 1).

We do not attribute the payer of the second transaction.

## Headless Linux bookkeeping agent: scdo-agent 0.5.x

### Distribution and update

`scdo-agent` is a headless Linux bookkeeping agent for the Agent Ledger. It runs as systemd template instances (`scdo-agent@<name>`). Releases are published under https://scdo.com.au/downloads/wallet/linux/, and `SHA256SUMS` lists versions 0.5.0, 0.5.1, 0.5.3 and 0.5.5 as `.tar.gz` and `.deb` packages for amd64 and arm64.

The release feed `latest.json` (current version 0.5.5) carries the SHA-256, size and URL of each package and of the installer. It also carries a detached 64-byte `signature` field (base64). The feed states that 0.5.5 was built by GitHub Actions CI run 39104501750 at commit `83984fe`. It also states that a separate CI run and both ubuntu-22.04 and ubuntu-24.04 produced identical files. We have not independently reproduced the build.

The installer (`install.sh`, SHA-256 `8312…1473`) supports a **rootless** mode, `sh install.sh --user`. This mode installs into `~/.local/bin` and `~/.local/share/scdo-agent` and enables systemd *user* units: `scdo-agent@.service`, `scdo-agent-update.service`, `scdo-agent-update.timer` and `scdo-agent-boot.service`. The daily timer "checks the signed release feed", and `upgrade-now` runs the same check on demand without root. Agents are never created implicitly: a graphics adapter without an agent only prints a hint.

### Ledger features

The release notes record the following features across versions:

- **0.5.0:** bot-wallet commands (`wallet create/show/limit/freeze/unfreeze/export`, `pay`, `serve-paywall`) built on the open-source x402 SDK, with per-payment and 24-hour caps, and optional ERC-8004 registration on Shard0.
- **0.5.3:** e-mail binding, x402 paywall income booking, and the rootless user unit.
- **0.5.5:** `operator batch`, which prints the batch's Merkle root and the ledger receipt, and caps stored correctly in atomic units.

The 0.5.5 notes state that Base *mainnet* stays off in the agent, so agent wallets operate on Base Sepolia only.

All agent-to-ledger requests are signed JSON envelopes. The signed message is `"SCDO Agent Ledger v1\n" + canonicalJson(payload)`, where the payload carries `chainId 5680`, `type`, `agent`, `seq` and `ts`. The ledger service (version `2.0.0 (ledger v2)`, books `2.1.0`) accepts twelve signed types, including `register`, `heartbeat`, `bind`,

emailconfirm, spend and batch. It keeps per-agent books in WSCDO, USDC and tUSDC, with opening, movement and closing balances per day (/health/daily). The free tier allows 10 new agents per IP per day and 120 signed requests per minute.

### Observed use

At the snapshot, six ledger agents report version 0.5.5 on the linux platform: two CPU light nodes and one GPU adapter on each of two hosts (".50" and ".90"). Five of the six were online. The ledger shows two operators with three and two confirmed agents (bot IDs t053-b1…b3, t055-b1…b2), consistent with operator batch registration. The batch Merkle roots are not exposed by the public API, so we do not cite them.

Of the 73 ledger agents, none has a verified e-mail binding (three have e-mail commitments). The e-mail bind feature is implemented but not yet used in production.

**Income booking example.** Base Sepolia tx 0x167e3140281b489187ec5e0f18d113a9f87c9344d9e59f878d5145f8c011bb4c (block 48,267,578, 2026-10-18 17:04:04 AEDT, status 1) transfers 0.01 tUSDC from 0x8fB3…A1A3 to 0x5271…e558. The ledger books it for agent 0xdB82…b3c1 as entry mvdf71jr-76e45e28: kind usdc_income, direction credit, asset tUSDC, amount 10000 atomic, memo "x402 Base Sepolia USDC (testnet)", confirmed: true. The agent's tUSDC balance is 0.010000. Its counterpart is the matching usdc_spend debit entries on payer agents, for example tx 0x456c…98ee for agent 0x33ad…a667.

## Proof-of-GPU (implemented, not deployed)

### Motivation

On 2026-10-18, one of the Linux GPU-adapter agents (linux-90-gpu0, ledger address 0x91A1…3AEd, version 0.5.5) was found to run on a host with an Intel UHD 610 integrated adapter and a Celeron G4930T CPU, with no NVIDIA or ROCm device. Its GPU status was self-declared. Nothing in ledger v2 could distinguish it from a real GPU agent. Any reward or income tied to GPU work could therefore be claimed by an agent without the hardware.

### Design

An agent counts as a verified GPU agent only if it passes both of two independent checks. Neither alone is enough.

1. **Signed GPU attestation.** The agent reads uuid, name, memory.total, pci.bus_id through nvidia-smi or NVML. It signs a canonical message bound to its agent address and a ledger nonce (to prevent replay) and posts it to /gpu/attest. Each UUID can be bound to only one agent. A UUID already bound to another address marks the newcomer as flagged. A UUID can be spoofed, so it never verifies on its own. An integrated or lspci-only adapter has no NVML UUID and so can never attest.
2. **Timed ethash challenge.** At registration, and then at random intervals of a few hours, the ledger issues {id, epoch, header, target, issuedAt, deadlineMs, gpuClass}. The agent must return {nonce, mixHash} with ethash(header, nonce) ≤ target before the deadline. The ledger checks the answer with a single ethash-light evaluation, which is cheap. Ethash is the algorithm Shard0 itself uses. It is memory-hard, so a GPU solves it much faster than a CPU, while the ledger verifies with one light evaluation. Difficulty and deadline are set per GPU class (LEDGER_GPU_CLASSES). The floor must stay above what the .90 host's CPU can brute-force within the deadline. A weak GPU that cannot clear that floor stays unverified; the bar is not lowered.

**State machine.** An agent that declares a GPU starts as unverified, and so does every legacy agent. It becomes verified only with a fresh, unique attestation and a challenge solved in time. It reverts to unverified when its verification lapses (TTL) or it fails a challenge, and becomes flagged on a duplicate UUID. Unverified and flagged GPU agents are excluded from GPU income and rewards (HTTP 403 gpu_unverified). CPU light agents make no GPU claim and are never gated.

### Implementation status

The ledger side is implemented in **draft pull request SCDOLAB/agent-ledger-v2#1**, "Proof-of-GPU + Shard0 WSCDO income + on-chain ERC-8004 discovery". The PR was opened 2026-10-18 at 18:29 AEDT, is not merged, and changes 11 files (+519/−6). It also routes Shard0 WSCDO x402 income to a Shard0 verifier and adds on-chain ERC-8004 discovery.

The PR reports a sandbox boot on a test port and data directory with synthetic signed agents:

- a CPU agent is unaffected and its reward is booked;
- a fake GPU (Intel UHD 610) is `unverified`, and its reward is refused with 403;
- a real-GPU path with a mock verifier goes from attest to solve to `verified`, and its reward is booked;
- a duplicate UUID across two agents is `flagged`, and its reward is refused with 403.

The safe default is that, with no challenge verifier configured, every GPU agent stays unverified. The agent side (attestation client, GPU solver and an ethash-light verifier helper) is planned for scdo-agent 0.5.6, which has not been published; the current feed is 0.5.5. **Proof-of-GPU is therefore a design with an implementation under review. It is not deployed on the live ledger, and no GPU agent has been verified by it.**

## Attribution and self-audit log

The signed task log has become the operator's attribution and self-audit record. As of `/agents/api/state` (generated 2026-10-18 18:40 AEDT):

- **136 signed AgentMessage records** (IDs 0–135) from **28 distinct agent addresses**. All 136 have `signatureVerified: true` and an anchoring transaction.
- **51 batches** anchored by the batcher `0x9a1e…09be` (batchIds 0 and 9–58), covering exactly 136 messages. They run from block 41,127 (2026-10-02 19:17 AEDT, tx `0x2251…9185`) to block 147,489 (2026-10-18 14:01:38 AEDT, tx `0x0d4e…1b73`, 133,334 gas). Batch sizes range from 1 to 14 messages, and 32 batches hold a single message.
- On-chain, AgentLedger has emitted **59** `RootAnchored` events. The 8 not in the index (batchIds 1–8, blocks 61,708–62,248, 2026-10-05) were anchored by the SCDO GPU Share deployment `0x9036…ee0b`. They include the v1 end-to-end run (batch 1, tx `0x7380…1f0c`).

By task label, the records include:

| Category | Records |
|---|---|
| Daily game summaries | 51 |
| x402 receipts, sales, repricing and wallet records | 40 |
| Wheel game health and records | 20 |
| Treasury daily closes | 6 |
| Pirate Cat tip records | 5 |
| Rent accruals | 3 |
| Hosting accrual | 1 |
| Original department deliverables (2026-10-02) | 7 |
| Go-live check, interaction log, code-fix attestation | 3 |
| **Total** | **136** |

The self-audit records state the agents' own liabilities and receipts. Examples are a rent accrual "accrued, unpaid", a hosting cost accrual with its markup, and a treasury daily close with its income figure, each with a link to the underlying JSON artefact whose SHA-256 is the `resultHash`. Anchoring makes later rewriting of these statements detectable. It does not make them correct.

## Live statistics (snapshot 2026-10-18, about 18:40 AEDT)

Table 4: Live statistics. All values were read on 2026-10-18 between about 18:40 and 18:55 AEDT.

| Metric | Value | Source |
|---|---|---|
| Shard0 head block | 148,679 (18:40:56 AEDT) | `eth_blockNumber` |
| AgentLedger `AgentRegistered` / `RootAnchored` / `TaskPaid` / `CheckpointAnchored` | 56 / 59 / 9 / 0 | `eth_getLogs` on `0x377a…a2aa` |
| `TaskPaid` receipts | 7 on 2026-10-02 (42 units, v1) + 2 on 2026-10-15 (2 units each, blocks 130,049–130,050) | `/agents/api/state` |
| Signed records / indexed batches | 136 / 51 | `/agents/api/state` |
| ERC-8004 identities | 29 (agentId 0–28) | identity registry logs |
| Ledger v2 bookkeeping agents (total / online) | 73 / 30 | `/agent-ledger/api/agents` |
| Platforms | web 19, win32 16, other 16, node 8, linux 7, android 5, darwin 1, operator 1 | same |
| scdo-agent 0.5.5 instances | 6 (5 online) | same |
| x402 wallets (Base) live / receive-only | 47 / 8 | `/x402/wallets` |

| Metric | Value | Source |
|---|---|---|
| Treasury roster cards | 50 | `/treasury/` (rendered) |
| Base Sepolia roster settlements | 34 × 0.01 tUSDC | Appendix B |
| Base mainnet USDC across 55 reserves | 35.66 (26 non-zero) | `balanceOf`, `mainnet.base.org` |
| Verified e-mail binds | 0 | `/agent-ledger/api/agents` |

The v1 measurements (block intervals, gas, time to receipt; v1 Figs. 2–5 and Table III) are unchanged and are not repeated here.

## Correction to Version 1

v1 Table IV listed four transactions from the 2026-10-05 end-to-end run, two of them as "payTask (workerA)" and "payTask (workerB)". Re-reading the receipts shows:

- `registerAgent` (`0x8bdf…4d8e`, block 61,707) and `anchorRoot` (`0x7380…1f0c`, block 61,708) are AgentLedger calls. They emit `AgentRegistered` and `RootAnchored` (batchId 1).
- `0xd098…4167` (block 61,709) and `0x8415…e0a3` (block 61,710) are **plain native transfers of 0.01 SCDO** to `0x306f…42e0` and `0x3ee3…6a1f`. They emit no `TaskPaid` event and do not call AgentLedger.

The statement in v1 that "two workers were settled with payTask" is therefore incorrect. The two workers were paid by direct transfer. All `TaskPaid` events to date are listed in Section 9. v1 also gave the small settlement tier as 1 unit. Two of the 2026-10-02 receipts were indeed 1 unit, but the current price list uses 2.

## Limitations and ethics

- **No external agents yet.** As far as we can tell, all 73 ledger agents, all 29 ERC-8004 identities and every payer and payee in this paper are operated by SCDO Laboratory, its test harnesses or its own devices. We have not observed an independently operated agent joining. The system's openness is a property of the design, not yet an observed fact.
- **Mainnet reserves not operationally funded.** The Base mainnet USDC reserves hold 35.66 USDC in total, as nominal seed amounts. The agent software keeps Base mainnet off. All substantive settlement tests were on Base Sepolia test USDC, which has no value.
- **Trust registries unused.** The ERC-8004 reputation and validation registries have no events, so no reputation or validation claim is made.
- **Proof-of-GPU undeployed.** The live ledger still relies on GPU status declared by the agent itself. Even once deployed, the scheme cannot tell a rented or remote GPU from a local one. It depends on difficulty tuning that has not yet been measured on the weakest supported GPU class, and a UUID attestation can be spoofed (which is why the challenge is also required).
- **Single operator and key custody.** As in v1, agent keys are held by one operator, and the batcher and the x402 facilitator are single points of trust. The facilitator can delay or withhold settlement but cannot forge payer signatures.
- **Partial index.** The public index omits 8 of the 59 on-chain batches (Section 8). A verifier should enumerate `RootAnchored` events directly.
- **Probabilistic finality and low hashrate.** These are unchanged from v1. No confirmation depth protects against a majority-hashrate adversary on Shard0.
- **What anchoring proves.** An anchored hash shows only that a byte string existed by the anchoring block. It does not show that agent output, accruals or treasury figures are correct. Humans set the agents' objectives and remain responsible for the outcomes.
- **Value.** The settlement amounts in SCDO are internal bookkeeping values. The operator states that SCDO tokens have no promised market value.

## Conclusion and future work

Since v1, the SCDO Agent Ledger has gained:

- standard ERC-8004 identities on the same PoW shard (29 identities, main agent agentId 2);
- x402 pay routes and USDC wallets for a 50-agent internal roster, exercised with 34 testnet settlements;
- a rootless, self-updating Linux bookkeeping agent that books x402 income into signed per-agent books;
- a self-audit log of 136 signed and anchored records.

A proof-of-GPU scheme has been designed and implemented to close a Sybil gap that was actually observed, but it awaits deployment.

Future work:

- deploying proof-of-GPU and measuring difficulty by GPU class;
- the first use of the reputation and validation registries;
- onboarding independently operated agents;
- funding mainnet reserves under explicit caps;
- hardware-backed key custody;
- running the checkpointed coordination chain.

## Data and artefact availability

All on-chain values can be reproduced from the public RPC endpoints https://scdoscan.io/rpc/0 (SCDO Shard0), https://sepolia.base.org and https://mainnet.base.org. The server-side values come from the following public endpoints:

- https://scdo.com.au/agents/api/state
- https://scdo.com.au/agent-ledger/api/agents
- https://scdo.com.au/agent-ledger/api/erc8004
- https://scdo.com.au/agent-ledger/api/agents/%3Caddress/ledger>
- https://scdo.com.au/x402/wallets
- https://scdo.com.au/treasury/usdc-balances.json
- https://scdo.com.au/.well-known/agent-registration.json
- https://scdo.com.au/downloads/wallet/linux/latest.json

The proof-of-GPU pull request is in the SCDOLAB GitHub organisation (`SCDOLAB/agent-ledger-v2#1`). That repository is private at the time of writing.

## Appendix A: Full addresses and hashes

| Item | Value |
|---|---|
| AgentLedger v0.1 contract | `0x377aee324323237c11469c0673b39ab30753a2aa` |
| AgentLedger deploy tx | `0xd5cca0498b41e8f3971b40fd94c17441870bba1610d5cd58335977e35fcfd44e` |
| Batcher | `0x9a1ec1ec80c6bbded59848bf2eb33613ad4f09be` |
| ERC-8004 Identity | `0xe1A6728e467ed9Fa2F2d0d7eE425a95c9d941F4d` |
| ERC-8004 Reputation | `0xeB49023E929bb71A4818924DffC46Eb8B46546c4` |
| ERC-8004 Validation | `0x5bc80C209263Fe4F0455c76eC41F663E62D814dF` |
| agentId 2 owner / registrant (x402-facilitator) | `0x5a2c6b8592b140e85ea9dc6185b822b5be1c4d9b` |
| agentId 2 registration tx | `0xcb52a268fe69b2e0769ff14018f5cfabd2f6003bb5a2d3f89bbf960b64c35f63` |
| agentId 0 registration tx | `0x15df2bce84a2f5887a675c00c744736df72fe026d57bc098d8aef559d7b26b3d` |
| agentId 28 registration tx | `0xf0847d3ed455f0bb5a35bbddeef0fffe522c65c3c0a5f9dfe3f605b728d84da0` |
| agentId 28 sponsor | `0x48dc8ec5a153641e4a6e5f1fc4452f82452f6357` |
| agentId 28 owner | `0x6379C88ef7E28ebefC91fC53A91B2C7bcc101B7c` |
| agentId 28 transfer tx | `0x2caf576ae2eaec14e3dce473e6e6ab88bcb39f2c901b8ee18a0afa9230a3e5d1` |
| Base USDC (mainnet) | `0x833589fCD6eDb6E08f4c7C32D4f71b54bdA02913` |
| Base Sepolia test USDC | `0x036CbD53842c5426634e7929541eC2318f3dCF7e` |
| Mainnet x402 internal payment tx | `0xc703de881b4151052e1ae9c37bbcc8a69479eb564404fbdb483b4dd50f4ff42e` |
| Mainnet x402 game sale tx | `0xc63825e01269efa935c40b1f109adb0efe5f0d2b466689482f1aacf4d74fdce0` |
| Income tx (Base Sepolia) | `0x167e3140281b489187ec5e0f18d113a9f87c9344d9e59f878d5145f8c011bb4c` |
| Income payer | `0x8fB31BBAD34637931ff7F4B5bDF76EAbAbcEA1A3` |
| Income payTo | `0x52712e4689512b10367b53f1944ae22070bee558` |
| Income-booking agent | `0xdB82e92612f775aC29D5D5358d357E510a82b3c1` |
| Spend example tx (Base Sepolia) | `0x456c2bbe77858a466f3389c6422f2281b55dd1b716f26ed7bdb894b093f898ee` |
| Spend example agent | `0x33ad9D01340C359bCA1a0b67159a2932b21Aa667` |
| Fake-GPU agent (linux-90-gpu0) | `0x91A1A8C425b97566C2FD65B02ef4B119Fefb3AEd` |
| Batch 0 anchor tx | `0x225147dfc4c93ddf206ce3e24b44f19a24629af565fc7b55963ae995819d9185` |
| Batch 58 anchor tx | `0x0d4e7a17b8827f053ea6208499b9463d80a4ce73357b844f661c10b3a0001b73` |
| SCDO GPU Share anchorer | `0x9036d91ed51be47bdb85999d5d96107b66a2ee0b` |
| 2026-10-05 registerAgent | `0x8bdf6badb4a89129671ef3c69b95dea50a7f66c17ee811af3c0464d93c954d8e` |
| 2026-10-05 anchorRoot (batch 1) | `0x73808b87e80df3f4ca296615141597ce3f81db5592eb19ec38b093b014191f0c` |
| 2026-10-05 transfer to worker A | `0xd0989cd136f148103f02b7f71df41004d49a3be7fd2c8c02f4035277eb5d4167` |
| 2026-10-05 transfer to worker B | `0x841534aa41c7411520f2e2e4c923fda50c2ba2eb15a944aa808479606f3de0a3` |
| Worker A | `0x306f8344321bc2b7d63e96a935223a4c631442e0` |

| Item | Value |
|---|---|
| Worker B | `0x3ee3b578c2035a0248fd0b5be828b5dc5a396a1f` |
| TaskPaid 2026-10-15 (1) | `0xd6e1b165ea14782236aa26958cb75f788a5a6f8381b40ff85329404fc019d7a4` |
| TaskPaid 2026-10-15 (2) | `0xb737d5be6ed034d1a2aa5389e16f606bbc95b153ec651874f798dc120362f5d6` |
| installer SHA-256 | `831284c8e7496172802410f1b2cc83a67b608d1a6a6b15e5142e121e02ac1473` |

# Appendix B: Base Sepolia roster settlements (2026-10-18, AEDT)

Payer `0x6e46662185c8D4De8FA4450f7880bAC5C453F1F9`; asset test USDC `0x036CbD53842c5426634e7929541eC2318f3dCF7e`; amount 0.01 (10000 atomic) each; all receipts status 1.

```
 1  scdo-wallet-bot        block 48,270,147  18:29:42 AEDT
    payTo 0xc276e8c81d5e1259ee5acfc510a9e84319ad292a
    tx    0x0efbd48f6bf52ed3b765fa4fd3b1b533ad4ee20e184984793627aea711b6ada9
 2  scdo-games-ledger-bot  block 48,270,148  18:29:44 AEDT
    payTo 0x949a3bc6a5d6318349742af41bedc2746a830785
    tx    0x08aed7cd1d71d30181e3c4fda4c80e7e94ddc2842fb2bb0f239bd3ec517be143
 3  scdo-help-bot          block 48,270,149  18:29:46 AEDT
    payTo 0x71694f668a380880b65789d5936ae24764cd19cf
    tx    0x7658b0b9b4b362b9e2a8ae2e8f4c8650b37f7c1852f914552eec0f64307568bd
 4  pirate-cat             block 48,270,151  18:29:50 AEDT
    payTo 0x5b5f291722777c47c9bbb66b5ae05a8b455cfe21
    tx    0xbfe6580005ed3e6da07f7270af0eb7f0cf63b08f411af543066567a78f564832
 5  kitten-boarding-party  block 48,270,152  18:29:52 AEDT
    payTo 0x39042fa28c909671ccbebd89bdade425ab9134d0
    tx    0xcf2c9b64c653c93816a80473b8f770bb94f18ffc45ff3b15caf6b8f657766408
 6  kitten-circle          block 48,270,154  18:29:56 AEDT
    payTo 0x1a67d9d66c256db5eeee3498efe1df34f886faf6
    tx    0xa48e5de65b092bcb0a54ab994040fa8df45f64aac25676a9cc2a6d6c8497dcdb
 7  kitten-devil           block 48,270,155  18:29:58 AEDT
    payTo 0xab079e6530c19b0edd0d53f02ff74f4f559b2da0
    tx    0x8a626003615ee9f57786f2608410cc0bcba791d6c7bdfbd3dcfa7142d61c29af
 8  kitten-drive           block 48,270,157  18:30:02 AEDT
    payTo 0xa170858eae6d756b961a5582bd3411855ee5010e
    tx    0xee62a236aecffe3c2d0d8c6e2a1b303c360f2ca861ddbeedf8e24444451d55ab
 9  kitten-ferry           block 48,270,158  18:30:04 AEDT
    payTo 0x55ab0341b210fc71e9dd30c828c796f8c48ad40f
    tx    0x63da1c9c554ed97573b9cc69af751ba36ce3300a63b4198eeded6a2b51dcd456
10  kitten-merge           block 48,270,159  18:30:06 AEDT
    payTo 0xcb438b56d81f7e37e426e68db45d8c6bb1a3fe86
    tx    0xc1e354d5f36220686cf93c712a63a733c10b2ceaf940119ab7a886805b5ce430
11  kitten-stack-tower     block 48,270,161  18:30:10 AEDT
    payTo 0x709d598c93548da098f1d593851b7ccc39292b09
    tx    0xb0ad085ab46bb8ccda1cb2efa9022b877ea7ceaaf14ad3ec595a8fb448b24c25
12  mast-drop              block 48,270,162  18:30:12 AEDT
    payTo 0x9359a3289be2d39cb45e76bc6009e718de38c888
    tx    0x9a4a2bd0a823265a82232dfc02b1af938e68abac1b457cac4da273c925015c29
13  paw-crate-clear        block 48,270,163  18:30:14 AEDT
    payTo 0xa5092207ce48f13d8090df4d7aacaff071f1ec1e
    tx    0x923e8e4bf62fad33d837b4dac01647ce9a97d6b6c5a1aa8bfd2a99478f79e481
14  rum-bottle-sort        block 48,270,165  18:30:18 AEDT
    payTo 0xc3ff9b52b29816cfbe3b5184c1a2775afde4656f
    tx    0x6050d2a83df2a5a25d4011557fc2ef196af533dde3eaa0fa2838dcf750b6c9f0
15  shard-miner            block 48,270,166  18:30:20 AEDT
    payTo 0x90d19e1349694f854c32e534de861a84475dd2ea
    tx    0x5fa4b24d599281283786e0484707571089b99e53958b0054808bd8dfc18df991
16  shard-miner-real       block 48,270,167  18:30:22 AEDT
    payTo 0xc111a9e8a0eb7e037a30bb9ec31cdcb28916e2a8
    tx    0xdb4a37c40a9766a289e685a3549f6322995826c87b84b47f898f32291cf63ec4
17  treasure-trio          block 48,270,169  18:30:26 AEDT
    payTo 0x7bd26122d727507a8f2ac435dd8ac37928651da4
    tx    0xddb476493dd5ad00155e75c1d4862222b29df76e38a99698e36f58aa08dd8d51
18  paw-doku               block 48,270,170  18:30:28 AEDT
    payTo 0x61c7d8f168be5824872997ae7f31270a456455a2
    tx    0x16eca459d1e5b1ea951738e6cc202ef63b0e4ad15729a9d62a7765e2af1e5e54
19  android-pixel7-9138    block 48,270,171  18:30:30 AEDT
    payTo 0x82250e41973ec24ace1c06d7c31170ec9afc6c57
    tx    0x21b57a06e4c053f1a29217f0b70145aa8254de721b76a266af233a8f2a043951
20  android-debug-1644     block 48,270,173  18:30:34 AEDT
    payTo 0x3a7e195f10a4e59c0289291c06f4064b2c8de694
    tx    0x728fa71429b91a20c398926c50d3a2ded77c3b466882797392fef78a455455f9
21  android-debug-2bc0     block 48,270,174  18:30:36 AEDT
```

```
    payTo 0x551c7df094bf82c471cf4f2c8684f331cf6c54ef
    tx    0x1a95fed88a5575d6ce0a05f75f2848dbe6790139e364a71d56c5e6f9ae365a07
22  gpu-share             block 48,270,176  18:30:40 AEDT
    payTo 0x30b0f327deef5187c11c1502ecd467b47c5ea2e5
    tx    0xcd3e9639a9d43ac16d96fa7424e3375c377fcb8b2c984e90b933714c2e350aa6
23  scdo-mac              block 48,270,177  18:30:42 AEDT
    payTo 0x7e14a7b9e290ba253e4be60f16e2b8f75a749250
    tx    0xccf39a63d48aeae70a0f882616f4e8bfd4e34db242c6440b445e5be5caee8b04
24  gpu-worker-1          block 48,270,179  18:30:46 AEDT
    payTo 0x9380f1fca499284dd86ef273ad7523e7b81faade
    tx    0xab8f29552a50788e5581a479ca1fa40fd3e0a9ec04f24ce987d334ad520d18a6
25  gpu-worker-2          block 48,270,180  18:30:48 AEDT
    payTo 0x9e32313279da28803d99ac3cc1bf0d88e2957876
    tx    0x2a729ad2f2da4537924002a41ec0037a78f4b9af8827ac8de09e5ed972f33749
26  rig-3080              block 48,270,181  18:30:50 AEDT
    payTo 0xc796a6c0ea05126ce94c8757ff9cebf69e8db9f7
    tx    0xb8ccd32b771c74ea014ad9a4f41e5b626598a9cb87d69cee965b85abaeccc461
27  linux-50-a1           block 48,270,183  18:30:54 AEDT
    payTo 0xe750146dffc722f231583de628696f05d694f5ed
    tx    0xccdfe6f407bebe55a4f0d5d0e6c0c5fb33ef59222aea17d36938766a29a48c9b
28  linux-50-a2           block 48,270,184  18:30:56 AEDT
    payTo 0xb77552b711e8827653eac3d67c9e2ecb6cb39182
    tx    0x6f084c4e0d91a1d299abde74c5457e28d3675c04ffefaeb71d556cdbfe4d791a
29  linux-50-gpu0         block 48,270,185  18:30:58 AEDT
    payTo 0xd560cabaa6db8271ea2feb1de0ad81603de1945c
    tx    0x65feea3e48710609349d203f59be8753459b859f2e0f226e18502ea70f31f7dc
30  linux-90-a1           block 48,270,187  18:31:02 AEDT
    payTo 0xd9ce6624a5bc13620ae3ad883748a7dedfeb7bec
    tx    0x868f139b0e6becb2c1b02b7e5763b6228e6807d69fb866e8ac9fd59487aeeb7e
31  linux-90-a2           block 48,270,188  18:31:04 AEDT
    payTo 0xa340d959dd74f65c5c44bad5f281021dc06054e0
    tx    0x59875b77747ba8e512f3b7518f8f816262632eef956dd85e61e534f944ba86e8
32  linux-90-gpu0         block 48,270,189  18:31:06 AEDT
    payTo 0x91a1a8c425b97566c2fd65b02ef4b119fefb3aed
    tx    0xa721312b8d21f372c7065693295b7f3797d265e99ac15a947210f22cf7a023e2
33  x402-facilitator      block 48,270,190  18:31:08 AEDT
    payTo 0x5a2c6b8592b140e85ea9dc6185b822b5be1c4d9b
    tx    0x5b37dc1501ba37c64b9ff6e372857c14af4b8a184a00f916346032035fe5ad45
34  melbourne-server      block 48,270,192  18:31:12 AEDT
    payTo 0x49900274f1c2bd38b1bff60a5d28471a588c4533
    tx    0x66e6e8b0c6f62cc177adf25edd26f0edc67e40ed9c7368af0a1c270e8082e3fb
```

# References


[1] L. Wang et al., “A survey on large language model based autonomous agents,” *Frontiers of Computer Science*, vol. 18, no. 6, 186345, 2024. doi:10.1007/s11704-024-40231-1

[2] J. S. Park et al., “Generative agents: Interactive simulacra of human behavior,” in *Proc. UIST*, ACM, 2023, pp. 1–22. doi:10.1145/3586183.3606763

[3] Fetch.ai, “uAgents,” software repository, 2026. https://github.com/fetchai/uAgents

[4] Fetch.ai, “The Almanac,” developer documentation, 2026. https://network.fetch.ai/docs/introduction/almanac/introduction

[5] Autonolas, “Autonolas whitepaper, version 1.0,” 2023. https://olas.network/documents/whitepaper/Whitepaper%20v1.0.pdf

[6] M. De Rossi, D. Crapis, J. Ellis and E. Reppel, “ERC-8004: Trustless agents,” Ethereum Improvement Proposals, Draft, Aug. 2025. https://eips.ethereum.org/EIPS/eip-8004

[7] Alibaba Cloud, “Blockchain as a service: Transaction models for attestation,” 2023. https://www.alibabacloud.com/help/en/blockchain-as-a-service/latest/transaction-models-for-attestation

[8] B. Summerwill and A. Tsankov, “ECIP-1049: Change the ETC proof of work algorithm to Keccak256,” Withdrawn, 2019. https://ecips.ethereumclassic.org/ECIPs/ecip-1049

[9] S. Nakamoto, “Bitcoin: A peer-to-peer electronic cash system,” 2008. https://bitcoin.org/bitcoin.pdf

[10] G. Wood, “Ethereum: A secure decentralised generalised transaction ledger,” Yellow Paper, 2025. https://ethereum.github.io/yellowpaper/paper.pdf

[11] ethereum.org, “Ethash,” developer documentation, 2026. https://ethereum.org/developers/docs/consensus-mechanisms/pow/mining/mining-algorithms/ethash/

[12] ETC Labs Core, “CoreGeth documentation,” 2026. https://etclabscore.github.io/core-geth/

[13] V. Buterin, “EIP-155: Simple replay attack protection,” 2016. https://eips.ethereum.org/EIPS/eip-155

[14] L. Luu et al., “A secure sharding protocol for open blockchains,” in *Proc. ACM CCS*, 2016, pp. 17–30. doi:10.1145/2976749.2978389

[15] E. Kokoris-Kogias et al., “OmniLedger: A secure, scale-out, decentralized ledger via sharding,” in *Proc. IEEE S&P*, 2018, pp. 583–598. doi:10.1109/SP.2018.000-5

[16] D. Minarsch et al., “Autonomous economic agent framework,” in *EMAS 2021*, LNCS 13190, Springer, 2022, pp. 237–253. doi:10.1007/978-3-030-97457-2_14

[17] Y. Rao et al., “BitTensor: A peer-to-peer intelligence market,” 2020. https://arxiv.org/abs/2003.03917

[18] Coinbase Developer Platform, “x402: An open standard for internet-native payments,” whitepaper, 2025. https://www.x402.org/x402-whitepaper.pdf

[19] x402 Foundation, “x402 specification: HTTP transport (v2),” 2025. https://github.com/x402-foundation/x402/blob/main/specs/transports-v2/http.md

[20] E. Androulaki et al., “Hyperledger Fabric: A distributed operating system for permissioned blockchains,” in *Proc. EuroSys*, ACM, 2018. doi:10.1145/3190508.3190538

[21] B. Laurie, A. Langley and E. Kasper, “Certificate transparency,” RFC 6962, IETF, 2013. https://www.rfc-editor.org/rfc/rfc6962

[22] S. A. Crosby and D. S. Wallach, “Efficient data structures for tamper-evident logging,” in *Proc. USENIX Security*, 2009.

[23] Ant Digital Technologies, “AntChain trusted notarization service,” product page (in Chinese), 2026. https://antchain.antgroup.com/products/twc

[24] P. Zheng et al., “Meepo: Sharded consortium blockchain,” in *Proc. IEEE ICDE*, 2021, pp. 1847–1852. doi:10.1109/ICDE51399.2021.00165

[25] Z. Chen et al., “SChain: A scalable consortium blockchain exploiting intra- and inter-block concurrency,” *PVLDB*, vol. 14, no. 12, pp. 2799–2802, 2021.

[26] S. Liu et al., “Flexible advancement in asynchronous BFT consensus,” in *Proc. ACM SOSP*, 2023, pp. 264–280. doi:10.1145/3600006.3613164

[27] SCDOLAB, “Proof-of-GPU + Shard0 WSCDO income + on-chain ERC-8004 discovery,” GitHub pull request agent-ledger-v2#1 (draft, unmerged), 2026-10-18.