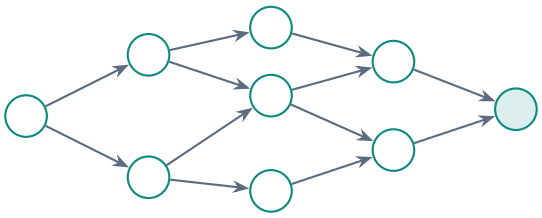
ZKLOOP LABS — TECHNOLOGY WHITEPAPER

Proof of Balance

Private, Scalable Payment Infrastructure for People, Businesses, and AI Agents



A plain-language introduction to the PoB payment platform: deterministic settlement, parallel execution, encrypted balances, programmable escrow, and payment rails built for autonomous agents.

zkLoop Labs ▪ info@zkloop.net ▪ July 2026 ▪ Draft for review

Contents

1 Executive Summary

Most blockchains are poor payment rails in the ordinary, merchant-counter sense of the word. They are slow at the checkout boundary, because finality is probabilistic and arrives block by block. They are hard to scale, because execution is usually serial and every participant must replay every transaction. And they are too public for businesses that do not want competitors reading every sale, every price, and every customer relationship off a shared ledger.

Card networks, on the other side, are fast and familiar — but they are built as universal, neutral, consumer-facing networks. Entire categories of legitimate business (metered services, stored-value programs, shared and delegated spending, operator-run economies) are structurally mismatched with how card networks must operate.

zkLoop Labs builds **Proof of Balance (PoB)**: payment infrastructure designed to close both gaps at once. The core ideas are simple to state:

- **Prove, don't publish.** Balances are stored as encrypted commitments. Every payment carries a zero-knowledge proof that value is conserved and no balance goes negative — without revealing any balance or linking any account.
- **Settle deterministically, not probabilistically.** A payment is final the moment a quorum of independent validators certifies its proof — typically two network round trips. There are no confirmations to wait for and no risk of reversal by a chain reorganization.
- **Order less, so you can parallelize more.** The ledger is a graph of balances, not a single line of blocks. Payments that touch different balances do not compete with each other, so throughput grows with hardware and quorum capacity instead of being capped by a global block size.
- **Keep the familiar surfaces — without public exposure.** The platform exposes a standard EVM-compatible chain (Solidity contracts, JSON-RPC, ordinary web3 tooling), a REST API for merchants who never want to see a blockchain, and an MCP tool interface for AI agents. Every chain is configurable to run public or *private*: EVM-compatible does not mean publicly readable.
- **No gas, ever.** Users and agents never buy, estimate, or budget a gas token. Abuse is prevented by a zero-knowledge minimum-balance check instead, and fees are ordinary business configuration: each merchant contract sets its own transaction-fee percentage for the operator and optional rewards back to the customer.
- **Assume your own components can misbehave.** The execution engine is treated as untrusted by design: the validator quorum independently re-executes every transaction and anchors it to quorum-certified state, so even a compromised engine cannot forge history or counterfeit value.

The same properties that make PoB a good fit for human checkout — deterministic sub-second settlement, enforced spending rules, privacy for the merchant's book of business — are exactly what autonomous AI agents need in a payment rail. This paper explains the technology end to end, and closes with the deployment model we consider its best first fit: **semi-closed-loop payment systems**, where an operator issues stored value that circulates among a defined set of merchants.

This paper is intentionally non-mathematical. The formal construction, with full safety and liveness proofs, is published separately as *"A Byzantine-Tolerant State Machine over a PoB Hypergraph"* (the PoB technical paper).

2 The Problem: Payments Do Not Fit Today's Rails

2.1 Where public blockchains fall short

A payment at a counter — physical or digital — has three non-negotiable requirements: the answer must come back in about a second, the answer must be final, and the transaction must not become public knowledge. General-purpose blockchains miss all three, for structural reasons rather than engineering immaturity:

- **Latency and finality.** Consensus-ordered chains confirm transactions in blocks, and confidence grows only as more blocks are added. "Probably final in a few minutes" is acceptable for settlement between institutions; it is unusable at checkout, and it is poison for software agents that must decide their next action *now*.
- **Scalability.** When every transaction must be placed into one globally agreed order and replayed by every node, the slowest part of the system sets the pace for all of it. Two payments in different shops on different continents still queue up behind each other.
- **Privacy.** Public ledgers are transparent by default. For a business, that means revenue, pricing, refund rates, supplier relationships, and customer behavior are readable by anyone — including competitors. Businesses do not adopt rails that publish their books.

2.2 Where card networks stop

Card networks solved latency and familiarity decades ago, but their structure imposes its own limits. A universal network must remain a neutral third party, apply uniform risk and dispute rules, see transaction data to fight fraud, and avoid taking custody of funds. That rules out businesses whose payment logic is inseparable from their business logic:

- **Unknown final amounts.** Fueling, parking, rentals, hotels, and metered services do not know the price when the transaction starts. On card rails this is emulated with preauthorizations, and normal customer behavior routinely turns into chargebacks.
- **High-frequency, low-value usage.** Pay-per-use and pay-per-view models drown in per-transaction fees, so operators fall back to prepaid credit systems, invoicing, and monthly aggregation — rebuilding a private ledger with none of the guarantees of one.
- **Shared and delegated spending.** A card handed to a family member, an employee, a contractor, or a machine carries *unlimited* authority. Real delegation needs rules — which merchants, how much, until when, for what purpose — enforced *before* money moves, not disputed after.
- **Operator-run economies.** Stored-value programs, membership economies, and jurisdiction-specific schemes need the operator itself to hold funds and define the rules. A neutral global network cannot take that role; purpose-built infrastructure can.

The gap PoB targets

Businesses in these categories need what a blockchain promises — enforceable rules, auditable settlement, no dependence on a network's permission — delivered with the latency, privacy, and cost profile of in-house infrastructure. PoB is built for exactly that intersection.

3 Proof of Balance, in Plain Terms

3.1 Sealed envelopes instead of public numbers

In PoB, an account balance is never written down in the clear. Each balance is stored as a *cryptographic commitment* — think of a sealed, tamper-evident envelope that locks in a number without showing it. The envelope cannot be opened by validators, but its contents can be *reasoned about* mathematically.

A payment takes some existing envelopes as inputs (the payer's balance), and produces new envelopes as outputs (the payee's new balance and the payer's change). Attached to every payment is a **zero-knowledge proof** — a short piece of cryptographic evidence that convinces anyone of two facts while revealing nothing else:

1. **Conservation:** the amounts inside the input envelopes equal the amounts inside the output envelopes. No value appears from or vanishes into thin air.
2. **Range:** every new envelope contains a non-negative amount. Nobody can pay with a balance they do not have.

Validators check the proof, not the balances. A PoB transaction contains no visible amounts, no account addresses, and no reusable signatures that would let an observer link one payment to the next.

3.2 A graph of balances, not a chain of blocks

The second departure from a conventional blockchain is the shape of the ledger itself. PoB's ledger — the *Chain of Balance* — is a graph: balances are the nodes, and each payment is an edge that consumes old balances and creates new ones (Figure 1). Every balance is **write-once**: it is created by exactly one payment and can be spent by at most one payment.

That single rule is the whole double-spending defense, and it does not require the network to agree on a global order of events. Independent validators each keep one promise — *never endorse the same balance being spent twice* — and every payment must collect endorsements from a quorum of them. Any two quorums overlap in honest validators, so two conflicting spends of the same balance can never both gather enough endorsements. Double-spending is prevented by *overlap*, not by *ordering*.

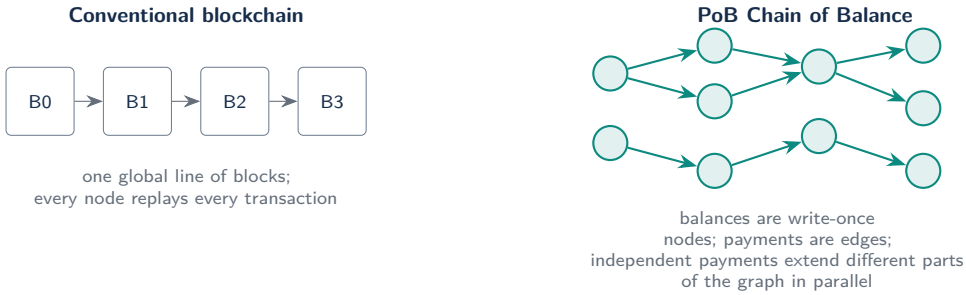


Figure 1: A single ordered chain forces global coordination. A graph of write-once balances lets unrelated payments proceed independently.

3.3 Deterministic settlement

Because there is no mining, no staking lottery, and no block interval to wait out, settlement in PoB is **deterministic**: a payment is either accepted — in which case it carries a threshold-signed certificate from the validator quorum and is immediately, irreversibly final — or it is rejected. Nothing is "probably confirmed." The commit protocol is two phases (an endorsement round and a commit round), so end-to-end settlement is a question of network round trips and milliseconds of proof verification, not minutes of block production (Figure 2).

Why this matters at the counter — and to a machine

A merchant terminal, a vending machine, or an AI agent gets a yes/no answer it can act on immediately. "Final" means final: the certificate can be stored as a receipt and verified by anyone, later, without asking

Figure 2: The life of a PoB payment. Settlement is a fixed, short pipeline — not a probabilistic process.

> the network again.

4 The Platform, End to End

PoB is not a single program but a small stack of layers, each with a clearly bounded job and a clearly bounded level of trust (Figure 3). From top to bottom:

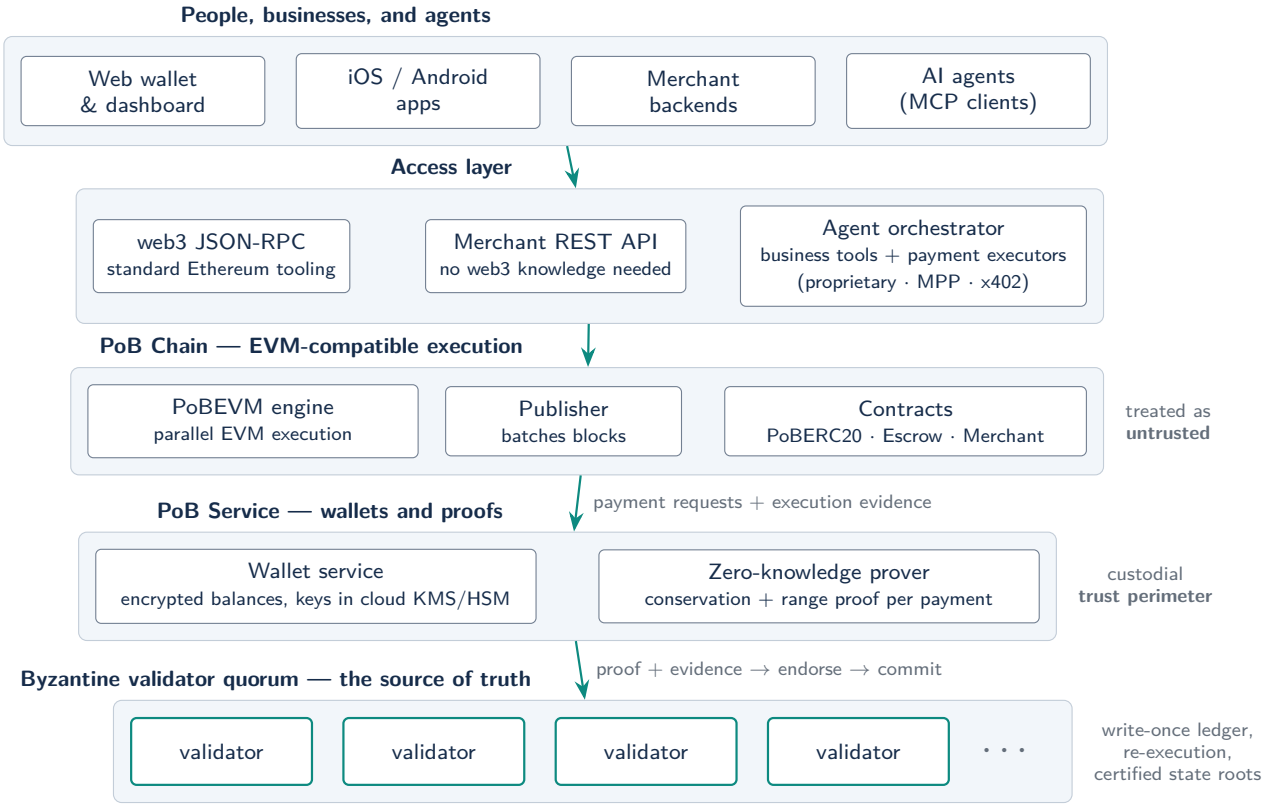


Figure 3: The PoB platform end to end. Each layer down is trusted less; the validator quorum at the bottom is the only source of truth.

4.1 Clients: web, mobile, merchants, and agents

Users hold wallets through a web application or through native iOS and Android apps. Sign-in uses **OPAQUE**, a modern password protocol in which the password itself never leaves the device — not even the operator's servers ever see it. The demo apps prove the full stack — registration, sign-in, wallet, payment — runs on

ordinary consumer phones with no special hardware.

Merchants integrate through a plain REST API: familiar operations (`hold`, `settle`, `release`) with an API key. No gas, no nonces, no key ceremony — a merchant never needs to know a blockchain is underneath. AI agents connect through an MCP tool interface described in Section 9.

4.2 PoB Chain: an EVM your tools already understand

The execution layer, **PoBEVM**, is EVM-compatible: Solidity contracts, standard JSON-RPC, and ordinary web3 clients work unmodified. Visibility is a per-chain configuration: a chain can run public or — as in business deployments — *private*, where the RPC boundary authenticates every caller and scopes what each may read (Section 6.2). Payment logic lives in a small set of audited-style contracts — a PoB-aware ERC-20 token, an escrow contract, and merchant contracts.

Under the hood, PoBEVM differs from a conventional node in one important way: it executes *non-conflicting transactions in parallel*, detecting conflicts optimistically instead of forcing all transactions through one serial lane. A publisher batches results into blocks for housekeeping, but — crucially — the chain's blocks are bookkeeping, not truth. Truth is established one layer down.

4.3 PoB Service: wallets and proofs

The wallet service holds encrypted balances and produces the zero-knowledge proof for each payment. Wallet keys are protected by cloud KMS/HSM hardware, and access is mediated by *capability tokens* — scoped credentials that can be limited by amount, time window, and permitted operations, and can be issued as one-time tokens. This is the custodial trust perimeter of the platform: the operator (or the business running its own deployment) is the custodian, which is precisely the model semi-closed-loop systems require.

4.4 The validator quorum: the source of truth

At the bottom sits a quorum of independent validators running the PoB state machine (bftsm). The quorum tolerates Byzantine faults — up to a third of its members can be compromised, offline, or actively malicious without endangering correctness. Validators keep write-once records, verify every zero-knowledge proof, independently *re-execute* every EVM transaction, and issue the threshold-signed certificates that make settlement final. Everything above this layer, including the chain engine itself, is checked by it.

5 Scalability and Latency

5.1 Throughput: remove the global queue

The deepest scalability limit of conventional blockchains is not block size or gas limits — it is the requirement of a single, globally agreed order of all transactions. PoB removes that requirement at both of its layers:

- **At the ledger**, the Chain of Balance needs no total order. Two payments that spend different balances are simply two independent extensions of the graph; validators can verify and certify them concurrently. The per-payment message cost is linear in quorum size, with no multi-round consensus among validators.
- **At the EVM**, PoBEVM executes transactions in parallel and uses optimistic concurrency control to catch the rare genuine conflicts. Payments to different merchants — the overwhelmingly common case — never contend.

The result is a system whose throughput scales with the capacity of its hardware and quorum, rather than being capped by a network-wide heartbeat. Costs stay flat as well: the chain is entirely gas-free (Section 5.3), so nothing spikes when the network is busy — and low-value payments stay economical, a precondition for

pay-per-use business models.

5.2 Latency: checkout-grade, by construction

Latency budgets in PoB are set by computation and round trips, both of which are engineering quantities rather than protocol constants:

- Generating the zero-knowledge proof for a payment takes on the order of **tens of milliseconds** on commodity hardware.
- Verification and endorsement by the quorum is **one round trip**; the commit certificate is a **second round trip**.
- There is no block interval to amortize against and no confirmation depth to accumulate.

In practice this places full, irreversible settlement within an ordinary interactive request — the same latency class as a card authorization, but with the finality of settlement rather than the beginning of a dispute window.

Determinism is a feature of its own

Beyond being fast, PoB settlement is *predictably* fast. A payment's outcome does not depend on network congestion, fee bidding, or miner behavior. For unattended systems — kiosks, machines, and AI agents — the elimination of "maybe, check again later" is worth as much as the raw speed.

5.3 A gas-free chain

Public blockchains charge gas for a good reason: without a per-operation price, anyone could flood the network with computation and grind it to a halt. But gas is also among the worst parts of the user experience — a second asset to acquire, a fluctuating price to estimate, a budget to manage — and it lands on the wrong party: the payer, or the payer's software. PoB chains use **no gas at all**. The platform separates the two jobs gas was doing and solves each directly:

- **Abuse control: prove a balance, pay nothing.** To have a transaction accepted, the sender must demonstrate — in zero knowledge — that its wallet holds at least a configured minimum balance. Nothing is deducted and no balance is revealed; the requirement simply ensures every transaction is backed by a real, funded wallet, which is what makes resource-exhaustion attacks expensive to mount.
- **Economics: fees are contract configuration, not a protocol tax.** Each merchant contract can configure a transaction-fee percentage and an optional customer-reward percentage. When a payment or an escrow settlement completes, the fee flows from the merchant to the operator and the reward flows back to the customer — automatically, inside the same atomic transaction. How much, and whether at all, is the merchant's own business decision, made per contract.

The paying side therefore never sees a network fee of any kind; the operator's revenue arrives as a share of merchant fees rather than a per-computation toll — the same shape as payment economics businesses already understand. And an entire category of friction disappears: nobody holds a special token, estimates a gas price, or has to answer "who pays the gas?" That last question matters more than it looks — for autonomous agents it is often the deciding one, as Section 9 discusses.

6 Privacy

6.1 Proof replaces disclosure

PoB's privacy posture follows one principle: the network should be able to *verify* everything while *learning* as little as possible. Concretely:

- **Balances are never published.** They exist only as encrypted commitments. Validators prove-and-check; they cannot read.
- **Payments are unlinkable at the ledger.** A PoB transaction carries no account addresses, no visible amounts, and no long-term public keys. An observer of the ledger cannot reconstruct who paid whom, for how much, or link one purchase to the next.
- **Merchant activity is not a public feed.** On a transparent chain, a merchant's entire revenue stream is competitive intelligence for anyone who cares to look. On PoB, a merchant's volume, pricing, and customer patterns are simply not on display.
- **Credentials stay private too.** OPAQUE sign-in means user passwords never leave the device; wallet keys live in KMS/HSM hardware, not in browsers or phones.

6.2 "But isn't an EVM chain transparent?" — the private chain

A careful reader will have noticed an apparent contradiction: PoB balances are encrypted, yet the platform also exposes an EVM-compatible chain — and Ethereum-style chains are famously transparent, with every account, transaction, and event readable by anyone. Would the EVM layer not leak exactly what the PoB layer protects?

It would, if the chain were public — but on this platform, that is a deployment choice, not a law of nature. Every EVM chain is configurable to run **public or private**, and the PoB value layer beneath it is exactly the same in either mode. Business deployments run private: the chain's RPC boundary then enforces an authorization policy on every call:

- **Every caller authenticates with the key it already has.** Before serving state, the endpoint issues a challenge; the client signs it with its ordinary Ethereum account key — a standard typed-data signature that every web3 wallet already supports — and receives a short-lived access token (minutes, renewed automatically). There is no new credential type and no custom client: most web3 libraries and wallets work as-is.
- **Participants see their own activity — and only their own.** An authenticated participant can send transactions and read its own transactions and receipts. It cannot enumerate other participants' transactions, pull full block contents, or scan the chain's event logs. This deliberately departs from Ethereum's everyone-reads-everything semantics — that is the point — while keeping the wire protocol and the tooling unchanged.
- **The owner has full access — because it is their chain.** The chain owner is the merchant or business entity operating the deployment, and the chain runs completely under its control. The owner sees everything on its own ledger, exactly as a business sees its own transaction records today.

The result is the right visibility for each party: participants get verifiable receipts for their own activity, the operating business gets its full books, and outsiders — including other participants — get nothing. Note that access control and correctness remain separate concerns: even this owner-controlled chain is not *trusted* — the validator quorum of Section 7 re-executes and certifies its every transaction regardless of who may read it.

6.3 Private chains without silos: one balance, across contracts and chains

Ordinarily, making a chain private buys confidentiality at a steep price: isolation. Two Ethereum-style chains are separate universes — a balance on one simply does not exist on the other, and moving value between them means bridges, wrapped assets, and a fresh set of trusted intermediaries.

PoB avoids that trade, because value never lives inside a chain in the first place. Balances live in the wallet's PoB layer — encrypted commitments certified by the validator quorum — and each EVM chain is an execution surface that spends from and settles into that shared layer. Three consequences follow that ordinary EVM chains cannot offer:

- **One balance can back many contracts.** A wallet's funds are not locked inside any single token or merchant contract. The same balance can stand behind different contracts, so a customer does not pre-fund each merchant separately or shuffle value from one contract to another before spending it.
- **One balance can back several chains.** Merchant A can run its own private chain and merchant B another, each fully under its owner's control — and a customer's single wallet balance can pay on both.
- **Value moves between chains on the fly.** A balance can be shared with, or transferred between, independently owned chains, settled by the same quorum with the same zero-knowledge guarantees — no bridge, no wrapped token, no additional trust.

Chains become lightweight, per-business execution surfaces over one shared, private value layer: privacy per chain, without fragmenting anyone's money.

6.4 Verifying a payment on a chain you cannot browse

The private chain raises one more fair question. On a public chain, a payee verifies a payment by looking it up: confirm the transaction sits in a block, then wait for enough blocks to pile on top of it. On a private chain there are no blocks to browse — so how does anyone check that a payment really happened?

PoB replaces block-watching with something more direct. Every transaction receipt carries the hash of its underlying PoB transaction. Holding that receipt, a client can ask the validator quorum for the PoB transaction itself — which contains the zero-knowledge proof and the threshold-signed certificate — and verify both on its own. That is a *deterministic* proof of inclusion and validity: no block browsing, no confirmation counting, and it works identically whether the chain is public or private. (The receipt already carries the PoB transaction hash today; the self-serve client verifier that fetches and checks the quorum record is still under development.)

6.5 Privacy with accountability — and its honest limits

PoB is engineered for *business privacy with auditability*, which is different from — and for commerce, more useful than — absolute anonymity:

- Every payment leaves a threshold-signed certificate: a durable, independently verifiable receipt for disputes and audits.
- The Chain of Balance can be audited structurally — conservation of value holds over the entire graph, so an auditor can confirm the system as a whole has never inflated, without opening individual envelopes.
- In an operator-run deployment, the operator's own services (the wallet custodian) necessarily know what they need to serve their customers — like any financial institution. The privacy boundary is drawn against *the public and other participants*, not against the regulated operator.

What PoB privacy is not

PoB is not an anonymity coin. It does not try to hide participants from the operator, from lawful audit, or from the merchant a customer chooses to transact with. It hides the things businesses legitimately need hidden: balances, transaction amounts, counterparties, and activity patterns — from the network, from other participants, and from competitors.

7 Security

PoB's security case rests on two independent pillars, each of which was subjected to an adversarial, end-to-end analysis (assume-the-worst review of the full payment path). The design stance throughout: the fewer components you must trust, the fewer components can betray you.

7.1 Pillar one: value cannot be counterfeited

Every payment's zero-knowledge proof enforces conservation and range. The proof system is built on Bulletproofs++, a peer-reviewed construction, and the platform's custom aggregation layer is *machine-checked for equivalence* against the audited generic verifier — specifically closing the class of "under-constrained circuit" bugs that caused a well-known inflation incident in another privacy-focused chain. New money can enter the system only through an explicit, single-use *mint* attested by a custodian oracle (the cash-in path of Section 10); the attestation is bound to hardware-held signing keys, and every minted commitment can be used exactly once.

7.2 Pillar two: even our own engine cannot cheat

The most unusual — and we believe most important — property of the architecture is that **the execution engine is assumed malicious**. The validator quorum does not take the engine's word for anything:

- **Re-execution.** Each validator independently re-runs every EVM transaction against the state evidence submitted with it, and requires the recomputed results — state root, gas, receipts, and logs — to match exactly. An engine cannot lie about what a transaction did.
- **Certified state roots.** The one value a dishonest engine could still choose freely is the *starting state* it claims to execute against. PoB closes this by keeping a quorum-certified register of state roots: every transaction must chain to a state the quorum has already certified, all the way back to genesis. A fabricated "alternative history" fails this check immediately.
- **Non-combinable disagreement.** Validators sign the exact content of what they verified. Two conflicting versions of the same transaction can never both assemble a valid quorum certificate, because any two quorums overlap in honest validators who will sign only one.

7.3 The trust model, stated plainly

- **The validator quorum is the root of trust.** With n validators, safety holds as long as fewer than one third are compromised ($n = 3f+1$ tolerating f faults) — the classical Byzantine bound. Validators can be distributed across operators, partners, or jurisdictions to make correlated compromise hard.
- **The wallet service is a custodian.** Like a bank's core system, it is inside the trust perimeter and is protected accordingly: keys in cloud KMS/HSM, least-privilege capability tokens (amount-limited, time-limited, one-time) for every wallet operation, and no long-term secrets on client devices.
- **The chain engine is untrusted.** As above — it is checked, not believed. This is a deliberate inversion of the usual deployment, where the node software is the most trusted component.

Security posture in one sentence

Counterfeiting is prevented by mathematics (zero-knowledge conservation proofs), history is protected by distribution (a Byzantine quorum that re-executes and certifies everything), and custody is protected by hardware (KMS/HSM keys behind scoped, expiring capability tokens).

8 Programmable Payments: Hold, Settle, Release

Real commerce is rarely "pay X, done." Bookings change, meters run, deliveries are accepted or rejected. PoB builds the standard preauthorization-and-capture pattern directly into its token contracts as three primitives (Figure 4):

hold
: The customer places funds in escrow against a merchant — a reservation of value, made when the final amount is not yet known.

settle
: When the outcome is known, the merchant settles the actual amount. The difference automatically returns to the customer; fees and optional customer rewards are handled in the same atomic step.

release
: If the transaction is cancelled, the hold is released and funds return to the customer — by rule, not by dispute.

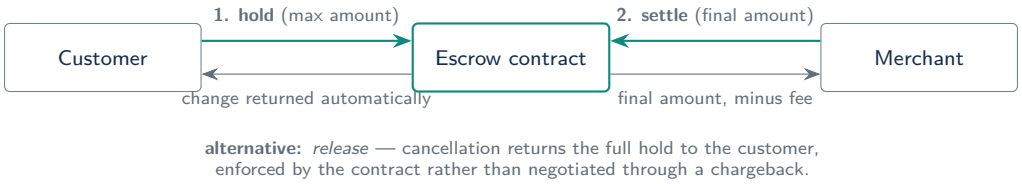


Figure 4: Escrow primitives map one-to-one onto preauthorization and capture — the flow behind fueling, parking, rentals, bookings, and metered services.

Because these rules execute inside contracts on the PoB chain, the "dispute process" for the everyday cases is simply the rule running as written: unknown final amounts stop being a chargeback generator and become a normal, cheap, private transaction shape. The same primitives are exposed uniformly across every surface — Solidity, the merchant REST API, and the agent tools of the next section.

9 Payments for AI Agents

9.1 Why agents break existing rails

Autonomous agents are becoming genuine economic actors: they search, compare, reserve, and increasingly need to *pay*. But an agent is not a cardholder, and it fails in different ways than a person does. An agent transacts fast, in volume, and unattended; it cannot "call the bank"; and the entity legally responsible for it — the user — needs guarantees that hold even if the agent reasons badly. That translates into five hard requirements for an agentic payment rail:

1. **Deterministic, low-latency settlement.** An agent mid-workflow needs a final answer in one interaction. Probabilistic finality forces the agent to poll, hedge, or guess — all failure modes at machine speed.
2. **Policy enforced by the rail, not by the model.** Spending limits must be guaranteed by infrastructure — cryptographic tokens and contracts — so that even a confused or manipulated agent *cannot* exceed them. Prompt-level guardrails are not a payments control.
3. **Escrow as the default shape.** Agents book, reserve, and commit to outcomes that resolve later. Hold-now / settle-when-known is the natural primitive, and PoB has it natively.
4. **Privacy for both sides.** High-frequency agent traffic on a transparent ledger is a real-time feed of a

business's operations and a user's life. PoB's sealed-envelope model keeps agent commerce confidential by default.

5. **No gas ceremony.** On a public chain, an agent that wants to pay must also hold the chain's gas token, estimate a fluctuating price, and someone must decide who funds it — the user, the agent's operator, or the merchant. That complication alone discourages autonomous payment. On PoB there is no gas at all (Section 5.3): costs are the merchant's per-contract fee configuration, and the agent never sees them.

9.2 How it works on PoB

The platform's agent surface is an **orchestrator** exposing business-level tools over MCP (the open Model Context Protocol used by ChatGPT, Claude, and other agent runtimes): *search offers, get a quote, create a reservation, extend it, cancel it, manage policy.* Agents speak commerce; payments happen behind the tools (Figure 5).

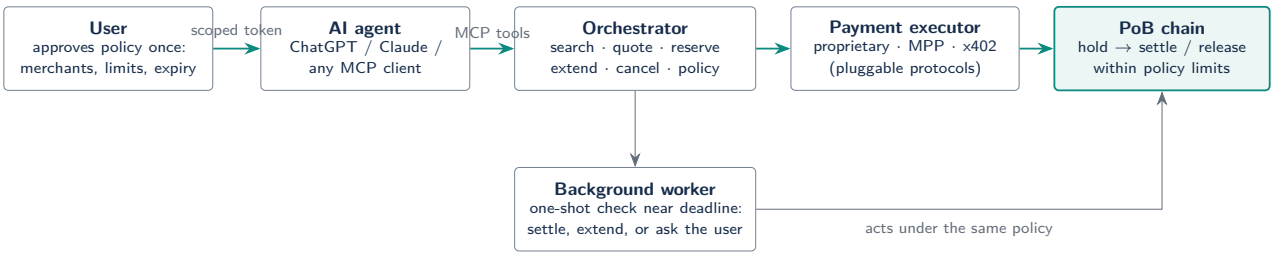


Figure 5: Agentic payment on PoB. The agent calls business tools; policy is enforced by scoped tokens and contracts underneath — not by the model's judgment.

Three design choices matter most:

- **User-approved policy, enforced below the agent.** The user authorizes an agent once, through a standard OAuth-style flow, and receives control in return: which merchants the agent may pay, the maximum it may hold, how long the authorization lives, and when the agent must come back and ask. The agent's credential is a scoped PoB wallet token — if the agent is tricked, the token still refuses.
- **Protocol-neutral payment boundary.** Machine-payment standards are young and plural. The orchestrator treats them as presentation formats: the same on-chain hold can be presented through a proprietary merchant webhook, through MPP (challenge–response machine payments), or through x402 (HTTP 402-based payment authentication). Merchants and agents keep their preferred protocol; settlement is PoB underneath, and new schemes can be added without touching the ledger.
- **Unattended follow-through.** Real bookings resolve later. A reservation can schedule a one-shot background check near its deadline that settles, extends, releases, or escalates to the user — under the same enforced policy as the original agent.

The fit, summarized

Deterministic settlement gives agents answers they can act on. Rail-enforced policy gives users guarantees that survive agent error. Native escrow matches how agents actually transact. Encrypted balances keep machine-speed commerce from becoming a public activity log. These are the properties PoB was built around — agentic payment is less a new feature than the payoff of the architecture.

10 Use Case: Semi-Closed-Loop Payment Systems

10.1 What a semi-closed loop is

A *semi-closed-loop* payment system is one where an operator issues stored value that customers can spend at a defined set of participating merchants — but not everywhere. Shopping-mall and multi-brand gift programs, fuel-station networks, transit and mobility platforms, campus and workplace payments, gaming and in-app economies, and marketplace credits are all semi-closed loops. It is one of the most common payment structures in the world, and one of the worst served: card rails take a percentage of every small transaction and offer no operator control, while fully in-house ledgers offer control with no independent guarantees. PoB is, we believe, the best-fit infrastructure for exactly this shape (Figure 6).

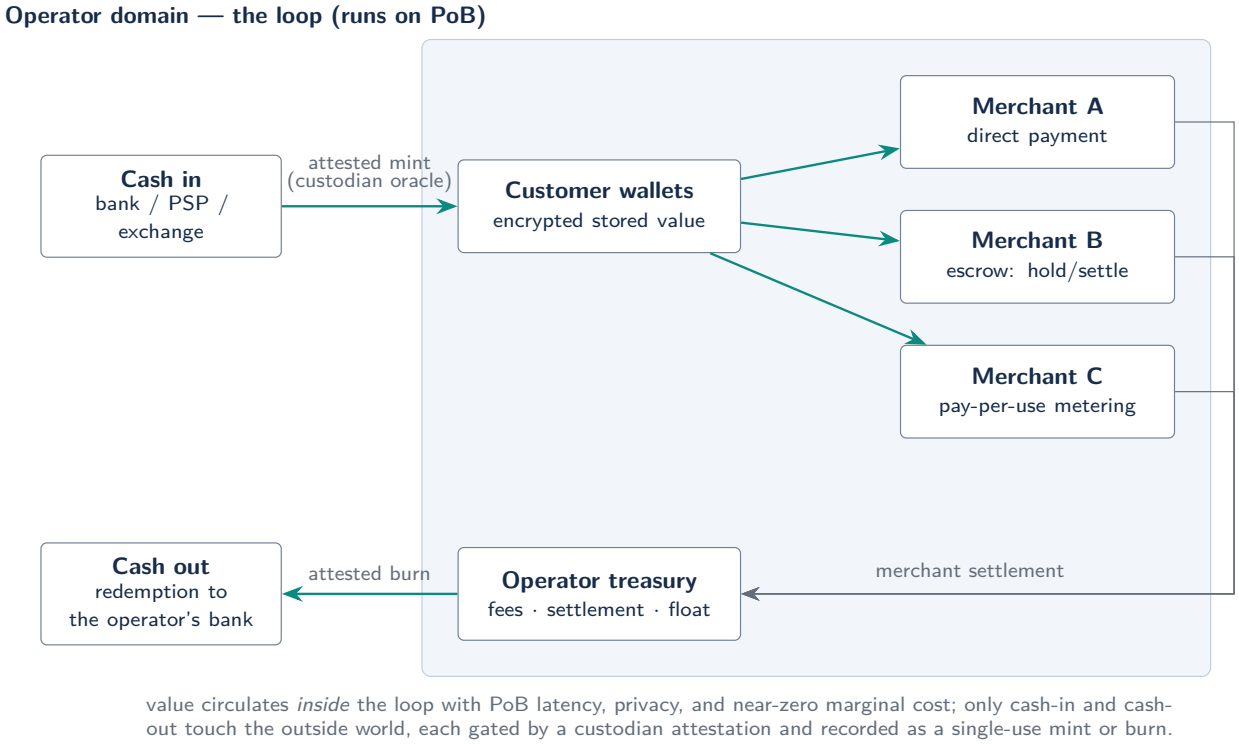


Figure 6: A semi-closed loop on PoB. The operator is the custodian; the validator quorum keeps the operator's ledger honest.

10.2 Why PoB fits this shape better than anything else

- **Operator custody is the native trust model.** PoB does not pretend to be a neutral global network; its wallet layer is explicitly custodial. The operator holds funds, defines the rules, and answers to its own regulator — while the Byzantine validator quorum (which can include independent members: partners, auditors, major merchants) guarantees that even the operator's own infrastructure cannot double-spend, inflate, or rewrite history. Trusted operation *with* independent verification is the combination neither card rails nor in-house ledgers offer.
- **Flexible, secure tokens.** Stored value is a contract, so the loop's rules are programmable: expiry, merchant scoping, per-purpose sub-balances, promotional value, delegated family or fleet spending under per-delegate limits — enforced by the same capability-token and contract machinery, not by after-the-fact monitoring. Every unit of value in the loop is backed one-for-one by an attested cash-in, and the conservation proof makes the float independently auditable at any moment — without exposing any individual balance.

- **Economics that survive small transactions.** No interchange percentage, no gas auction. A 40-cent locker rental or a per-minute meter tick costs what the infrastructure costs, which makes pay-per-use viable without prepaid-credit workarounds.
- **Privacy inside the loop.** Merchants in the same loop are often competitors. On PoB, participating merchants cannot see one another's volumes, prices, or customers: each participant reads only its own activity on the operator's private chain (Section 6.2), and nothing is broadcast to the public.
- **Loops that can span chains.** Each participating business can even run its own private chain. Because balances live in the shared PoB layer (Section 6.3), one customer balance can move among those chains on the fly — a "loop" need not be a single system, but can grow into a federation of independently owned private chains.
- **Escrow where the loop needs it.** The businesses that live in loops — fueling, mobility, rentals, bookings, metered services — are precisely the unknown-final-amount businesses of Section 8. Hold/settle/release is built in.
- **A ready path to agentic commerce.** A loop with an MCP orchestrator is immediately agent-accessible: a user's assistant can book the meeting room, top up the fuel card, or extend the rental — inside the same operator-defined, rail-enforced policy as every other participant.

11 What PoB Is — and Is Not

PoB is **payment infrastructure**, not a payment network. There is no universal acceptance mark, no network-wide rulebook, and no obligation to be open to everyone. An operator deploys the stack, acts as custodian, composes its own validator quorum, and defines its own loop — with cryptographic guarantees that its infrastructure follows its own rules.

	Card networks	**Public blockchains**	**PoB platform**
Finality	authorization instant; settlement and disputes take days	probabilistic, minutes to final	deterministic certificate, sub-second
Throughput	high, centrally operated	capped by global ordering	parallel; scales with quorum and hardware
Fees	percentage interchange	market-priced gas	gas-free; per-contract merchant fee and rewards
Privacy	network sees all data	public by default	encrypted balances on an access-controlled private chain
Programmability	fixed network rules	general but public	contracts + escrow + scoped capability tokens
Custody	prohibited for the network	user-managed keys	operator as custodian, quorum-verified
Agent readiness	card-sharing, no real delegation	uncertain finality; who pays gas?	gas-free; scoped tokens, policy, escrow, MCP surface

Table 1: Three rails, three sets of structural assumptions. PoB targets the cases where the first two structurally cannot go.

Equally important is what PoB does not claim. It is not an anonymity system aimed at evading oversight; the operator remains a regulated custodian with full ability to comply. It is not a global currency or a speculative asset; tokens in a loop represent attested, redeemable stored value. And it is not a finished universal network

— it is a platform an operator deploys for a defined economy, which is precisely why it can make guarantees universal networks cannot.

12 Conclusion

The blockchain industry has spent a decade trying to make global, transparent, probabilistically-final ledgers behave like payment systems. PoB starts from the opposite end: take the payment requirements as fixed — checkout latency, real finality, business privacy, programmable rules, sustainable economics — and keep only as much "blockchain" as those requirements justify. What survives is a specific and, we believe, correct combination:

- a write-once graph of encrypted balances instead of a public chain of blocks;
- zero-knowledge conservation proofs instead of published ledgers;
- a Byzantine validator quorum that certifies — and re-executes — everything, instead of trust in any single component, including our own;
- an EVM-compatible surface, a REST surface, and an MCP surface, so people, businesses, and AI agents each get the interface they already understand.

The near-term deployment we are building toward is the semi-closed loop — operator-issued stored value, circulating privately and cheaply among participating merchants, escrow-native, and open to policy-bound AI agents from day one. It is the setting where every property of the platform is load-bearing, and none is decorative.

Further reading

Formal foundations: *A Byzantine-Tolerant State Machine over a PoB Hypergraph* — the technical paper with the complete model and safety/liveness proofs. **Machine payments:** the PoB escrow drafts for MPP and x402 (payment-boundary protocol specifications). **Landscape:** an honest comparison with other payment and privacy chains — Tempo, Arc, Canton, Aztec, and others — is maintained on the zkLoop site, where it can stay current. **Contact:** info@zkloop.net.