Chain-of-Shards:
nested private chains with selective public finality

Abstract

Chain-of-Shards starts from a simple problem: public blockchains are immutable, but every transaction competes for blockspace. It moves that transaction volume onto nested private chains, where the application runs at very high throughput and pays mostly for compute. Each chain produces its proofs in its private domain, according to its deployment profile, then publishes to Ethereum whichever checkpoints need to become interoperable, auditable, or consumable by a contract. Gas is saved for the moments that call for the public layer: bridging, settlement, and audit.

1 Introduction

The blockchain trilemma is often presented as one of the main structural challenges facing blockchain networks: it pits security, scalability, and decentralization against one another. Bitcoin and Ethereum sit on the secure, decentralized side: anyone can verify them, but they remain slow and expensive. A second group favors speed and low cost (BNB Smart Chain and Base are telling examples). Their interoperability is often mistaken for decentralization, but the two are not the same thing: these chains are largely operated by a single company or a small set of validators. In practice, "fast and interoperable" today almost always means "fast and centralized."

The most valuable thing a public blockchain offers, immutable state, is also its most expensive resource. On Ethereum, every write enters the fee market, with its auctions, its congestion, and its system-wide costs. Chain-of-Shards reserves that immutability for the moments when it delivers value beyond the chain, and moves everyday execution onto fast, verifiable private chains.

This distinction runs through recent work on rollups, validiums, zero-knowledge bridges, and data availability [1–3]. It also sharpens a central point about interoperability: a fact proved off-chain means something different once a destination contract can read it on the public layer. In recent work, we formalized that boundary with the *Local*, *Proved*, and *Checkpointed* states [4].

Chain-of-Shards is a next-generation blockchain. A shard is a private chain in a nested family, attached to a common anchoring layer. Together, these chains form a polycentric fabric: each keeps its own deployment profile and its own publication cadence. A single anchoring layer can carry vast families of private chains, each identified by its own public seed. Each chain runs autonomously, pays mostly for its own compute (a low, predictable cost), then publishes to Ethereum only the state roots that need to become interoperable, auditable, or consumable by a contract.

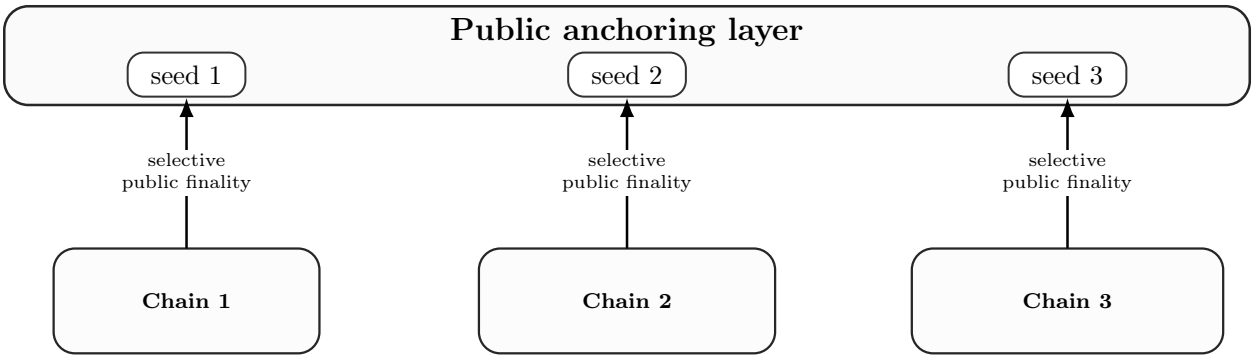

Figure 1: Each chain starts from a public seed, runs in its private domain, and publishes checkpoints to Ethereum when selected heights need to become public.

2 What defines a chain

A chain begins with a public seed. The seed is the starting point anyone can go back and read: it stores the chain's identity and lists its execution rules. It gives the chain a stable origin.

From that seed, the chain builds its history. At each height (each new block), it computes a commitment that summarizes its state: a balance, a log, or a state root.

The deployment profile describes how the chain actually operates. It answers four simple questions: who validates the chain, how its state can be recovered, who governs changes to it, and how block ordering is produced. This framing avoids reducing decentralization to a single label: two chains can anchor to the same public layer and behave very differently once in operation.

Each chain decides which heights need to become public. The rest can stay private (still useful, still proven) without ever being recorded on Ethereum. A chain can publish often, to serve bridges or fast settlement, or rarely, making public only the facts that matter most. All of these chains keep a verifiable origin and a clear proof trail.

3 *Zero-knowledge recursive state*

Speed comes from a simple separation: the private chain moves forward block after block, while zero-knowledge proofs provide a compact check on what was executed. As the chain advances, the validator (which in Chain-of-Shards also acts as the prover) generates a validity proof for each transition. Each proof condenses execution into a short, verifiable object tied to the seed.

Recursion gives each proof a longer reach. Because a proof commits to the proofs and commitments before it, the current height carries the chain's cryptographic continuity. An observer can verify the latest result at very little cost, with a clear link back to the history already proven.

This is the state the diagram calls the *zero-knowledge recursive state* (Figure 2). It sits after private execution and before publication on Ethereum. It prepares the move to the public layer without forcing it: the proof establishes validity; the checkpoint makes it public.

4 The three states

Our recent research on cross-chain interoperability formalized three states: *Local*, *Proved*, and *Checkpointed* [4]. These states form a cumulative progression: a commitment is local once execution sets it; it becomes proved when an off-chain proof attests to the state transition; it becomes checkpointed when a transaction records the proof and makes the commitment consumable on Ethereum.

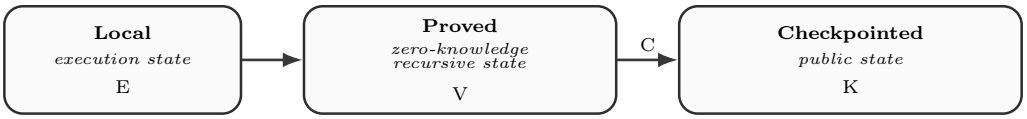

Figure 2: The three states and the operational sequence E → V → C → K (the four signals defined in Section 5).

In practice, a proven height comes with a compact cryptographic justification. An observer can verify it without replaying the whole execution. Once the height is checkpointed, the public checkpoint carries the chain's identity, the height, and the commitment that can be consumed on Ethereum. At that point, the commitment becomes a public fact attached to its chain.

This separation is crucial for contracts and bridges. An off-chain proof shows that a result is cryptographically valid inside the private domain; a public contract consumes that result the moment the corresponding checkpoint becomes readable on Ethereum. The boundary is sharp: the proof establishes validity; the checkpoint makes it public.

This architecture allows much finer control over costs. The chain produces its proofs in its private domain. Applications keep their internal cadence, then publish the heights that need public anchoring: a game publishes a tournament root, a financial system publishes a settlement, an enterprise chain publishes an audit proof, and a bridge publishes the state root needed for a withdrawal or a cross-domain transfer.

5 *Checkpointability*

Checkpointability answers a practical question: if the private height a participant is acting on were checkpointed right now, would Ethereum accept it? The answer has direct economic value. The private chain concentrates execution and proving in its own domain; Ethereum steps in when a height needs to become public. Checkpointability lets a participant know, before paying the gas to publish, that the height they are using has a valid path to the public layer.

Our recent research on *checkpointability* formalized this view with four signals: E for an eligible height, V for a verified proof, C for a public call that succeeds when simulated against the current state of Ethereum, and K for a confirmed checkpoint [5].

$$\mathbf{E} \longrightarrow \mathbf{V} \longrightarrow \mathbf{C} \longrightarrow \mathbf{K}$$

A user acting on a private chain does not need to publish that action right away. Verifying the proof is enough. If the proof is valid, they know the height can be publicly anchored at any time. They can keep operating in the private domain, defer paying gas, or let another participant publish the checkpoint later and make the state interoperable. Assets stay verifiable in the private domain, then become usable on Ethereum once the corresponding checkpoint is confirmed.

Checkpointability gives the private chain a permanent link to Ethereum. A proven height can stay private for as long as needed while remaining publishable on demand. That guarantee of "publishability" is what makes it possible to operate with confidence in the private domain.

6 Immutability

A Chain-of-Shards commitment follows a cycle that can be traced end to end. The application first produces a transition in its private domain. The chain computes a new commitment, records it in its local history, and ties the result to the seed. At this stage, the result is real for the chain's participants and fits into the chain's local cadence.

Next comes the proof. The validator generates a zero-knowledge proof and binds it to the right chain through the verification key. The chain now holds a proven result: the local state has a compact cryptographic

justification. In a recursive construction, the proof commits to the previous proofs and commitments, with no need to replay everything. This is the step that delivers local finality: validators and observers can verify the result at very little cost.

Publishing comes last. The chain selects a height, prepares the expected checkpoint call, and posts it to Ethereum. The checkpoint turns the proven commitment into a public fact. A bridge contract, a settlement system, or an auditor can then read the public checkpoint, attach it to the right chain, verify the proof, and consume the published result.

An application can stay fast in its private domain, keep its proofs available, then make public the result that carries external value. The seed is where coordination happens, the private chain is where the work gets done, and the proof connects the two. The result is an architecture where state travels as commitments, proofs, and checkpoints.

7 Deployment profiles

The deployment profile makes decentralization observable. It shows who validates the chain, how its state is recovered, who governs changes to it, and how its blocks are ordered.

Table 1: How to read the deployment profile.

Axis	**What it makes verifiable**
Validators	Who executes, proves, signs, and relays each step of the chain.
Data availability	Where the recoverable state, the reconstruction data, and the supporting proofs live.
Governance	Who authorizes upgrades, key rotations, parameter changes, and revisions to how the chain publishes to Ethereum.
Ordering	How transactions are ordered, how the block-producer role rotates, and how that ordering is attested and verified.

The deployment profile keeps data availability, governance, and ordering separate, chain by chain. That separation turns three risks (data withholding, governance capture, and censorship) into three distinct attack surfaces. It also distinguishes between two systems that look identical on paper but behave very differently once in operation.

Some heights can stay meaningful and locally proven without ever becoming public facts. Selective public finality becomes an explicit choice: the chain decides when its state should matter publicly and be usable across domains. Proofs and publication run on separate schedules, which gives applications far finer control than the usual all-or-nothing.

This framing makes these dimensions measurable: the Herfindahl-Hirschman index quantifies validator concentration, and the Nakamoto coefficient gauges resistance to censorship. Decentralization becomes objectively measurable.

8 Finality

Block production is sub-second, and finality strengthens over time. Once block n is produced, a zero-knowledge proof for it lands in a later block $n + x$.

Each round involves three validators (call them A, B, and C), one per step:

$$\text{step 1: } A, \qquad \text{step 2: } B, \qquad \text{step 3: } C.$$

Each validator produces a proof, signs it, and ties its step to the previous ones: B's step commits to A's result, and C's step commits to the combined result of A and B. The round forms a short sequence of explicit, verifiable agreements.

Every node runs the same checks from the seed: the expected validator, the round step, the transactions, and the proofs that go with them. A block that names an unexpected validator, commits to an invalid transaction, or breaks the expected order is rejected. Security comes from this consensus.

Finality becomes public when a checkpoint records the height on Ethereum. The height then becomes readable and usable by a contract, a bridge, a settlement system, or an auditor.

9 Conclusion

Chain-of-Shards starts from a simple economic fact: every transaction published on Ethereum passes through the fee market. That market provides strong security and verifiability, but it becomes expensive when it has to carry an application's entire volume. Chain-of-Shards moves that volume onto fast private chains, where the application pays mostly for compute and produces its proofs at its own pace.

The public layer then keeps its most valuable role. It steps in when a result must leave the private domain to serve a bridge, a settlement, an audit, an asset withdrawal, or a contract on Ethereum. The checkpoint is that crossing, and gas is paid exactly where it buys something: interoperability, and facts that matter beyond the chain that produced them.

This separation gives the model its strength: transactions stay fast in the private domain, proofs provide cryptographic verification of every transition, and publication makes selected heights interoperable. Chain-of-Shards can process a large volume of transactions in near real time while maintaining a high level of decentralization.

References

[1] Louis Tremblay Thibault, Tom Sarry, and Abdelhakim Senhaji Hafid. Blockchain scaling using rollups: A comprehensive survey. *IEEE Access*, 10:93039–93054, 2022.

[2] Stefanos Chaliasos, Denis Firsov, and Benjamin Livshits. Towards a formal foundation for blockchain ZK rollups. In *Proceedings of the 2025 ACM SIGSAC Conference on Computer and Communications Security*, pages 2714–2728, 2025.

[3] Muhammad Bin Saif, Sara Migliorini, and Fausto Spoto. A survey on data availability in layer 2 blockchain rollups: Open challenges and future improvements. *Future Internet*, 16(9):315, 2024.

[4] Jonathan Oleszkiewicz and Babu Pillai. Cross-chain interoperability: Local, proved, and checkpointed states, 2026. Manuscript.

[5] Jonathan Oleszkiewicz. Checkpointability: Separating proof availability from publication liveness, 2026. Manuscript.