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.

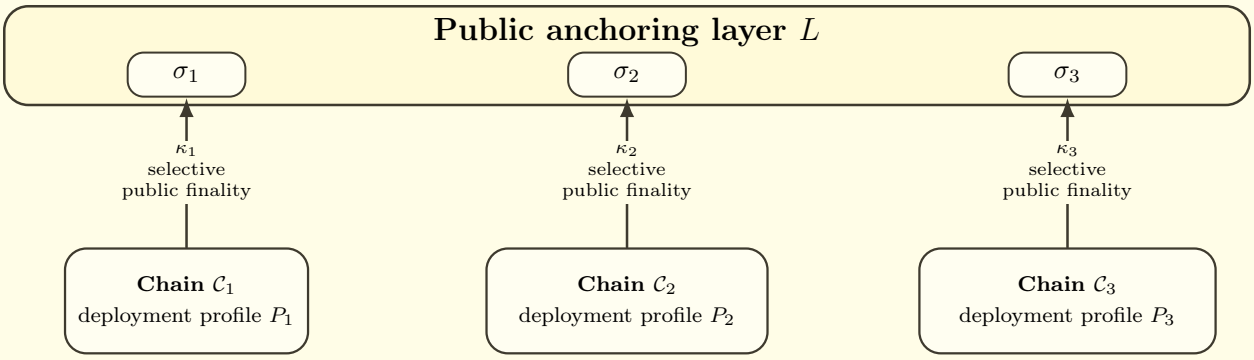

Figure 1: Each chain starts from a public seed, runs in its private domain, and publishes checkpoints to Ethereum when selected heights need to become public.

2 What defines a chain

Let L be a public anchoring layer. Every nested chain i starts from a public seed:

$$\sigma_i = (id_i, g_i, \gamma_i, vk_i), \tag{1}$$

where id_i is the chain's identifier, g_i its genesis commitment, γ_i the admissibility policy written into the seed, and vk_i the verification key for its checkpoint proofs. The policy γ_i commits to the expected profile, the validators, the rotation rule, the round timing, and the parameters that make local ordering verifiable.

The seed gives the chain its public origin. It fixes the parameters that let an observer, a validator, or a contract attach a proof to the right chain.

The resulting chain is:

$$\mathcal{C}_i = (\sigma_i, P_i, \rho_i^0, \rho_i^1, \ldots), \tag{2}$$

with the deployment profile as realized:

$$P_i = (V_i, A_i, G_i, O_i). \tag{3}$$

Here, V_i denotes the active validators; A_i describes the replication of the recoverable state or of the data needed to reconstruct it; G_i specifies the update authorities; O_i defines the local ordering rule.

For each chain, the seed anchors the deployment profile P_i in a common origin: identity, genesis commitment, admissibility policy, and the expected rules for rotation and verification. The profile P_i then specifies how that configuration operates in practice: who validates, who recovers, who governs, and who orders. Every node starts from the same seed to recompute the expected validator for each round and to check that every block follows the announced order.

This structure turns the family of chains into a space amenable to analysis. A single anchoring layer can then carry thousands of private chains. Each keeps its own publication policy. Some publish frequently, to serve bridges or fast settlement. Others accumulate private proofs and publish the heights that matter most, as their policy dictates. All of them keep a verifiable origin and an identifiable proof trail.

The commitment ρ_i^h summarizes the chain's state at height h. It can represent a state root, a log commitment, a computation output, or an application-level digest. What matters is the attachment: every commitment belongs to a named chain born from a public seed; it is verified under a known key and, where appropriate, published on L. That attachment replaces free-floating bridge metadata with full chain semantics.

3 *Zero-knowledge recursive state*

The proved stage can be read as a zero-knowledge recursive state. For a chain i and a height h, the commitment ρ_i^h describes the current state, while the proof π_i^h establishes the transition that leads to it. Recursion adds an explicit link to the history already proven: the current proof commits to the previous accumulator and produces a new one.

Write a_i^h for this recursive accumulator:

$$a_i^0 = \rho_i^0, \qquad a_i^h = \mathsf{Acc}(a_i^{h-1}, \rho_i^h, \pi_i^h).$$

Here, Acc is a cryptographic accumulator function: it combines the previous accumulator, the current commitment, and the current proof into one short value that represents the proven continuity up to height h. In practice, the proof at height h attests to the current transition and confirms that it extends the accumulator already accepted. An observer can therefore check the latest height and the chain's continuity without replaying the whole sequence.

The verification relation can then be read as a check of both height and continuity:

$$\mathsf{Verify}(vk_i, x_i^h, \pi_i^h) = 1.$$

It confirms two facts: the transition producing ρ_i^h is valid, and the accumulator a_i^h extends its predecessor a_i^{h-1} under the seed σ_i. (Here x_i^h is the public input for height h: it exposes the commitment and the accumulators that the proof links together.) This proof model makes plain what an observer gets: a compact proof of the current transition and a cryptographic link to the proofs before it. The zero-knowledge recursive state sets the stage for selective publication: as long as a height stays private, it remains verifiable in the private domain.

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.

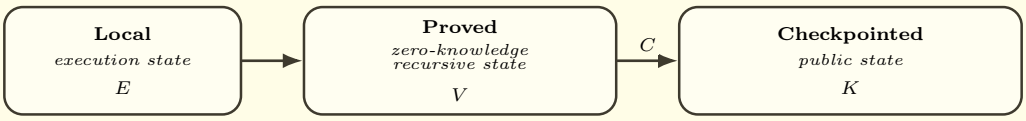

Figure 2: The three states and the operational sequence $E \to V \to C \to K$ (the four signals defined in Section 5).

For a height h, the chain pairs the public input x_i^h with a proof π_i^h.

The verification

$$\mathsf{Verify}(vk_i, x_i^h, \pi_i^h) = 1$$

moves the commitment to the proved stage: the local result now has a compact cryptographic justification.

The checkpoint κ_i^h steps in when this height must be consumed on L. The public checkpoint carries three elements: the chain's identity, the height, and the proven commitment. The proof establishes that the result is valid in the private domain; the checkpoint makes it usable in the public one.

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 ρ_i^h, records it in its local history, and ties the result to the seed σ_i. At this stage, the result is real for the chain's participants and fits into the local cadence defined by P_i.

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 id_i, verify the proof under vk_i, 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 $P_i = (V_i, A_i, G_i, O_i)$ makes decentralization observable.

Table 1: How to read the deployment profile.

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

The deployment profile P_i 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.

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

8 Validator failure

A private chain does not go dark when its validators stop cooperating. Chain-of-Shards handles this case by construction: the seed σ_i stays public, and with it the chain's identity, its genesis commitment, and the verification key vk_i. A third party can therefore attach an existing proof to the right chain and verify it without depending on the validators V_i.

The A_i axis of the deployment profile makes state recovery visible. A deployment where the reconstruction data stays under a single validator's control shows up as more concentrated; one where the data is replicated beyond any single operator's control shows up as more robust. State availability becomes a legible property of the profile rather than a matter of implicit trust in whoever happens to be validating.

Checkpointability then bounds the operational risk. Any height already proven can be published by any participant who can supply the proof, the commitment, and an admissible public call. Failure or censorship among the active validators can halt the production of new blocks; the proven history keeps its integrity, and heights whose proofs are in circulation keep their path to public anchoring. The risk boundary becomes explicit: what is proven and replicated survives its validators; what depends on a single custodian is visible in A_i before anyone relies on it.

The essential point is simple: active validators keep the chain moving forward, while heights already proven live on with their proofs. As soon as a participant can supply the proof, the commitment, and the admissible public call, they can get that height recognized on Ethereum. The seed identifies the right chain, and A_i spells out how the state can be recovered. A user can thus tell the difference between day-to-day operation and what is already verifiable and publishable.

9 Finality

Chain-of-Shards finality progresses in stages. The private chain produces sub-second blocks; their finality then strengthens over time, as validators prove each block's execution with zero-knowledge proofs. The seed

remains the common starting point: it carries the identity, the genesis commitment, and the expected profile, and lets every node recompute the expected validator and the round step, and identify the proof to verify.

Local finality. Block n is produced and becomes visible on the private chain. Nodes accept it when it follows the deterministic order announced by the seed: right round, right step, the expected validator, and a consistent local transition.

Proved finality. The zero-knowledge proof for block n lands in a later block $n + x$. The validator expected at that step produces the proof, signs it, and attaches it to the state in question. At this stage, the block is no longer merely visible: its state transition is verified by a compact proof, signed by its validator.

Deferred finality. A complete round gives the private chain its finality. For a round R whose deployment profile fixes a rotation A, then B, then C, up to N, the sequence reads:

$$
\begin{aligned}
A:&\ \text{proves and signs;}\\
B:&\ \text{verifies } A, \text{ then proves and signs a state that commits to } A;\\
C:&\ \text{verifies } A+B, \text{ then proves and signs a state that commits to } A+B;\\
&\qquad\qquad\vdots\\
N:&\ \text{verifies } A+B+\cdots+(N-1), \text{ then proves and signs a state that commits to } A+B+\cdots+N.
\end{aligned}
$$

Each validator checks that the steps before it followed the expected rotation and extended the commitments already produced. A block that names an unexpected validator, or commits to an unexpected state, is rejected by the other nodes. At the end of the round, the chain holds more than an isolated transition proof: it has reached deferred finality, built by accumulating recursive proofs and commitments.

Public finality. A height finalized in the private domain becomes consumable on Ethereum once a checkpoint is recorded and confirmed. It then becomes readable and usable by a contract, a bridge, a settlement system, or an auditor.

10 Where Chain-of-Shards sits

Table 2: The lines of work Chain-of-Shards builds on.

Line of work	**What it established**	**How it plays out in Chain-of-Shards**
Rollups / validiums [1, 2]	The separation of execution, proof, data availability, and public settlement.	A proven private chain whose selected heights go public through a chosen checkpoint.
Client-side validation [6]	Local verification of states or assets without blanket publication.	Proven private heights, verifiable in their own domain and usable through a checkpoint.
Anchored private sidechains [7, 8]	The value of a dedicated execution domain tied to a public root (seed).	An explicit progression from local to proved to checkpointed state.
Sequencers and ordering [9]	Why ordering, roles, and censorship resistance matter.	The profile P_i and progressive finality through zero-knowledge proofs, describing who validates, signs, relays, and verifies.

Table 2 reads as a map of what carries over. Chain-of-Shards builds on what rollups, client-side validation, anchored sidechains, and the work on sequencing have each established. Its contribution is to bring these dimensions together around one explicit unit: the nested private chain, whose heights stay private in the normal flow of operation and go public through a chosen checkpoint.

This composition yields three properties. First, it attaches every commitment to a seed. Second, it keeps cryptographic validity separate from public status. Third, it makes the deployment profile a visible, integral part of the chain's architecture. Selective public finality brings fine-grained control. An application can keep its execution private while the chain proves at its internal cadence, then publish on L the heights that matter for a bridge, external verification, an audit, or a settlement.

11 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.

[6] Maxim Orlovsky. RGB I.0: Scalable consensus for client-side validated smart contracts, 2025. IACR Cryptology ePrint Archive, Paper 2025/1400.

[7] Peter Robinson. Requirements for ethereum private sidechains, 2018. arXiv:1806.09834.

[8] Peter Robinson. The merits of using ethereum mainnet as a coordination blockchain for ethereum private sidechains. *The Knowledge Engineering Review*, 35:e30, 2020.

[9] Shashank Motepalli, Luciano Freitas, and Benjamin Livshits. SoK: Decentralized sequencers for rollups, 2023. arXiv:2310.03616.