Cross-Chain Interoperability: Local, Proved, and Checkpointed States

Jonathan Oleszkiewicz
BaaS.sh
France
jonathanolesz@gmail.com

Babu Pillai
Griffith University
Australia
pillai.babu@outlook.com

***Abstract*—Cross-chain applications need to know when a fact proven about one domain has become a contract-consumable fact on another. In selectively posted L1–L2 systems (validium-style), a height may already be locally stable and proved off-chain, yet no public checkpoint may exist on the anchoring layer L for a bridge or settlement contract to read. We study this minimal cross-domain setting and ask which strictly on-chain signals suffice for a contract to decide that a proved checkpoint claim is now consumable. We introduce a three-state model—*Local*, *Proved*, and *Checkpointed*—and show that checkpointed visibility cannot be determined from the proof-side view alone: in this model it additionally requires two on-chain observables, public eligibility and a proof-bound checkpoint record on the designated verifier/anchor path. We instantiate the model on a deterministic LocalV1 driver with real off-chain Groth16 proofs and an anvil-simulated anchoring layer. Across 10-, 50-, and 100-block prefixes, checkpoint materialization remains compact and nearly invariant at the contract boundary, consuming 198091–198169 gas while posting a fixed 168 B payload and 420 B of calldata. The resulting reproducible trace isolates a minimal interoperability primitive: the on-chain condition under which a proved cross-domain claim becomes consumable by a downstream contract.**

***Index Terms*—blockchain, cross-chain, interoperability, zero-knowledge proofs**

I. Introduction

Cross-chain and cross-domain interoperability hinges on when a statement proven about a source domain becomes a contract-consumable fact on a destination domain. Selective-posting Layer-2 nested chains create a visibility gap between off-chain proof completion and on-chain contract consumption on the anchoring layer L. For one nested chain instance, a height can become *local* when its prefix is stable under the chain's own rule, *proved* when a witness verifies against the instance's verifier relation, and only optionally *checkpointed* when the same claim is written back to L through the designated verifier/anchor path and becomes publicly readable there. This distinction matters for bridge, settlement, and cross-chain logic because proof completion alone does not tell a contract whether a canonical checkpoint fact exists on L.

While our prototype studies a single nested chain instance posting to a single anchoring layer, the question is a minimal interoperability primitive: any cross-domain consumer contract that conditions execution on a checkpointed commitment must decide, from on-chain signals alone, when a proved claim is publicly materialized and safe to consume.

We make three contributions. First, we define a minimal nested-chain object with a realized deployment descriptor, a public eligibility predicate, and a commitment history. Second, we identify the strictly on-chain observation boundary for checkpointing: proof-side information suffices for Proved, whereas $\mathrm{Checkpointed}$ additionally requires public eligibility and a proof-bound checkpoint record on L; proof-side non-determination under selective posting follows as a corollary. Third, we provide a purpose-built, bit-for-bit reproducible three-prefix trace on a pinned prototype stack that quantifies the contract-visible checkpoint footprint (payload bytes, calldata bytes, gas) for checkpoint materialization. Figure 1 summarizes the setting.

II. Cross-Chain Positioning and Related Work

We position our contribution as a minimal cross-domain (L1–L2) interoperability building block: the observation boundary a destination-chain contract faces when consuming checkpointed state commitments produced off-chain by a different domain.

Broad surveys review interoperability architectures, trust assumptions, and cross-chain mechanisms [1]. Recent CCW work studies interoperability patterns at the protocol/specification level [2]. Bridge-oriented work compares cross-rollup message bridges by gas, time complexity, and added trust assumptions, and also develops trustless cross-chain bridge constructions [3], [4]. At the Layer-2 systems level, surveys systematize rollups, channels, and related protocol families [5]. Recent formal analyses of rollups study how Layer-1 contracts enforce security and censorship-resistance mechanisms such as forced queues, upgrade controls, blacklist policies, and escape hatches [6], [7]. Those papers reason about protocol patterns, bridge behavior, or which mechanisms a rollup must expose to remain safe under a hostile or unavailable operator, but they do not isolate the narrower per-instance question studied here: when one proved height becomes a contract-consumable public checkpoint fact. We answer with a tightly scoped experiment: a reproducible measurement of the minimal public checkpoint footprint tied to one formally defined claim.

III. Minimal Model and Strictly On-Chain Observation Boundary

Running example. In a selectively posted validium-style deployment, a burn or lock at height h may already be proved

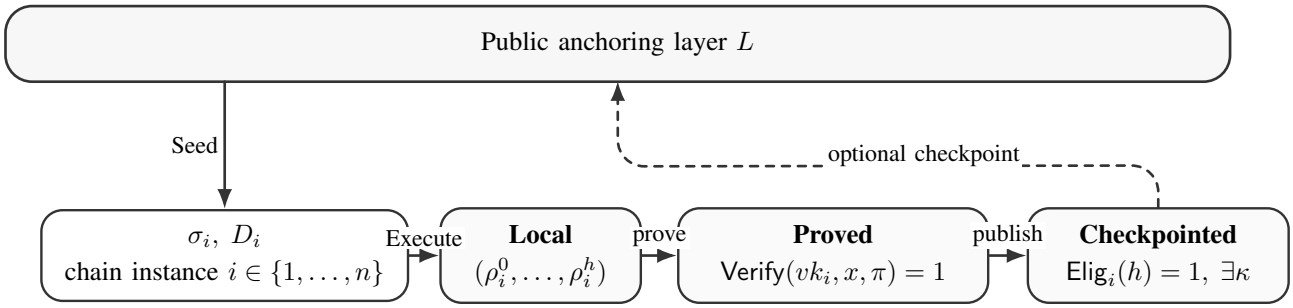

Fig. 1. One nested chain instance under a shared public layer L. The proof-side view fixes the instance description, stable history, stability witness, and proof witness, while checkpointing adds public eligibility and a contract-readable proof-bound record on L.

off-chain, yet a settlement contract on L should release only after the designated verifier/anchor path marks h eligible and stores a matching proof-bound checkpoint record. StarkEx-style validium exemplifies this split through DAC-backed off-chain data availability and availability-verifier checks, while public-DA rollups sit nearer the coupled end of the spectrum [8]–[10].

Let L be a public anchoring layer. In the model, L denotes the contract-visible anchoring layer; in our prototype instantiation, L is realized by an anvil-simulated EVM chain. We study one nested chain instance derived from a public seed on L: the seed fixes the genesis commitment and verifier relation, while later checkpoint publication is policy-governed rather than constitutive. Formally, a chain instance is

$$\mathcal{C}_i = (\sigma_i, D_i, \rho_i^0, \rho_i^1, \dots), \tag{1}$$

with public seed $\sigma_i = (cid_i, g_i, vk_i)$, where cid_i is the chain identifier echoed in the public checkpoint claim, and realized deployment descriptor

$$D_i = (S_i, \Theta_i^{\mathrm{pf}}, \Theta_i^L), \tag{2}$$

where S_i is the local stability rule, Θ_i^{pf} the remaining proof-side context, and Θ_i^L the designated verifier/anchor path on L together with the public eligibility interface it exposes. We assume the seed/history consistency invariant $\rho_i^0 = g_i$, and let ω denote whatever stability evidence S_i requires. Write

$$\mathsf{Elig}_i(h) := \mathsf{Adm}(\Theta_i^L, cid_i, h) \in \{0,1\} \tag{3}$$

for the public eligibility predicate returned by that designated path on L. For each height $h \geq 1$, define the chain-level claim

$$\mathrm{Claim}_i(h) := (cid_i, h, \rho_i^{h-1}, \rho_i^h). \tag{4}$$

This is intentionally a minimal checkpoint claim. The local state is determined by the stability rule:

$$\mathrm{Local}_i(h) \iff \exists\omega\ .\ S_i(h, \rho_i^0, \dots, \rho_i^h, \omega) = 1. \tag{5}$$

The intended cumulative reading is $\mathrm{Checkpointed}_i(h) \Rightarrow \mathrm{Proved}_i(h) \Rightarrow \mathrm{Local}_i(h)$. The first two states are defined directly, while the trust model below explains why the public checkpoint condition implies $\mathrm{Proved}_i(h)$:

$$\begin{aligned}\mathrm{Proved}_i(h) \iff\ & \mathrm{Local}_i(h) \wedge \exists x,\pi\ . \\ & \mathsf{Decode}(x) = \mathrm{Claim}_i(h) \\ & \wedge\ \mathsf{Verify}(vk_i, x, \pi) = 1,\end{aligned} \tag{6}$$

$$\begin{aligned}\mathrm{Checkpointed}_i(h) \iff\ & \mathsf{Elig}_i(h) = 1 \\ & \wedge\ \exists\kappa \in \mathcal{K}_i^L\ . \\ & \mathsf{Decode}(\mathsf{Payload}(\kappa)) = \mathrm{Claim}_i(h),\end{aligned} \tag{7}$$

Here $\mathcal{K}_i^L$ is the set of proof-bound checkpoint records materialized by the designated verifier/anchor contract for chain i on L and readable through its public interface. Stored records may carry auxiliary fields such as compact proof bytes or posting metadata; Decode projects from $\mathsf{Payload}(\kappa)$ to the claim-relevant tuple $\mathrm{Claim}_i(h)$. The two public observables are non-redundant: $\mathsf{Elig}_i(h)$ authorizes height h under the deployed policy, whereas κ shows that a matching public record was actually materialized on L.

Implementation assumption (proof-bound designated path). The designated verifier/anchor path on L is the unique source of records in $\mathcal{K}_i^L$. We assume those records are proof-bound: a record enters $\mathcal{K}_i^L$ only after the public path has recognized height h as local under S_i and bound the stored payload to the verifier relation fixed by the public seed, whether by direct on-chain verification or by an authenticated verifier-side hand-off. Under this assumption, (7) implies $\mathrm{Proved}_i(h)$.

Prototype mapping. In our prototype instantiation, the designated path is an Anchor contract that accepts `materialize` only with a valid EIP-712 attestation [11] from a designated verifier over a canonical payload. The stored payload binds $\mathrm{Claim}_i(h)$ and carries compact identifiers (e.g., vk hash and circuit id) and an artifact digest so that Decode projects back to the claim-relevant tuple.

For fixed (h, ω, x, π), define proof-side equivalence between chain instances by

$$\begin{aligned}\mathcal{C}_i \equiv_{\mathrm{pf}}^{h,\omega,x,\pi} \mathcal{C}_j \iff\ & \sigma_i = \sigma_j \\ & \wedge\ S_i = S_j \\ & \wedge\ \Theta_i^{\mathrm{pf}} = \Theta_j^{\mathrm{pf}} \\ & \wedge\ (\rho_i^0, \dots, \rho_i^h) = (\rho_j^0, \dots, \rho_j^h) \\ & \wedge\ S_i(h, \rho_i^0, \dots, \rho_i^h, \omega) = 1 \\ & \wedge\ \mathsf{Decode}(x) = \mathrm{Claim}_i(h) \\ & \wedge\ \mathrm{Claim}_i(h) = \mathrm{Claim}_j(h) \\ & \wedge\ \mathsf{Verify}(vk_i, x, \pi) = 1.\end{aligned} \tag{8}$$

Equivalently, the common proof-side view is

$$\pi_{\mathrm{proof}}(\mathcal{C}_i, h, \omega, x, \pi) := (\sigma_i, S_i, \Theta_i^{\mathrm{pf}}, \rho_i^0, \ldots, \rho_i^h, \omega, x, \pi)$$

which forgets the public checkpointing descriptor Θ_i^L and hence the strictly on-chain observables needed for checkpointing on L.

Strictly on-chain observation boundary. For fixed (h, κ), define the public checkpoint view

$$\pi_L(\mathcal{C}_i, h, \kappa) := (\mathsf{Elig}_i(h), \kappa).$$

Operationally, the public checkpoint tuple is recovered by decoding $\mathsf{Payload}(\kappa)$ on L. Across the class defined by (1)–(7), π_L is sufficient for a strictly on-chain consumer to decide whether $\mathrm{Checkpointed}_i(h)$ holds, and neither $\mathsf{Elig}_i(h)$ nor κ can be dropped in general. Omitting $\mathsf{Elig}_i(h)$ merges policy-authorized checkpoints with records that should not trigger settlement or bridge logic; omitting κ merges eligible heights that have not yet been materialized publicly with heights that have. Thus proof-side information determines *proved*, whereas checkpointed public visibility requires, in this minimal model, two on-chain observables: public eligibility and a proof-bound checkpoint record.

Corollary (proof-side non-determination under selective posting). If a deployment permits proved heights that are not routinely posted, there is no function F from proof-side views to $\{0, 1\}$ such that, for every chain instance $\mathcal{C}_i$ and every quadruple (h, ω, x, π) with $\mathrm{Proved}_i(h)$,

$$F\big(\pi_{\mathrm{proof}}(\mathcal{C}_i, h, \omega, x, \pi)\big) = 1 \iff \mathrm{Checkpointed}_i(h).$$

The corollary follows by constructing chain instances that agree on π_{proof} while differing in either $\mathsf{Elig}_i(h)$ or the presence of a matching stored record κ.

Coupled posting regime. If every proved height of interest is publicly eligible and yields a checkpoint record κ such that $\mathsf{Decode}(\mathsf{Payload}(\kappa)) = \mathrm{Claim}_i(h)$, then $\mathrm{Proved}_i(h) \Rightarrow \mathrm{Checkpointed}_i(h)$ on that regime. This marks the coupled design regime in which proving and checkpoint materialization collapse together at the heights selected by the public policy.

Table I places three adjacent families through this boundary: selective-posting validium, a coupled-end public-DA rollup, and a rollup with an escape hatch [6]–[8], [10].

TABLE I
ADJACENT PROOF-ENABLED FAMILIES THROUGH THE CHECKPOINTING BOUNDARY.

Family	Extra public observation beyond proof-side	Relation
StarkEx-style validium	public eligibility for h plus a checkpoint record materialized on L	generally $\mathrm{Proved} \not\Rightarrow \mathrm{Checkpointed}$
Public-DA rollup	routine public checkpoint record on L with eligibility fixed by policy	coincide at routinely posted heights
Rollup with escape hatch	normal public eligibility and record, or escape-hatch-triggered public record on L	separate normally; collapse when the public path is taken

IV. Prototype Instantiation and Measured Checkpoint Path

a) Experimental protocol.: We instantiate the model on a deterministic LocalV1 driver with a versioned workload commitment, off-chain Groth16 proofs [12] over a simplified segment transcript relation, and an anvil-simulated anchoring layer to isolate the strictly on-chain observation boundary. We run one campaign with $h \in \{10, 50, 100\}$ and exactly three runs per height. In each run, the driver replays one stable prefix, partitions it into m deterministic proof segments, generates real off-chain Groth16 proofs, locally verifies the full artifact, and publishes exactly one `materialize` transaction on the designated verifier/anchor path. Table II reports medians with $[\min, \max]$ over the three runs. We report the contract-visible checkpoint footprint as $B_{\mathrm{post}} := |\texttt{payload}|$, $B_{\mathrm{att}} := |\texttt{signature}|$, $B_{\mathrm{calldata}} := |\texttt{tx.input}|$ (selector+ABI), and C_{pub} as the receipt gas used by `materialize` (excluding deployment). Inclusion latency depends on deployment parameters, so gas and byte counts are the paper-facing public quantities for this pinned path; each run uses a fixed recorded `seedRun`.

b) Pinned environment and artifacts.: All runs use the same pinned prototype stack, recorded in every descriptor: Apple M2 Pro / arm64 / macOS host, driver version 0.1.0, arkworks-groth16 0.5.0, canonical BN254 proof encoding, and a fixed source snapshot. The anchoring contracts are compiled with Solidity 0.8.27 with optimizer enabled (200 runs) for EVM Cancun. Each run emits a canonical artifact bundle (descriptor, manifest, payload, and proofs) from which the reported metrics are recomputable. To facilitate reproducibility, the implementation code, raw run traces, regeneration scripts, and experimental artifacts will be released in a public repository upon acceptance.

c) What we treat as results.: Each row captures one complete collaboration cycle on one implemented stack: replay LocalV1 exactly, generate and verify real off-chain Groth16 proofs, bind the resulting artifact to a fixed payload and EIP-712 attestation [11], and then materialize one compact checkpoint record for downstream contract consumption. We report two classes of data with different interpretive weight. First, the public checkpoint footprint (B_{post}, B_{calldata}, C_{pub}) is the primary paper-facing result, because it is directly tied to the contract-visible observation boundary. Here B_{art} denotes the total serialized artifact size (`workload.json`, `descriptor.json`, `manifest.json`, `vk.bin`, and proof blobs), while the proof blobs alone occupy $B_{\mathrm{art}}^{\mathrm{proofs}} = 128m$ B on this stack. Second, off-chain proving times and artifact sizes are reported as stack-specific context with full environment metadata. LocalV1 replay time is recorded for completeness, but we do not treat it as a result because the driver is intentionally synthetic and optimized for deterministic replay. We also deploy a minimal consumer contract that decides *Checkpointed* from eligibility and record only, and validate its state transition around each `materialize` call.

TABLE II
PAPER-FACING PUBLIC FOOTPRINT AND STACK-SPECIFIC CONTEXT OVER THREE RUNS ON THE IMPLEMENTED PROTOTYPE STACK. THE LOCAL COLUMN RECORDS THE DETERMINISTIC REPLAY REGIME ONLY.

h	**Local**	**Proved**	**Checkpointed**
10	deterministic	m=4	B_{post}=168 B
	LocalV1	τ_{prove}=3.45 s	B_{att}=65 B (fixed)
	replay	[3.45, 5.06]	B_{calldata}=420 B (fixed)
	no timing	B_{art}=3.7 KB	C_{pub}=198.1k gas
	claim	$\tau_{\mathrm{ver}}^{\mathrm{off}}$=0.62 [0.61, 0.94] s	[198.091, 198.091]k
50	deterministic	m=20	B_{post}=168 B
	LocalV1	τ_{prove}=18.52 s	B_{att}=65 B (fixed)
	replay	[16.52, 19.90]	B_{calldata}=420 B (fixed)
	no timing	B_{art}=11.0 KB	C_{pub}=198.1k gas
	claim	$\tau_{\mathrm{ver}}^{\mathrm{off}}$=3.05 [3.03, 3.08] s	[198.116, 198.140]k
100	deterministic	m=40	B_{post}=168 B
	LocalV1	τ_{prove}=38.95 s	B_{att}=65 B (fixed)
	replay	[35.72, 42.95]	B_{calldata}=420 B (fixed)
	no timing	B_{art}=20.1 KB	C_{pub}=198.1k gas
	claim	$\tau_{\mathrm{ver}}^{\mathrm{off}}$=6.24 [6.21, 6.77] s	[198.145, 198.169]k

Table II should be read as a marginal progression $\mathrm{Local} \rightarrow \mathrm{Proved} \rightarrow \mathrm{Checkpointed}$ on one implemented stack; m denotes the number of deterministic proof segments. The Local column records the replay regime only. The Proved column reports contextual off-chain costs on this stack, while the Checkpointed column reports the compact public footprint that a downstream contract would observe. The chosen heights induce $m \in \{4, 20, 40\}$, i.e., a $10\times$ increase in certified prefix length and proof count; across these repeated runs, the public checkpoint footprint remains compact and nearly invariant while proof work stays off-chain and scales with segmentation, which is exactly the separation claimed by the model. Operationally, a downstream bridge or settlement contract would read the checkpoint record only after it appears on L. *Proved* alone is not enough for such a contract, because the proof-side artifact may exist without any canonical public fact at height h. In this sense, *Checkpointed* is the first state at which the claim becomes contract-consumable on the anchoring layer. Posting the full artifact bundle (B_{art}) would expose about $22\times$–$123\times$ more public bytes than the fixed checkpoint payload ($B_{\mathrm{post}} = 168$ B) across these measured prefixes (Table II).

d) Interpretation.: The trace is collected on one pinned host with three runs per height and one fixed proof backend, providing a controlled baseline instantiation of the model. Across these prefixes, the contract-visible checkpoint record remains compact and nearly invariant while proof generation and artifact production remain off-chain and scale with the chosen segmentation.

V. CONCLUSION

This paper isolates a gap that matters for selective-posting L2s and validium-style systems: local stability, proof validity, and contract-readable checkpointing on L need not coincide. In this minimal model, closing the gap from *Proved* to *Checkpointed* requires two public observables: eligibility and a proof-bound checkpoint record. Our prototype implementation realizes this minimal public observation boundary on a deterministic LocalV1 driver with real off-chain Groth16 proofs and an anvil-simulated anchoring layer. The main experimental takeaway is the contract-visible checkpoint footprint: the record remains compact and nearly constant across the measured prefixes (168 B payload, 420 B calldata, 198091–198169 gas), while proof generation and artifact production remain off-chain and stack-specific. We do not claim L1 latency or production-host realism; the claim is that the model isolates exactly what a downstream contract must see to treat a checkpoint as a public fact on L, and that this boundary is experimentally reproducible on the implemented stack.

REFERENCES

[1] K. Ren, N.-M. Ho, D. Loghin, T.-T. Nguyen, B. C. Ooi, Q.-T. Ta, and F. Zhu, "Interoperability in blockchain: A survey," *IEEE Transactions on Knowledge and Data Engineering*, vol. 35, no. 12, pp. 12 750–12 769, 2023.

[2] G. Llambias, L. Gonzalez, and R. Ruggia, "An approach to formalise blockchain interoperability patterns," in *2025 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, 2025, pp. 1–9.

[3] D. L. Fekete and A. Kiss, "Trust-minimized optimistic cross-rollup arbitrary message bridge," *Journal of Network and Computer Applications*, vol. 221, p. 103771, 2024.

[4] T. Xie, J. Zhang, Z. Cheng, F. Zhang, Y. Zhang, Y. Jia, D. Boneh, and D. Song, "zkbridge: Trustless cross-chain bridges made practical," in *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, 2022, pp. 3003–3017.

[5] A. Gangwal, H. R. Gangavalli, and A. Thirupathi, "A survey of layer-two blockchain protocols," *Journal of Network and Computer Applications*, vol. 209, p. 103539, 2023.

[6] S. Chaliasos, D. Firsov, and B. Livshits, "Towards a formal foundation for blockchain ZK rollups," in *Proceedings of the 2025 ACM SIGSAC Conference on Computer and Communications Security*, 2025, pp. 2714–2728.

[7] F. G. Figueira, M. Derka, C. L. Chiu, and J. Gorzny, "A practical rollup escape hatch design," in *2025 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, 2025, pp. 1–5.

[8] StarkWare Industries, "Data availability: StarkEx documentation," accessed 2026-03-31. [Online]. Available: https://docs.starkware.co/starkex/con_data_availability.html

[9] ——, "Contract management: StarkEx documentation," accessed 2026-03-31. [Online]. Available: https://docs.starkware.co/starkex/perpetual/shared/contract-management.html

[10] V. Buterin, D. Feist, D. Loerakker, G. Kadianakis, M. Garnett, M. Taiwo, and A. Dietrichs, "EIP-4844: Shard blob transactions," Ethereum Improvement Proposal, 2022, created 2022-02-25. [Online]. Available: https://eips.ethereum.org/EIPS/eip-4844

[11] R. Bloemen, L. Logvinov, and J. Evans, "EIP-712: Typed structured data hashing and signing," Ethereum Improvement Proposal, 2017, created 2017-09-12. [Online]. Available: https://eips.ethereum.org/EIPS/eip-712

[12] J. Groth, "On the size of pairing-based non-interactive arguments," in *Advances in Cryptology – EUROCRYPT 2016, Part II*, ser. Lecture Notes in Computer Science, vol. 9666, 2016, pp. 305–326.