By Jacob McDorman, Co-Founder & CTO at Saga, reviewed by Mustafa Al-Bassam, CEO and co-Founder of Celestia Labs

This exploration has been inspired by Decentralized Rollups Sequencing as a Service via Interchain Security (Celestia Forum).

TL;DR

- Saga provides a platform to organize validators into sequencers and punish misbehavior via interchain security
- Validators become sequencer via an on rollup scheduler on a weighted per-epoch basis
- Two mechanisms detect service reliability and fraud faults, respectively: a challenge to process a set of transactions and fraud proof generation
- Celestia provides the underlying DA that secures fraud proofs and reliability challenges

ARCHITECTURE

Saga is a Cosmos interchain security and validator services designed to address analogous obstacles – bespoke token issuance and validator solicitation prerequisites. The Saga service model is simple: provisioners of blockchains pay validators to operate blockchains, à la a traditional service model. Saga however is more generally capable; it can deploy any technology as a service that implements (1) the consumer side of the interchain security protocol and (2) that has a tenable operating model. Ostensibly Rollmint could meet these two requirements, therefore form a composition with Saga that would constitute the aforementioned sequencer as a service.

Interchain Security

An implementation of consumer-side interchain security must primarily: receive validator set changes produced by an interchain security provider and emit slashing intents to that corresponding provider. For the sequencer service, Rollmint is our consumer of interchain security. Within a rollup design context: Rollmint receives validator sets to determine the set of sequencers and emit slashing intents to punish misbehaving sequencers per our operating model.

[

951×124 5.17 KB

](https://forum.celestia.org/uploads/default/original/1X/03213e279f22304990851c7b6e61956c9af6db1f.png)

Operating Model

In this model, the set of validators inherited from Saga via interchain security are responsible for operating the sequencer service. They are assigned per rollup sequencer duty by a scheduler proportional to their known stake weights. This scheduler maps a validator set and scheduling epoch* tuple to a single (assignee) validator. Furthermore, because the scheduler mapping is known by each validator operator and is implemented on the rollup chain, it represents provable service terms for the sequencer as a service.

The sequencer service must ensure transactions are always processed. Both downtime and transaction censoring violate this expectation. Validators with sequencer duty must remain available and must not censor. Prospective validators can be challenged** to process a set of transactions committed to Celestia to determine either misbehavior; bearing in mind, the duration of the challenge should distinguish a busy validator or a congested Celestia from a misbehaving validator. Any faulted validator is replaced by the succeeding epoch's assigned validator to maintain availability.

The above reliability fault should be visible on the rollup chain. The fault can be partly reasoned from the inequality of the schedule-anticipated block signatures and actual block signatures; this inequality however can produce false-faults in the event that a validator performs sequencer duty early, though this false-fault occurrence can be averted if the Rollmint generational block construction ignores candidate parent blocks that are proposed out of turn.

The sequencer service must prevent fraud. The model subsumed here is the optimistic fraud proof model – fraud proofs are generated by observing parties** and broadcasted via Celestia. Furthermore, fraud proofs should be incorporated into an honest sibling block of the affected rollup to make the fault observable on the rollup chain, or alternatively, fraud proofs can be relayed directly to the Saga interchain security client.

The preceding designs form the basis for a system capable of punishing unreliable validators. The transformation on the rollup chain of misbehavior to a slashing invocation via interchain security seems feasible.

Footnote * We have not discussed the scheduler epoch. The choice of epoch reference source may be controversial; ideally the choice should be internally measurable on the rollup chain and tamperproof to ensure correct availability fault reasoning. One potential source may be: multiples of aggregated transaction count or even aggregated gas consumption. This choice seems both internally measurable and tamperproof, but no rigorous work has been applied to such a claim.

Footnote ** We have not discussed the incentives for third-party actors who challenge availability or fraud. Presumably these actors are rewarded a portion of the validator-slashed capital. The implementation of this behavior would need to alter

the interchain security interface and Cosmos slashing logic, though these alterations seem reasonable.

Fraud Provable IBC

Our interchain security functionality calls for an optimistic rollup IBC client. This client would synchronize block headers per usual but hold a block delivery until the fraud challenge window has expired uncontested, as well as insure that the correct validator-sequencer signed the block to handle unwanted orphans.

Further nuance is warranted. We will hold for later discourse.

SUMMARY

We have devised a sequencer as a service deployable on Saga. Our approach depicts a consumer-side interchain security method with a tenable operating model compatible with Saga, consequently facilitating a service that provides a simple means to launch Rollmint + Celestia rollups.

These designs are a work in process and need further exploration – holes found and depths plumbed.