native rollups - Forschungsingenieurtagung L2day

12th of June 2025 - Berlin

Workshop host: Justin Drake

—

Justin introduces a mental how to think about for different layers of a chain:

- settlement: L1 vs L2
- data: rollups (eg. validium, optimium)
- sequencing (eg. centralised, based)
- execution (eg. native)

An extreme version of a rollup would be based and native → ultrasound rollup

Native rollups tackle different types of problems

First order problems:
- execution bugs (in the zkEVM, or opEVM)
    - reason for having security councils
- EVM maintenance
    - burden for governance (is an attack vector itself)

Aspirational goal of native rollups: bulletproof security → no bugs, no security council, no governance attack

Second order problems:
- (lower the) barrier to entry (eg. an Ethereum „shard" in 100 lines of solidity → no need for complicated fraud proofs, circuits, councils)
- (lack of) synchronous composability
    - synergies with delayed execution
    - synergies with based sequencing

The proposal: EXECUTE (a new precompile)

- takes inputs: *pre_state_root*, *post_state_root*, *trace* (including TXs, contexts, state access witnesses which can be optionally SNARKified), *gas_used*
- outputs *true/false*

precompile checks that:

- the stateless execution of TXs starting from *pre_state_root*, ends at *post_state_root*

- provides a form of EVM introspection

Note: witness size is substantial, but in case of optimistic rollups only needs to be called in a fraud proof challenge

The proposal contains another precompile: DERIVE

- takes inputs: TXs data (eg. pointers to blobs)
- outputs: TXs that feed into the EXECUTE precompile

What it does: starts with *txs_data*, processes it - meaning: unpacking, concatenating, decompressing, serializing, …

returns: TXs for consumption by EXECUTE → is easier with „lean data“ and bloblessness vs inflexible blobs

What are ways of enforcement?

Enforcement is subjective

- i.e. non-enshrined and offchain
- long-term maybe enshrined → allowing for native validiums

Enforcement has two main flavours:

- pure re-execution
- pure SNARKs

alternatives: multiproving (TEEs, n-of-k committees, anything which has real-time proving)

In the case of **re-execution**:

- no use of blobs
- limited by small gas limit (eg. 10m)
  which is sufficient for optimistic rollups and optimiums, because EXECUTE and DERIVE are only called for disputes

- minimal overhead for validators
  no state or bandwidth impact, no state growths, no state accesses
  execution is parallelisable across cores

In the case of **SNARK-based enforcement**

- with a snarkified L1 we get enforcement „for free“
- allows for potentially unlimited gas limit

Conclusio - definition of a native rollup

→A rollup that uses EXECUTE to process user transactions.

mental model: a native rollup is a programmable shard
- allowing for customisations: eg. wrt sequencing, bridging, governance, gas token, system transactions

Challenges

- native rollups are not universal, require EVM equivalence (excluding eg. SVM, CarioVM, …); though a game-changer could be exposing RISC-V on the L1
- no state diffing possible → requires enshrined zkEVM

One common criticism of native rollups: today's rollup can deviate from EVM and support r1 curve → response: L1 needs to adapt (see EIP-7212)