1k(x)

Interoperability between L2s: Latest developments, Framework and Challenges

Marshall Vyletel and Wei Dai
1kx

We have a definition issue...

shumo | UPA powered 🪐 🧬 🚀
@shumochu

It takes me a while to align semantics of the technical terms with protocol researchers, here is my proposal:
- **Asynchronous Composability**: There are some kinds of composability between rollups. You can argue that we already have that using bridges today.
- **Atomic Inclusion**: A set of transactions on di… can be guaranteed to be settled in a given peri…
- **Synchronous (atomic) composabi…**
can call a smart contract in Rollup B…

vitalik.eth
@VitalikButerin

Synchronous atomic composability is very overrated imo. Like, think about what are some specific cross-L2 things *you* are already doing or envision yourself doing that could be more seamless. For me, the top two are:

1. I have coins on Optimism, I want to pay Bob, but Bob is only on Arbitrum.
2. I have coins on Taiko, I want to use a dapp on Polygon, so I need to send-to-self to Polygon in order to use that dapp.

These are not fancy nerd problems that can be fixed by solving synchrony. These are UX problems that can be fixed with:

avaunt.xrd
@a_vaunt

I would swap Atomic Inclusion and Atomic Composability personally.

IMO, Atomic composability is the combination of Atomic Execution; guaranteeing tx's from different rollups will 'execute' together and Atomic commitment (inclusion); ensuring all state changes will happen or not.

12:40 PM · Jun 19, 2024 · **480** Views

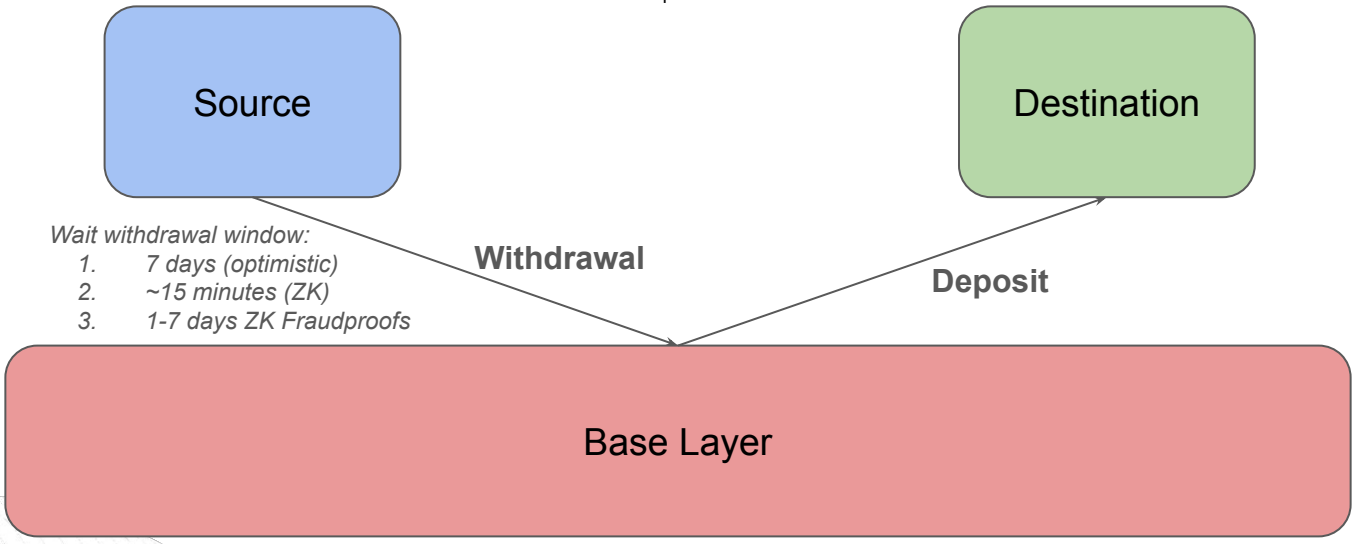
INTRODUCING...

The 6-Level Framework

1. L1 Async
2. Atomic Inclusion
3. Shared Settlement
4. Atomic Execution
5. *Block-Level* Composability
6. *Tx-Level* Composability

1. L1 ASYNC - I

Definition: Manual withdrawal from the source rollup, waiting the withdrawal window, and manually depositing into the destination rollup



1. L1 ASYNC - II

Advantages/Capabilities:

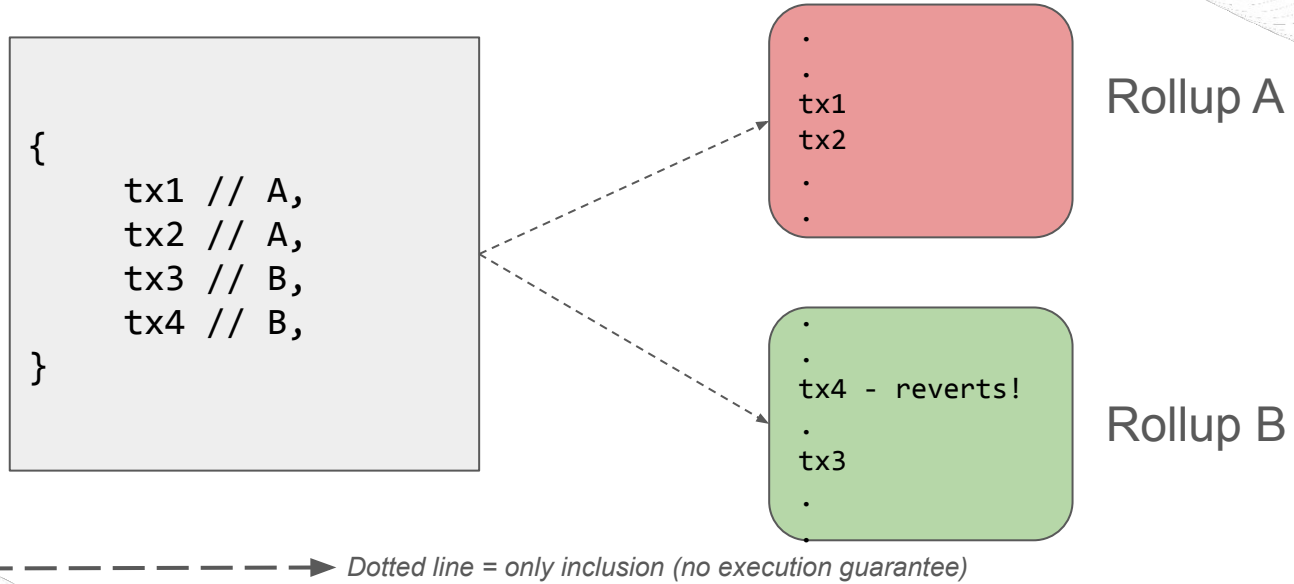
1. Trustless
2. No wrapped assets

Necessary Infrastructure:

3. None

2. ATOMIC INCLUSION - I

Definition: A bundle of transactions will either all be *included* in the next block, or none will be

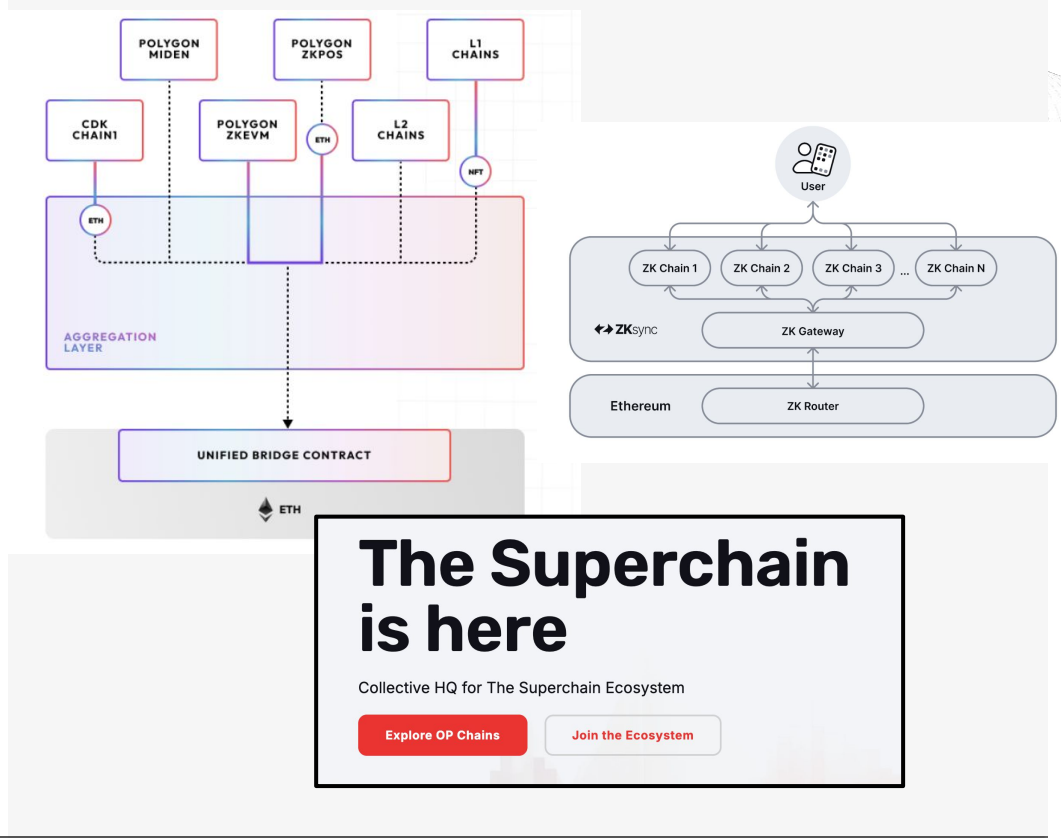


2. ATOMIC INCLUSION - II

1. **Advantages/Capabilities:**
    a. None - Arbitrageurs could use this, but riskier
    b. Essentially equivalent to L1 Asynchronous
2. **Necessary Infrastructure:**
    a. “Lazy” shared sequencer
    b. *Technically* possible manually if rollups are not at max throughput

3. SHARED SETTLEMENT - I

Definition: Multiple rollups connecting to the base layer via a shared bridge contract and proof aggregation layer

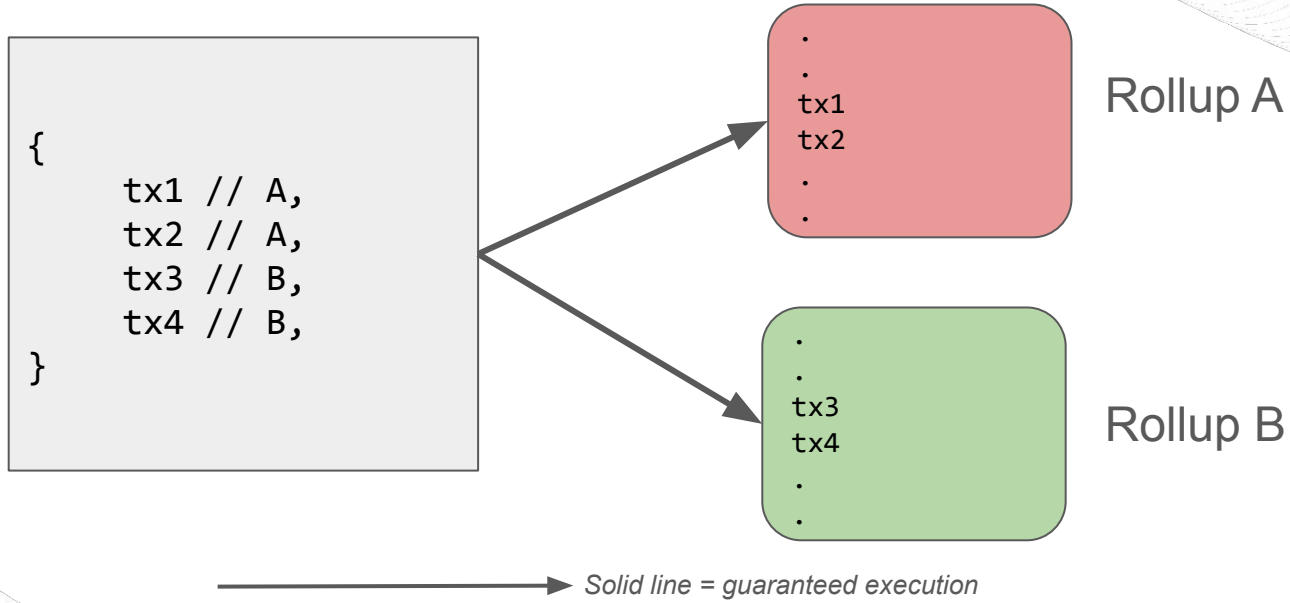


3. SHARED SETTLEMENT - II

1. **Advantages/Capabilities:**
    a. **No wrapping assets** - tokens can be native to all connected rollups
    b. In-protocol shared messaging standards
2. **Required Infrastructure:**
    a. Shared bridge contract
    b. Relays* - (an entity to facilitate message passing)
        i. Not necessary if async - rollups can read x-rollup messages from shared bridge contract
    c. Proof Aggregation

4. ATOMIC EXECUTION - I

Definition: all transactions within a cross-rollup bundle will either all execute successfully, or none will

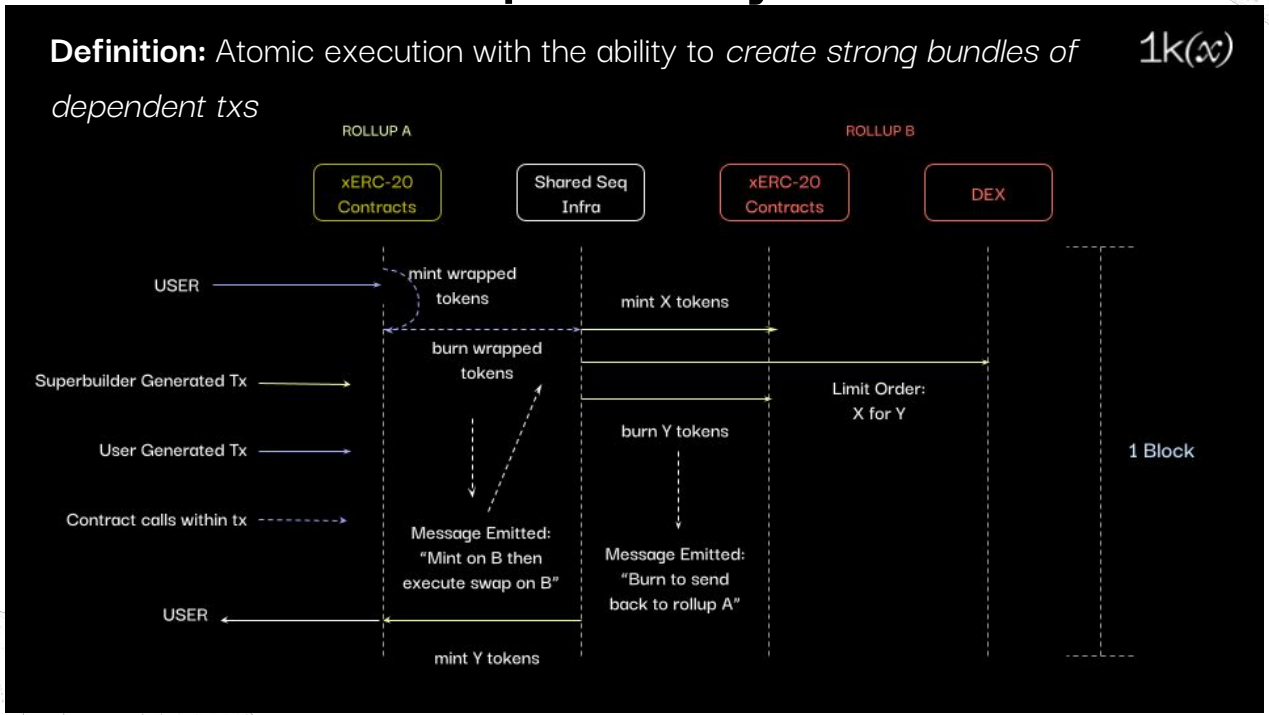


4. ATOMIC EXECUTION - II

1. **Advantages/Capabilities:**
    a. Cross-rollup arbitrage opportunities
2. **Required Infrastructure:**
    a. Shared Sequencer
    b. Superbuilder*
        i. Not *technically* required (full node sequencers)
    c. Soft-finality layer*
        i. Could be achieved with decentralized shared sequencer network

5. Block-Level Composability - I

Definition: Atomic execution with the ability to *create strong bundles of dependent txs*

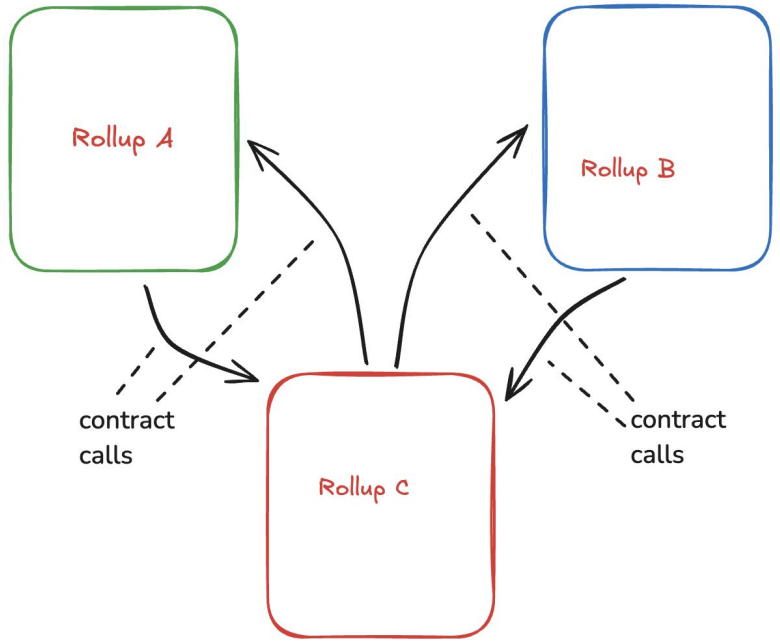


5. Block-Level Composability - II

1. **Advantages/Capabilities:**
    a. Creating bundles of dependent cross-rollup transactions
    b. Single block cross-rollup swaps / limit orders
2. **Required Infrastructure:**
    a. Shared Sequencer
        i. Or superbuilder
    b. Soft-confirmation Layer
        i. Ensures validity of bundles

6. TX-Level Composability - I

Definition: The ability to compose between two or more rollups within a single transaction, mirroring smart-contract level composability

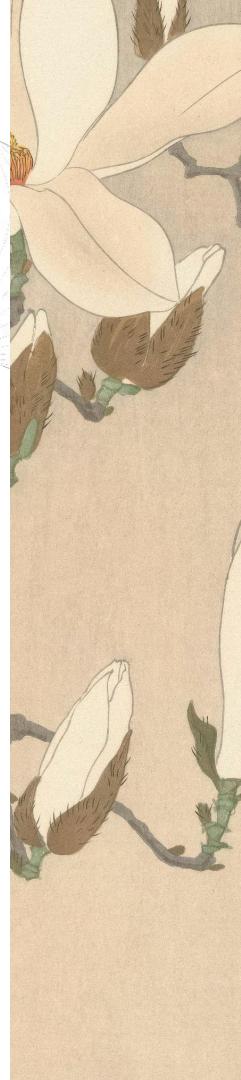


6. TX-Level Composability - I

1. **Advantages/Capabilities:**
    a. Smart-contract-like experience across all rollups
    b. Guarantee that state changes prior to any call can be reverted when it returns
    c. Cross-rollup flashloans
2. **Required Infrastructure:**
    a. VM-Level changes* - have to build a dapp differently
    b. Shared Sequencer
    c. Superbuilder

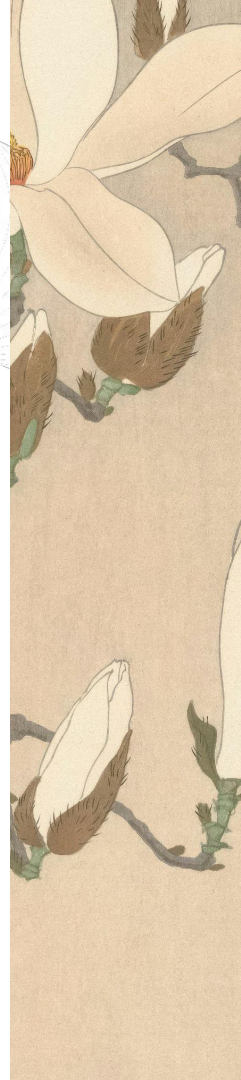
So what's the takeaway? - I

1. **Block-Level Composability is the right goal:**
    a. Cross-rollup bundles of *dependent* transactions
    b. Complicated "single-tx" level infrastructure is unnecessary
        i. Flashloans are not what we should be optimizing for
    c. Essentially no BRIDGING - burn and minting
2. **We *need* shared state to compose:**
    a. This will likely be shared sequencing or shared "Coordinating"
    b. Proof aggregation*
    c. Shared settlement*

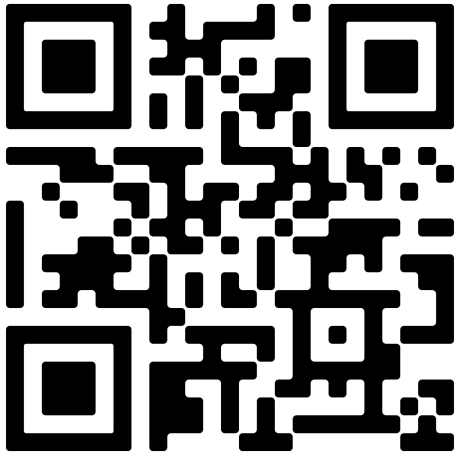


So what's the takeaway? - II

1. **Rollup clusters are great for interop:**
    a. Yes, short-term walled gardens but this is OK
2. **Open Questions:**
    a. Coordinating between rollups with diff block times
    b. Native asset transfer is risky
    c. Cross-chain MEV
    d. Superbuilders vs. Shared Coordination (subtle diff)



Future Outlook



We'll likely see:

1. Many different solutions
    a. Intents/traditional bridges using this infra
    b. Trustless interop with fewer capabilities
2. Walled garden rollup ecosystem
    a. Which isn't such a bad thing..

Dev experience:

1. Not yet clear - likely most of this abstracted away, but TBD
2. May require subtle changes to have control over composability