Using a hybrid UTXO and account-based state model in a ZK rollup

Bobbin Threadbare
Polygon Miden
Goal

Build a **scalable decentralized** rollup with **privacy-enabling** architecture
What is a decentralized Rollup?

Security inherited from Ethereum

Separate L2 chain with its own consensus mechanism

Permissionless set of operators

Ethereum
Challenges of a decentralized rollup

- Consensus mechanism
- Execution bloat
- State bloat

Topic of this talk
What is execution bloat?

All transactions executed by the network

- Block producers execute all transactions in a block
- All other nodes re-execute all transactions in a block
What is state bloat?

State size grows with time

- Nodes need full state to validate blocks
- Nodes need full state to produce new blocks
Why are execution and state bloat bad?

Execution bloat
State bloat

Centralization
Need powerful machines

Less privacy
Everyone sees everything

Not sustainable
Ever growing state
What we want to achieve

<table>
<thead>
<tr>
<th>Minimize execution bloat</th>
<th>Minimize state bloat</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Transactions executed only once</td>
<td>Can be done with ZKPs</td>
</tr>
<tr>
<td>• Transactions executed concurrently by distinct actors</td>
<td>Requires concurrent state model</td>
</tr>
<tr>
<td></td>
<td>• No need to know the full state to validate blocks</td>
</tr>
<tr>
<td></td>
<td>• No need to know the full state to produce blocks</td>
</tr>
</tbody>
</table>
## State model options

<table>
<thead>
<tr>
<th>Account-based state</th>
<th>UTXO-based state</th>
</tr>
</thead>
<tbody>
<tr>
<td>Great for expressive smart contract</td>
<td>Great for concurrent transaction execution</td>
</tr>
<tr>
<td>Not great for concurrent transaction execution</td>
<td>Needed for anonymity</td>
</tr>
<tr>
<td>Bad for anonymity</td>
<td>Not great for expressive smart contracts</td>
</tr>
</tbody>
</table>
Our approach

Account model +
UTXO model +
ZK proofs

Actor-based model with concurrent off-chain state
transaction model
Actor model

- Actors are state machines with “inboxes”
- Actors communicate via message passing
- Messages are produced and consumed asynchronously
Actor model in Miden

- Accounts maintain state and expose interface methods (Miden VM programs)
- Notes carry assets and specify a “spend script” (Miden VM program)
- Two transactions are needed to move assets between two accounts
Anatomy of a transaction

- Executed against a single account
- Consumes 0 or more notes
- Produces 0 or more notes
Executing transactions

- A note is consumed by executing its script
- Note script can call account’s interface methods
- Account methods can modify account’s state and create new notes
- Note scripts are executed sequentially one after another
Proving transactions

- Correctness of tx execution is proven with a STARK proof
- STARK proofs for all transactions are generated in parallel
Building a block proof

Recursive proof aggregation

Must be done by network (e.g., by the block producer)

Recursive proof aggregation

Can be done locally

Block proof

batch proof

\[ \text{tx proof} \]

\[ \text{tx proof} \]

\[ \text{tx proof} \]

\[ \text{tx proof} \]

\[ \text{tx proof} \]

\[ \text{tx proof} \]
Local vs. network transactions

network: block producer executes and creates proof

Local: user prepares, executes, and creates proof
Handling shared state

• Two users independently execute **tx1** and **tx2** which create notes 1 and 2
• Block producer creates and executes **tx3** which consumes notes 1 & 2 and outputs notes 3 and 4
• Two users independently execute **tx4** and **tx5** which consume notes 4 an 5
## Transaction mode comparison

<table>
<thead>
<tr>
<th>Feature</th>
<th>Local execution</th>
<th>Network execution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Can be used with shared state</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Can be private</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Client hardware requirements</td>
<td>High</td>
<td>Low</td>
</tr>
<tr>
<td>Fees</td>
<td>Low</td>
<td>Higher</td>
</tr>
</tbody>
</table>
state model
Miden rollup state

State $n$

- Account DB
- Notes DB
- Nullifier DB

Updated accounts
Created notes
Nullifiers of consumed notes

Block $n$

State $n+1$

- Account DB
- Notes DB
- Nullifier DB
Account DB

Account DB stores **current state of all accounts**

For accounts with **on-chain state**, the entire state is stored by the nodes

For accounts with **off-chain state**, only the account hash is stored by the nodes

**Sparse Merkle tree**  
(account id → account hash)

![Sparse Merkle tree diagram](image-url)
Notes DB stores **all notes ever created**

Notes can be added to the MMR even if **most nodes are discarded**

**Inclusion witnesses never become stale**, but they may need to be extended.
Nullifier DB keeps track of all consumed notes

Nullifiers are organized into epochs - e.g., 4 - 6 months

Nodes are expected to keep nullifiers for last 2 epochs
# Miden state growth drivers

<table>
<thead>
<tr>
<th></th>
<th>Account DB</th>
<th>Notes DB</th>
<th>Nullifier DB</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Primary:</strong></td>
<td>number of accounts with on-chain state</td>
<td>number of unconsumed public notes</td>
<td>throughput (TPS)</td>
</tr>
<tr>
<td><strong>Secondary:</strong></td>
<td>number of accounts</td>
<td>number unconsumed notes</td>
<td><strong>nullifier</strong></td>
</tr>
<tr>
<td><strong>Pruning:</strong></td>
<td>discard on-chain account data (but retain account hash)</td>
<td>discard on-chain note data</td>
<td><strong>epoch length</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><strong>n/a</strong></td>
</tr>
</tbody>
</table>
conclusion
# Flexible transaction modes

<table>
<thead>
<tr>
<th>Network execution</th>
<th>On-chain data</th>
<th>Off-chain data</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Public transactions</td>
<td>Stateless transactions</td>
</tr>
<tr>
<td></td>
<td>Local transactions</td>
<td>Private transactions</td>
</tr>
</tbody>
</table>
Addressing execution bloat

No re-execution
All transactions, including network transactions, are executed only once

Concurrent processing
Transactions can be processed concurrently by distinct network participants

Local execution
Transactions not affecting accounts with shared state can be executed and proven locally
Addressing state bloat

**Dynamic pruning**
Block producers can independently decide which parts of the state to keep.

**Light verifying nodes**
Verifying nodes can discard vast majority of the state (i.e., the nullifier database).

**State size driven by throughput**
State size depends primarily on the current TPS rather than total number of accounts or notes.
thanks
Using a Hybrid UTXO and Account-Based State Model in a zkRollup

Bobbin Threadbare
Project lead, Polygon Miden
Section 1 title here.
Section 1 title here.

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

- Sollicitudin
- Consectetur
  - Condimentum
    - Magna
    - Ligula
Section 1 details with an image.
Enter title here.

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.
Enter your main point / statement here.

Section 1 details with a main point. Enter title here.

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.
Section 2 title here.
Section 2 title here.

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

- Sollicitudin
- Consectetur
  - Condimentum
    - Magna
    - Ligula

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

- Sollicitudin
- Consectetur
  - Condimentum
    - Magna
    - Ligula
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Section 2 details with an image.
Enter title here.
Section 2 details with a main point. Enter title here.

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.
Section 3 details with a main point. Enter title here.

Enter your main point / statement here.

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.
Section 4 title here.
Section 4 details with a main point. Enter title here.

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.
Enter your main point / statement here.
Here’s the timeline.

Event 1
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam.

Event 2
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam.

Event 3
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam.
Thank you!

Your Name
Your title, your organization
email@emailaddress.com
@twitterhandle