Lisk Interoperability

Cross-chain interactions between homogeneous state machines

By Alessandro Ricottone
Outline

- Cross-chain certification
  - General overview and comparison to industry
- The Lisk ecosystem
  - Connecting to the ecosystem
  - State model of a Lisk chain
- Cross-chain updates
- Cross-chain messages
10000 Feet View of Interoperability

- Multi-shards architecture
  - all chains share same state which is partitioned among them
  - shards submit blocks to mainchain validators that verify all state transitions
  - the whole system reorganizes together
  - example: Polkadot, Ethereum

- Independent-chains architecture
  - each chain has a separate state and state transition functions
  - chains have an independent set of validators that verify and finalize blocks
  - each chain can reorganize independently
  - example: Cosmos, Lisk
Cross-chain Certification

- Chain 1 trusts information from chain 2 because of **certificates**
  - sent by relayers
  - authenticate **state** of chain 2
  - contain **signature** of active validators
  - update about **future validators**

Certificates attest to the state transitions on chain 2.
Lisk Cross-chain Certification

- Mainchain trusts information from sidechain because of **certificates**
  - created from block headers
  - certificates are included in **cross-chain updates** together with **cross-chain messages**
  - contain **BLS signature** of active validators
  - this interaction is symmetric

**cross-chain updates** contain certificates and **cross-chain messages**
# Cross-chain Certificates

<table>
<thead>
<tr>
<th>Certificate</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Block ID</td>
<td>identify block header</td>
</tr>
<tr>
<td>Height</td>
<td>used to validate certificate</td>
</tr>
<tr>
<td>Timestamp</td>
<td>used to validate certificate</td>
</tr>
<tr>
<td>State root</td>
<td>authenticates state of sending chain</td>
</tr>
<tr>
<td>Validators hash</td>
<td>authenticates set of future validators</td>
</tr>
<tr>
<td>BLS signature</td>
<td>aggregated signature of current validators</td>
</tr>
</tbody>
</table>

- **Block ID**: identify block header
- **Height**: used to validate certificate
- **Timestamp**: used to validate certificate
- **State root**: authenticates state of sending chain
- **Validators hash**: authenticates set of future validators
- **BLS signature**: aggregated signature of current validators
Lisk Ecosystem

- Gamingchain
- Exchange
- Prediction Market
- Oracle

Direct channel
All sidechains register on the mainchain

- **Name** of the sidechain
- **ID of the genesis block** of the sidechain
  - Used to compute the sidechain network ID
- **Sidechain validators** to sign first cross-chain update from sidechain
Mainchain Registration

Afterwards mainchain registers on the sidechain

- The sidechain own **chain ID** and **name** computed during the sidechain registration
- **Mainchain validators** to sign first cross-chain update from mainchain
# Registration Process

<table>
<thead>
<tr>
<th>Chain account</th>
</tr>
</thead>
<tbody>
<tr>
<td>chain ID</td>
</tr>
<tr>
<td>name</td>
</tr>
<tr>
<td>network ID</td>
</tr>
<tr>
<td>last certificate height</td>
</tr>
<tr>
<td>timestamp</td>
</tr>
<tr>
<td>state root</td>
</tr>
<tr>
<td>validators hash</td>
</tr>
<tr>
<td>status</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Validators</th>
</tr>
</thead>
<tbody>
<tr>
<td>[active validators]</td>
</tr>
<tr>
<td>bls key</td>
</tr>
<tr>
<td>bft weight</td>
</tr>
<tr>
<td>certificate threshold</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Channel</th>
</tr>
</thead>
<tbody>
<tr>
<td>inbox</td>
</tr>
<tr>
<td>root</td>
</tr>
<tr>
<td>size</td>
</tr>
<tr>
<td>append path</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>outbox</th>
</tr>
</thead>
<tbody>
<tr>
<td>root (also stored separately)</td>
</tr>
<tr>
<td>size</td>
</tr>
<tr>
<td>append path</td>
</tr>
</tbody>
</table>

| partner chain outbox root |
| message fee token ID |

[Validator List]
Inbox and Outbox

- **Inbox** and **outbox** are append-only Merkle trees
- Cross-chain messages sent to a chain are appended to the outbox
- The same message is appended to corresponding inbox when received

![Diagram showing inbox and outbox Merkle trees](image-url)
State Tree

- Every chain uses a sparse Merkle tree for its state, called **state tree**
- The root of this tree is called **state root**
- SMTs allow to prove efficiently that an entry has a certain value

```
account balance Bob
```
State Tree

- The state tree is split into several **module subtrees**
- Each module defines generic key-value map
- Application modules implement custom logic

![State Tree Diagram]

- application module
- token module
- interoperability module
State Tree

- Interoperability module contains the channel state
- Cross-chain messages are validated from the outbox root to the state root
Cross-chain Updates

- Transactions trigger CCMs
- Validators create certificates from finalized blocks
- Relayers collect certificates and CCMs in a CCU
- Relayers post the CCU on the receiving chain
- CCMs are executed directly on receiving chain
Mainchain Routing

- A sidechain will exchange CCUs with the Lisk mainchain
- CCMs targeting other sidechains will be forwarded by the mainchain
Cross-chain Messages

- **Nonce**: identifies CCM in the ecosystem
- **Module ID and asset ID**: identifies transaction
- **Params**: contains transaction parameters
- **Fee**: paid to the relayer
- **Sending chain ID**: identifies sending chain
- **Receiving chain ID**: identifies receiving chain
- **Status**: extra information about CCM
Cross-chain Token Params

- **Amount**: the amount of transferred tokens
- **Token ID**: used to identify the transferred tokens
- **Recipient address**: used for crediting the tokens
- **Sender address**: used for reimbursement if message fails
- **Data**: data field accompanying the transfer
CCM can Trigger a CCM

- Message is sent to receiving chain
- Processing message generates new message
- Example: Message triggers an error
Thank you!