Upgrading Aragon Voting

Jorge Izquierdo - Aragon One, CTO
EthCC - March 6th, 2019
Aragon v0.5 beta launch in Rinkeby

NEXT WEEK

Ethereum Community Conference
March 8-10, 2018 | Paris, France
Aragon

- DApp to create and manage decentralized organizations
- Built on the Ethereum VM
- Available on the web, desktop and mobile
- 353 DAOs running on the Ethereum mainnet

app.aragon.org
Aragon Voting v1

- Uses MiniMeToken as the only supported governance token
- Simple majority voting
- +$1m secured with Voting
- Voting is the default root authority for Aragon DAOs
Forwarding

- aragonOS primitive for encoding complex permissions
- Users express their intent and Aragon figures out how to make it happen
- Forwarders can execute aragonOS' EVMScripts when some arbitrary logic checks out
Aragon Voting v2
Aragon Voting v2

1. Remove MiniMe requirement
Aragon Voting v2

1. Remove MiniMe requirement
2. Really cheap voting
1. Support for all ERC20 tokens*

* tokens that use a standard data structure for storage
Aragon Voting v2

1. Support for all ERC20 tokens*
2. Off-chain vote aggregation

* tokens that use a standard data structure for storage
The double voting problem

- ERC20 tokens are fungible

- Can't detect if one particular token already voted, unless:
  - Tokens are locked during the vote location (staking)
  - Token balances can be snapshotted (MiniMeToken)
EVM Storage Proofs

- Prove what the value a given storage slot in a contract was at a past block height
- Derive storage values from a trusted block hash
- Uses `eth_getProof` (EIP1186)
- For Voting, prove the token balance of a holder at a snapshot block

[github.com/aragon/evm-storage-proofs](https://github.com/aragon/evm-storage-proofs)
Account proof

Block hash → storage root

- Use `BLOCKHASH` to get block hash for block X
Account proof

Block hash → storage root

• Use `BLOCKHASH` to get block hash for block X
• Submit the block header and verify its hash matches

---

github.com/aragon/evm-storage-proofs
Account proof

Block hash → storage root

- Use `BLOCKHASH` to get block hash for block X
- Submit the block header and verify its hash matches
- Extract the state root from the block header

Block hash → storage root

[Diagram showing the process]

github.com/aragon/evm-storage-proofs
Account proof

Block hash → storage root

- Use **BLOCKHASH** to get block hash for block X
- Submit the block header and verify its hash matches
- Extract the state root from the block header
- Submit merkle proof of the account state in the state trie

```
github.com/aragon/evm-storage-proofs
```
Account proof

Block hash → storage root

- Use BLOCKHASH to get block hash for block X
- Submit the block header and verify its hash matches
- Extract the state root from the block header
- Submit merkle proof of the account state in the state trie
- Verify merkle proof against the state root

[Diagram showing block hash leading to storage root]

github.com/aragon/evm-storage-proofs
Account proof

Block hash → storage root

- Use `BLOCKHASH` to get block hash for block X
- Submit the block header and verify its hash matches
- Extract the state root from the block header
- Submit merkle proof of the account state in the state trie
- Verify merkle proof against the state root
- Extract storage root from the account state

[Diagram showing block hash to state trie]

github.com/aragon/evm-storage-proofs
Storage proof

Storage root $\rightarrow$ slot value

- With an authenticated storage root
- Submit merkle proof of the storage slot value in the storage

github.com/aragon/evm-storage-proofs
EVM Storage Proofs

github.com/aragon/evm-storage-proofs
EVM Storage Proofs for ERC20s

**Upsides**

- Use all existing ERC20 tokens for Voting
- Snapshotting doesn't impose a cost for all transfers

**Downsides**

- Verifying a full proof uses ~500,000 gas
- BLOCKHASH can still only get the last 256 blocks
- Breaks EVM abstraction by checking storage directly, use carefully

[github.com/aragon/evm-storage-proofs](http://github.com/aragon/evm-storage-proofs)
Optimistic contracts

- Offloading computation heavy operations from the network
- Ensure computation correctness with bonded relayers instead of repeating it in every node
- Naïve implementation can scale up to 80x*: use 100k gas for computations that would take the block gas limit (8M)

* for an optimistic execution with no input
Commit to the result of executing a pure function with some input

Challenges can be done during a certain time window, which executes the computation and compares to the committed result.

Challenger is awarded the deposit of the relayer on a valid challenge.

After the challenge window has passed, the result is taken as valid.
Voting Relay Protocols (VRP)

- Family of layer 2 protocols that aggregate votes off-chain and submit partial tally **optimistically**
- Update the tally once for many votes without verifying storage proofs
- Individual votes are multiple orders of magnitude cheaper to cast
- Vast design space with different trade-offs
- Voters just sign their vote message and send it to the relayer
Simple VRP (SVRP)

- VRP, except that **only one relayer** is allowed to relay votes at a time
- The relayer slot can be centrally appointed or auctioned
- The relayer stakes tokens as collateral that can be slashed if a fraud proof is submitted
- The relayer may ask for a fee payment to include a vote or sponsor voting
SVRP and centralization

- If the relayer doesn't return a signed inclusion commitment, the voter can submit the vote directly.
- If the relayer commits to relaying a vote and they don't, they are slashed.
- Data required to prove fraud is available through EVM logs.
- If someone is monitoring, the relayer will always be slashed if submitting fraudulent votes.
The SVRP bottleneck

- Naïve optimistic relies on the data required to submit fraud proofs being available with EVM logs
- In SVRP, the input data is the array of vote messages being relayed
- Cost per input byte = \(88\) gas
- 1 vote message = \(68\) bytes
- Hard cap of 938 votes relayed per batch (80% of the block gas limit)
- Compared to Voting v1, SVRP scales voting by \(10.72x\)

**VRP messages**

All messages must be RLP-encoded before sending them over any transport protocol or submitting them on-chain.

**Vote message**

Components:
- Voting app address identifier (4 bytes: `bytes4(hash(address))`)
- Proposal id (3 bytes: 14,706,125 proposals per Voting app)
- Casted vote (1 byte: 256 options)
- Voter balance (16 bytes, claimed user balance in the voting app’s token at the snapshot block of vote id)
- Voter ECDSA signature for message hash (64 bytes, with the trailing bit trick)

Total batch size: 88 bytes
The data availability problem

• Just submitting the hash on-chain and proving that the hashed data is available is impossible from a smart contract's worldview

• If the data is actually not made available, fraud couldn't be stopped, as the data is required for challenging

• To solve this issue, relayers could be required to submit on demand the full data blob to the chain so a challenge can happen, which can be used for griefing relayers
Optimistic data availability

- Used for Availability VRP (AVRP)
- A set of \( N \) validators is selected, but only one of them can relay at a time
- Validators receive relay requests and send them to other validators to receive a signed commitment that they have the data and result
- Later, a validator is randomly appointed as the relayer and they can relay it if they have signatures of \( 2/3 \) validators
- Only one honest validator is needed for the integrity of the system (unless \( 2/3 \) validators are malicious and withhold data)
SVRP is now open source

github.com/aragon/srvp
Voting v2: TLDR

- EVM Storage Proofs
  - Use any ERC20 to vote (no staking required)
  - Bring any existing token to Aragon for governance
  - [github.com/aragon/evm-storage-proofs](github.com/aragon/evm-storage-proofs)

- Voting Relay Protocols
  - Aggregate votes off-chain and relay partial tallies
  - Voters don't need ETH, and costs are reduced by 10-1000x
  - [github.com/aragon/svrp](github.com/aragon/svrp)

- Research discussions: [forum.aragon.org](forum.aragon.org)
- Come work with us: [wiki.aragon.org/jobs](wiki.aragon.org/jobs)
Thanks

aragon.org

@AragonProject @AragonOneTeam @izqui9