Blind-OVOTE

off-chain Anonymous Voting with on-chain binding Execution on Ethereum

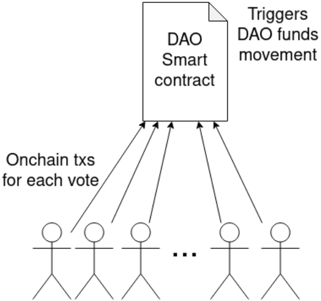
2022-11-18
Ethereum Dev Barcelona Meetup

ARAGON ZK Research

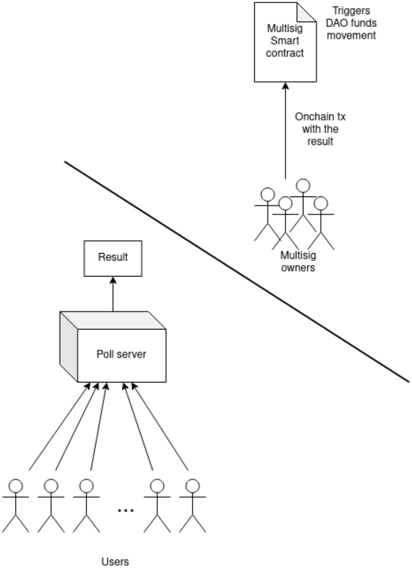
Motivation

- High Ethereum mainnet gas costs
- Current status of DAOs
  - People vote in a poll
  - A small group (usually <10 people) vote in the multisig contract to execute the result of the poll
- **Users anonymity** without costly recursive proof systems
  - Want to define who can vote
  - Without revealing who is voting what

Direct on-chain voting



Poll + multisig system

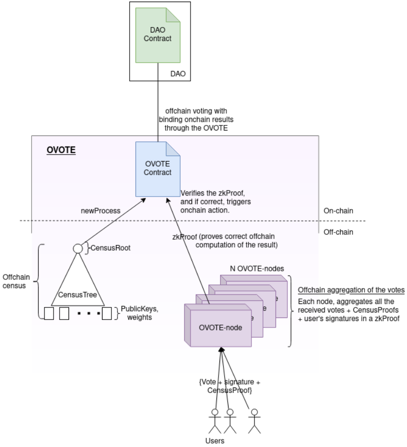


Current systems

Current systems used in production are either on-chain and expensive, or

- non-trustless: users must trust that the operator does not manipulate the results
- non-binding: there is no cryptographic proof of the result → users must trust a someone to do the on-chain execution (usually through a multisig of <10 members)
- non-anonymous: users voting choices are public and everybody can know what the others are voting. And in cases where the data is not public, it could still be leaked

OVOTE, first step


DAO Contract
DAO
offchain voting with binding onchain results through the OVOTE
OVOTE
OVOTE Contract
newProcess
Verifies the zkProof, and if correct, triggers onchain action.
On-chain
Off-chain
zkProof (proves correct offchain computation of the result)
CensusRoot
Offchain census
CensusTree
PublicKeys, weights
N OVOTE-nodes
OVOTE-node
Offchain aggregation of the votes
Each node, aggregates all the received votes + CensusProofs + user's signatures in a zkProof
{Vote + signature + CensusProof}
Users

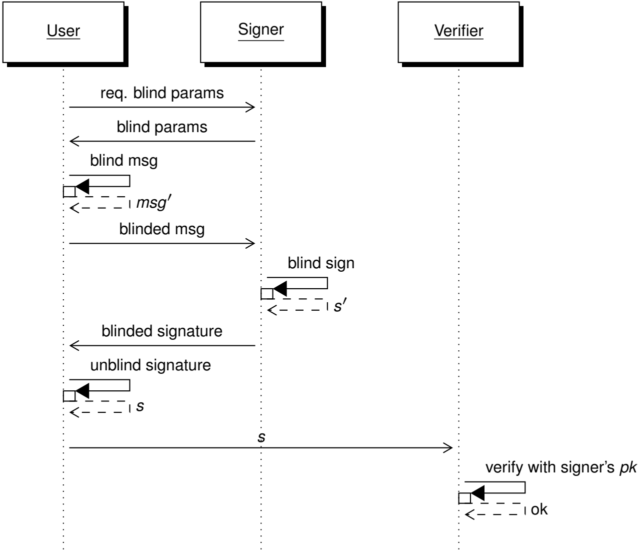
In July 2022 we proposed OVOTE: off-chain Voting with on-chain Trustless Execution

- MerkleTree-based off-chain census
- Users vote off-chain
- RollupNode aggregates all the votes & sigs verification and produces a zkSNARK proof
- Results + zkSNARK proof are verified on-chain in Ethereum

Non-anonymous: RollupNode knows the relation between voters and the votes.

OVOTE paper: https://research.aragon.org/docs/ovote

Blind Signatures over EC



Signature scheme

Schnorr Signatures

key generation: $x \in \mathbb{Z}_p$, $X = xG$

signing: $r \in \mathbb{Z}_p$,

$$\begin{cases} R = rG \\ s = r + \mathcal{H}(R, m) \cdot x \end{cases}$$

$\sigma = (s, R)$

Schnorr Blind Signatures

key generation: $x \in \mathbb{Z}_p$, $X = xG$

req params: $r \in \mathbb{Z}_p$, $R' = rG$

blind params: $\alpha, \beta \in \mathbb{Z}_p$, $R = R' + \alpha G + \beta X$

blind msg: $m' = \mathcal{H}(R, m) + \beta$

blind sign: $s' = r + m' \cdot x$

unblind sig: $s = s' + \alpha$

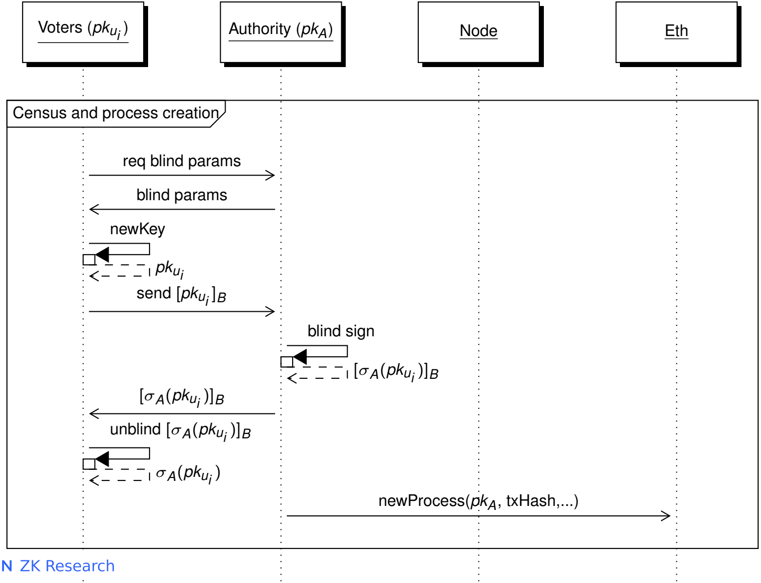
$\sigma = (s, R)$

Verification: $sG == R + \mathcal{H}(R, m)X$

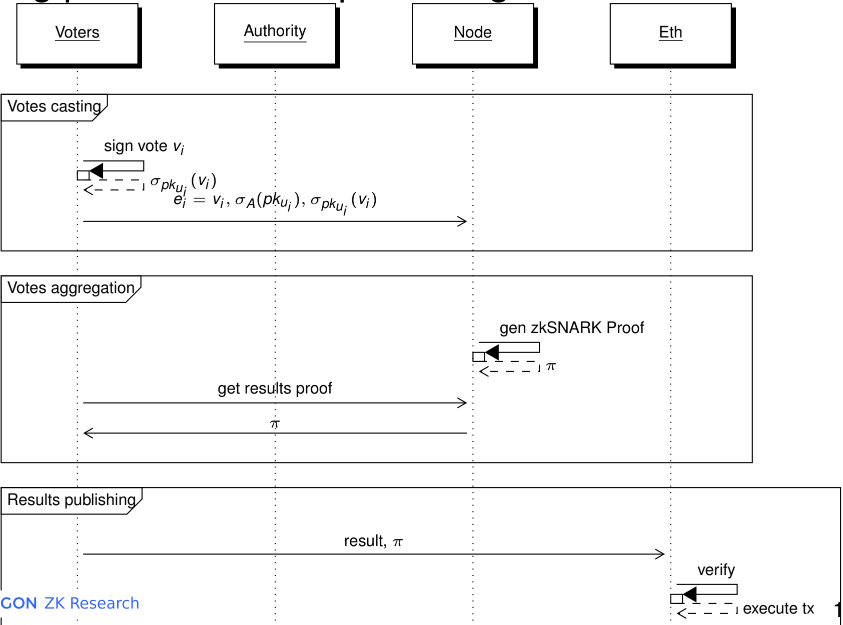
zkSNARK

- Focus on Ethereum: we want to end up triggering onchain transactions
  - Groth16 over BN254
    - BabyJubJub for keys
    - Poseidon for hashing
- Using arkworks toolset

Census and process creation



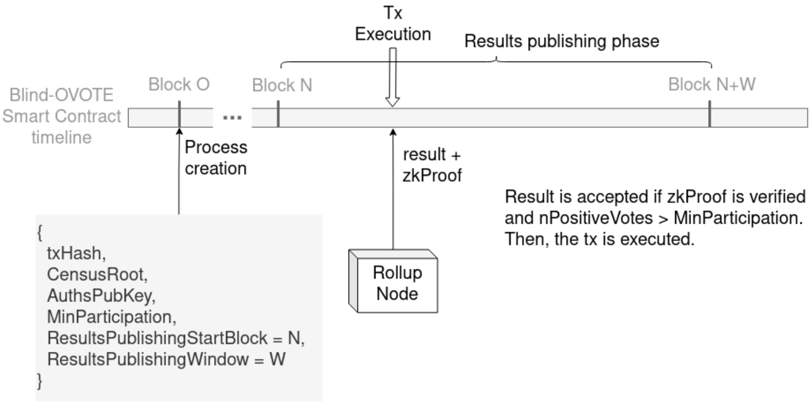
Voting phase & results publishing



Multiple Authorities

- To prevent a single Authority from introducing fake voters
- Voters require *N* blind-signatures from different non-colliding Authorities
- different recognized parties of the community could be the doing the signatures

Voting process timeline



Tx Execution

Results publishing phase

Blind-OVOTE Smart Contract timeline

Block O

Block N

Block N+W

Process creation

result + zkProof

Result is accepted if zkProof is verified and nPositiveVotes > MinParticipation. Then, the tx is executed.

Rollup Node

```
{
  txHash,
  CensusRoot,
  AuthsPubKey,
  MinParticipation,
  ResultsPublishingStartBlock = N,
  ResultsPublishingWindow = W
}
```

> Very similar to OVOTE process timeline.

Results publication

- Multiple nodes
- Blind-OVOTE Contracts:
  - zkSNARK proof verification takes 250k gas
  - results publication takes 350k (including the zkSNARK proof verification)

Signatures verification SNARK circuit

Input	Description
m	array containing the signed values
s	signature, $\{s, R\}$, where $R \in \mathbb{G}$
X	public key

Table 1: Blind & non-blind signatures verification circuit public and private inputs.

The circuit checks the equation

$$sG == R + h(R, m)X$$

where $h(m)$ is a Poseidon hash [1] over $\mathbb{Z}_p$, with an array of $\mathbb{Z}_p$ elements as input.

Since the circuits are computed to be proven over the Bn254 curve, we use the BabyJubJub elliptic curve for the signature scheme, thus $\mathbb{G}$ is the group of the BabyJubJub curve and $\mathbb{Z}_p$ is the base field of it, which is the scalar field of the Bn254 curve.

Blind-OVOTE SNARK circuit

Public inputs		Private inputs	
Input	Description	Input	Description
chainID	hardcoded in the Contract deployment	pk_{u_i}	users public keys
pID	processID, determined by process creation	w_i	weight of each pk_i
pk_{A_j}	Authority's public key, determined by process creation	v_i	votes values
R	result	σ_{A_j}	authority signatures over $pk_{u_i} + w_i$
		σ_{u_i}	users signatures over v_i

Table 2: Blind-OVOTE circuit public and private inputs.

The checks defined by the circuit constraints are:

i. $pk_{u_i} + w_i$ are signed by pk_{A_j}

ii. v_i is signed by pk_{u_i}

iii. $v_i \in 0, 1$

iv. $R = \sum v_i \cdot w_i$

v. There are no repeated pk_{u_i}

Properties

- *Universal verifiability* results & proof verfiable by any actor.
- *Unforgeability (tamper-evident)*: nobody can manipulate the votes or add fake votes.
- *Trustless*: Thanks to the previous properties, no-one needs to trust in any specific party.
- *Binding execution*: Due to the universal verifiability property, the proof verification can trigger on-chain actions (e.g. moving funds of a DAO), directly from the voting process result.

Properties 2/2

- *off-chain/gasless voting*: the RollupNode aggregates the computation and verification of all the off-chain votes, signatures and census-proofs, in a succinct validity proof. The only transactions executed on-chain are the *process creation* and the *results publishing*.
- *Scalability*: Thousands of votes can be aggregated in a single Ethereum tx.
- *Chain agnostic census*: The census is build off-chain, and the proof of correct results computation can be published into any EVM chain (furthermore, into any chain that supports Pairing computation). So a Blind-OVOTE census could be used in Ethereum mainnet, but also in other chains.
- *User anonymity* By the usage of blind signatures, user identity is kept anonymous in front of all the different parties (Authority, Rollup Node, Ethereum).

Drawbacks

- Census is not auditable:
    - The census creator (Authority) could blind-sign extra public keys, introducing fake voters.
    - This in OVOTE was not a problem due the use of MerkleTrees for the census. Here we use BlindSignatures for anonymity, so our census is not auditable.
    - But: we **can mitigate** this by having mutliple non-colliding Authorities for the census creation (see slide "Multiple Authorities").

Practical implications

- Fewer trust assumptions than for the current in use → broader use cases.
- Not an ideal solution, but one step forward, with working code ;-)

Implementation

Implementation of this scheme using arkworks:

- github.com/aragonzkresearch/**ark-ec-blind-signatures**:
  Blind signatures over elliptic curve implementation (native & R1CS constraints)
- github.com/aragonzkresearch/**blind-ovote**:
  Blind-OVOTE scheme implementation, contains the library to be used in Voter's browsers, Authority server, and Rollup node

Blind-OVOTE main take aways

- L2 validity rollup combined with blind signatures over elliptic curves
  - off-chain **anonymous** Voting with on-chain trustless execution on Ethereum
  - Main idea: votes are anonymous and aggregated off-chain, and proved on-chain through a zkSNARK proof
  - Resulting in **constant gas costs while scaling up to thousands of voters** through a single Ethereum transaction
  - Drawbacks: Census creation
- Next step, recursive proofs
- **Blind-OVOTE technical document available at https://research.aragon.org/docs/blind-ovote**