For background, see: Minimal anti-collusion infrastructure

One of the challenges of MACI is that it requires data to be submitted on-chain for each vote, which incurs significant transaction fees. This post suggests a mechanism by which votes can be off-chain by default; a vote would only need to go on-chain if the coordinator is actively attempting to censor voters.

When a user makes a vote, take the following steps:

1. The user locally generates their vote, as in regular MACI
2. The user sends their vote to the coordinator
3. The coordinator replies with a signature, specifying which position of the next batch the user's vote will be included at
4. When the coordinator submits their next batch, they submit only a hash to the chain. The hash must be the Merkle root of all votes that have been sent to the coordinator since the previous round. The coordinator is also required to publish the full set of votes (eg. on IPFS).

The ZK-SNARK enforces that the final vote results are the result of processing all submitted votes, including votes that have been published via this hash mechanism.

Users have access to two extra features:

1. They have the ability to submit a vote directly to chain. The coordinator is forced to process both types of votes (included via hash, and included directly)
2. If a user has a cryptographically signed promise, they can publish that promise on chain as a challenge. Any future message that the coordinator submits (a batch or the final result) must come with a SNARK proving that the correct votes have been included at the provided challenge positions.

This ensures the following properties:

- Assuming an honest coordinator, onchain costs go down to O(1) per batch period
- Censorship resistance is maintained, because users can go onchain worst-case
- We mitigate attacks where the coordinator pretends to accept a vote, but then fails to include it, hoping that most voters will not notice or re-open any kind of software daemon, by providing signatures:
- If a user fails to get a signature immediately, they go straight to voting onchain.
- If a user gets a signature, and they do come back to check after the batch, they can check on IPFS (or ask the coordinator) for the Merkle branch associated with their vote; if it does not match their signature, they can publish their signature to chain, which effectively halts the entire vote and prevents it from giving a result. Hence, trying to censor even one voter becomes extremely risky for a coordinator.

- If a user fails to get a signature immediately, they go straight to voting onchain.
- If a user gets a signature, and they do come back to check after the batch, they can check on IPFS (or ask the coordinator) for the Merkle branch associated with their vote; if it does not match their signature, they can publish their signature to chain, which effectively halts the entire vote and prevents it from giving a result. Hence, trying to censor even one voter becomes extremely risky for a coordinator.