Thanks Trenton Van Epps for review and suggestions

Intro

There has been interest in using DAOs to make decisions for a long time. Private DAOs have not really been investigated.

Even if all users in a DAO are anonymous, it's still possible to link a single DAO member's actions together. This is problematic.

Recently the idea of using a mixer to mix voting tokens between every vote has been proposed. However, this seems to be optional privacy and leads to the same problems of anonymity sets that we have with mixers. These problems being that your privacy is based upon other people opting to also mix their tokens.

It would be interesting to make an anonymous DAO. Where actions are not attributable to members unless they want. Or all other members want to reveal it.

Here we describe how to do this with semaphore.

Semaphore

Semaphore is an anonymous signaling mechanism. We create an account for every member of the DAO. A smart contract is then used to allow these members to signal anonymously.

A snark is used to

1. Prove membership of a merkle tree
2. Calculate a nullifier.

The

nullifier = hash(external_nullifier, identity_nullifier, identity_path_index)

identity_nullifier, identity_path_index

are unique to the user.

This means that for each external_nullifier

each user has a unique nullifiers. Furthermore, unless you can reverse the hash function you cannot link a signal to a given user.

Check https://github.com/kobigurk/semaphore for more info.

In the rest of this post we will use the smart contract to check things for specific external_nullfiers knowing that when we change one it will result in a different signal.

DAO construction

The DAO is constructed by making a semaphore group. The semaphore group is able to hold votes using semaphore signals.

Add proposals

The smart contract lets anyone create a proposal as long as they provide a valid proof of membership in the group. Also we prevent spam proposals by requiring that external_nullifer = "proposal" + epoch

. The smart contract only allows unique nullfiers.

For example epoch can be a month long, allowing each participant to make one proposal per month.

Each proposal is created and assigned a proposal ID.

Vote on proposals

To vote members make a semaphore signal with external_nullifier = hash(proposal_id)

this will allow each member a single vote for each proposal.

The smart contract only allows unique nullfiers to be included as valid votes.

After the voting period is over the smart contract executes the proposal if it passed.

Example.

Imagine my identity_nullifier = 0

and identity_path_index = 0

if i try to vote on proposal with proposal_id = 9

the external_nullifier = 9

. My nullifier = hash(external_nullifier, identity_nullifier, identity_path_index)

so if i try and vote twice my nullifier will always equal hash(0,0,9)

So in order to double vote i need to either

1. Find a collision in the hash function.
2. Break soundness of the snark.

Because the nullifier is calculated inside the snark. Proof that it was done correctly is part of the snark proof.

Gas payments

Each transaction requires gas to participate with the system. In order to overcome this we can use the burn relay to refund users their gas.

This requires that someone pay the gas for all the participants. In order to subsidize this we can have a proposal fee. Where to create a proposal you have to pay the gas for everyone to vote on it.

We can then use Burn relay registry: Decentralized transaction abstraction on layer 2 which is being implemented https://github.com/lsankar4033/micromix_relayer

Conclusion

Here we introduced anonDAO based upon semaphore. A group where the members are known but actions are not attributable to any member. Future work is required to implement this and test it.

The semaphore circuit has already been build. Its just a matter of linking a sample DAO with this circuit and building the UI.

Its important to note that this construction is private but not collusion resistant where someone can prove they took an action which allows them to be trutlessly bribed. In order to overcome this you would need something like Minimal anti-collusion infrastructure proposed by Vitalik and being implemented https://github.com/barryWhiteHat/maci