Abstract

Here I describe the construction providing coercion resistance voting. Voters can hide their identities from anybody, and also they cannot be bribed. Also, nobody knows the intermediate result of voting before the final result is counted.

The voter can be bribed in cases:

- cooperation of \geq \frac{C}{3}

counting systems, where C is the total number of counting systems and one voter

- cooperation of all voters and one counting center.

The system is fault tolerance in a case <\frac{C}{3}

counting centers go down.

Setup

The users' personalized accounts have rights to create unique identifiers. For example hash(secret+salt)

is published, where the the identifier is computed as hash(address+entropy)

. After then the identifiers can be conducted to other addresses anonymously (in such case as in anonymous transactions). So, we have a lot of addresses with rights to vote and the set of identifiers. And nobody knows which identifier belongs to which users. Counting centers can associate the identity with any ephemeral address that is used in voting, but this brings no information about a voter's personality.

Voting

The voters publish the messages to the blockchain. In each message, we keep \frac{2C}{3}

nonzero encrypted rows for the corresponding subset of counting centers and \frac{C}{3}

zeroes. Also, we keep the hash of the current voting vector.

The voting vector is something like [0, 0, 0..., 1, 0, 0, 0]

. We can use fixed point arithmetic and check, that \vec{v}^2 \leq 1

to validate the vector.

Each nonzero row R_i

is composite of the pointer to previous message (or to the identifier if this is the first vote) and vector \vec d_i

We need to check via the snark, that all pointers are same and \sum \vec d_i = \vec v - \vec v^*

, where \vec v^*

is voting vector from the previous message (equals zero in the 1st message case).

Counting

In case when only up to one message is pointed to any message this is simple to count all votes. \vec V = \sum \vec d_{ij}

, where all counting centers sum their parts separately, and after that we get resulting vector onchain, counting the sum of all counting centers' results.

But in real case, we have tree structure like blockchain with forks.

.

In each branching point let's consider the youngest message valid. So, here the valid branch is ACG.

When we compare any pair of nodes in one branching point (for example B and C), at least \frac{C}{3}

counting centers will know enough information to select the youngest node (C in this case).

So, counting centers may review messages one by one. If at least one counting center proof that the node is banned, we set

the flag onchain that it is banned, all counting centers update their states and do not use it and all nodes pointed to it.

After that, we can sum all remained vectors (via reduce on zkSNARK) and obtain the result.

Coercion and collusion resistance

The counting center cannot prove that the current message is the last message from the current voter. Also, the voter cannot prove that his current message is the last. It is achieved because each counting center has \frac{1}{3}

messages with unknown information for him.

Fault tolerance

Let’s consider that \sum \vec d_i r_{ij} = \vec 0

for j = 1..\frac{C}{3}

, and each square submatrix in r_{ij}

has same range as one’s size. That means that if some d_i

are abcent (the counting center send nothing), we can anyway compute total sum.