MACI and collective bribes:

MACI is a great infrastructure to prevent collusion for governance processes like voting. It offers a mechanism ensuring that no one can reliably bribe individuals.

However, bribing does not have to target individuals directly: E.g. all actors could also be bribed as a collective to favor a particular outcome. This post describes the bribe attacks targeting collectives and investigates possible solutions.

Bribing the collective

Imagine the usual MACI setup is given. We have a registry R that contains n public keys $K_1,\dots,K_n$

that are allowed to vote.

The bribing party will deploy a bribe contract that grants every public key of R a bribe b_i

, if the briber's preferred vote outcome wins.

The bribe contract can be set up completely trust-less: It can get funded before the MACI process starts, it can read the MACI outcome via smart contract calls, and hence it can distribute the grants fully trustlessly.

Assuming this bribe contract is deployed, each voter has an additional incentive to vote for the briber's preferred outcome, to increase the chance of the payout.

Let's investigate the incentives and their impacts in case of a binary election

: Each voter has the option to vote either for party A or party B and a 51% majority is needed to win the election.

For simplicity, it is assumed that each voter has an intrinsic motivation to vote for their preferred party over the other party, and this motivation or utility can be represented by a monetary value: v_i

for the i-th voter. Of course, there will be actors that are not influenced by monetary incentives, they will always vote for their preferred outcome and then their v_i

might be infinity.

Now, if $b_i > v_i$

, the optimal economic strategy for the voter is to vote for the briber's preferred party, to increase the likelihood of the higher bribe payout. Hence, the briber can choose his bribe b, such that $v_i<b$

for half of the $i \in 1..n$

If the briber's preferred party wins, the bribing cost must be at least $\Sigma_{i \in n}b = n*b$

. If $v_i =v_j$

for all i,j, this means that the bribing party has to buy out the whole utility of all voters to influence the vote.

Let's compare this to a non-MACI voting process: Let's assume that bribing individuals would be possible and that the bribing party would only pay a bribe to voters that voted for their outcome. Since each voter has $\sim 1/n$

chance of being pivotal and influencing the outcome, each voter's personal expected-value from voting is v_i/n

. Hence, as long as $v_i/n < b_i$

, the optimal strategy is to vote for the briber's preferred party. The total cost for the briber would be: $n*(b_i/n) \simeq b_i$

, assuming $v_i<b_i$

Hence, under the bottom line, MACI has increased the bribe costs by a factor of n by eliminating individual bribes and only allowing collective bribes for this particular scenario. This is valuable, as an attacker has to roughly buy out the utility of all voters to influence the vote.

However, there are still two problems in practice:

- Some applications can not guarantee that the sum of their voter's utility is higher than the value that could be extracted from their decisions. For example, many future institutions might have only a utility U for normal soulbound token holders, but the institutions have to make decisions about transactions bigger than n*U. One concrete example could be a new decentralized arbitration system - like Kleros - governed by soulbound token holders. For these applications, it is necessary to improve the ratio between bribing costs, and voter's utility.

- Some applications might have a huge set of participants for whom a particular governance decision has only low utility. In an extreme case, if half of the voters have only an \epsilon

utility for a specific governance decision, then the bribing cost is still only \epsilon *n

via collective bribes. This holds true, even if for the other half of voters, the utility of this governance process is very high, and they would be hard to bribe.

Countering group bribe attacks:

Next, let's investigate how MACI can be improved to defend better against collective bribes. At first, one observes that for MACI processes with a public outcome, collective bribes can always be started in a trustless way, with the setup described above. Secondly, the bribe payout can be improved to prevent negative karma for bribed voters: By using technology – similar to tornado cash -, bribes can be paid to the participants in a non-trackable way by giving each public voter key the ability to claim their bribes into a new address in a non-trackable manner. This could be enabled by preparing for each voter a note (cf. tornado cash notes) which can only be claimed by a zk-proof that can only be generated with the private key of the voter.

In the context of soulbound tokens, this means that bribed soulbound tokens would not receive negative karma, as no one could prove that they misbehaved.

Since collective bribes can not be easily prevented, in the following a method is proposed to increase the cost-factor of collective bribes compared to individual bribes beyond the sum of the utility of all voters. Fortunately, by introducing a small specification change, this can be archived:

Increasing the cost of group bribing attacks with fake voters:

Imagine the registry R contains not just the n public keys K_1,…,K_n

but also for each key K_i

a voting weight vector (w_{i1}, w_{i2}, … w_{in})

is given. The weights are non-public information and are only known by the maintainer of R and by the voting operator.

At the start, the registry R is empty. For adding new k

public keys to the registry, the maintainer would add s*k

accounts to the registry. The k*(s-1)

additional accounts will be fake accounts that have a voting weight vector of only zeros (0, … ,0)

. This can be done in a trust-less manner: the operator could generate a zk-proof that (s-1)*k

of the added accounts have a zero voting weight vector.

In the setting of soulbound token, the determination and addition of weights could also happen in a trustless manner: E.g. for each non-zero weight of a newly added account, there could be the requirement for a verification process that outputs a zk-proof verifying the correctness of the weight addition. Via recursion, these proofs could be used to build an overall zk-proof verifying that all weights for a registry were updated correctly.

These proven properties must be NOT publicly known about the soulbound token holder and must be only shared with the operator. If the account holder shared these properties openly and would be able to prove certain properties to a bribing party, the account holder would get bribable and hence should be removed from the registry again. The voting weight vectors can then later be used arbitrarily by the voting algorithms.

Given this setup, imagine an attacker wants to start a collective bribe. Since an external attacker can not know who of the voters in the registry are fake voters, they have to bribe all voters in the registry. Hence, for n valid voters with positive voting weights, the briber has to bribe s*n

public keys. This increases the cost of collective bribing by a factor s

, where s

is theoretically arbitrary large, but for sure will have practical limitations. If someone started a collective bribe, the registry maintainer should have access to all fake accounts and also claim the bribe rewards for the fake accounts, to incur a cost for the collective briber.

This approach is promising, as the collective bribes can be made arbitrarily ineffective by increasing the number s

.

However, this setup will only work, if the privacy of the real voters is guaranteed. If the voters are known or can be derived statistically, the protection will not work, as the briber can just bribe the collective of likely voters.

This is tricky in many applications: Image a DAO that wants to recruit their voters from the crowd of soulbound tokens holders that fulfill some criteria/metric. If the information about the criteria is public, then the briber can just bribe the soulbound addresses directly instead of the addresses in the registry R. Hence, each DAO has to source their voters from a huge set of potential accounts with non-public properties/information about these accounts. The information must be private, but still, this information must be verifiable to the registry R maintainer to facilitate the assignment of the correct voting weights.

Open questions

- Are there other ideas for protecting MACI setup from collective bribes? I am all ears!
- How important is bribe resistance against bribes of the collective in practice?

I am really curious about your thoughts.