Two weeks ago I reviewed a quadratic funding grant at DoraHacks ETH Hackathon Beijing, details here.

A parallel online BSC quadratic funding grant was hosted on HackerLink around the same time, from Jan 20 2021 to Mar 15 2021. During the time, DoraHacks promoted the event and invited cool developers and dev teams to present at DoraHacks' Bilibili live streaming channel.

There were 108 projects submitted BUIDL to HackerLink for the BSC grant, 92 registered on-chain (since the grant is running on a smart contract, projects have to be in a registry in order to receive votes and funding).

The projects covered a wide scope of current development of the blockchain space — Defi, decentralized insurance, NFT games, infrastructures, DAOs, zk rollup, decentralized social networks, decentralized content platforms, etc.

[

image

1205×657 160 KB

](https://ethresear.ch/uploads/default/original/2X/0/0d9bf7cf3a34c506ed76aff4c54f672a94aee53a.jpeg)

We made a lot of friends and had a lot of fun during the event.

Now I want to share some of our experiences. In my previous post, I have discussed topics like sybil attack, collusion, voter motivation and voting behaviors. And here I would like to further discuss these topics.

1. Mechanisms to Further Prevent Sybil Attack

We talked about how large-scale community participation and fees can depress sybil attack in the previous article. However, fees cannot stop sybil attack. People can make their calculations on how many tokens they can use to attack and obtain maximum return.

So is there a way to prevent sybil attack for on-chain quadratic voting? I don't think so (just as it's also impossible in centralized setting). However, we can implement some mechanisms in our future rounds to further improve the situation.

Let's discuss four methods.

1. Increase sybil attack cost with staking requirements

In a nutshell, the real on-chain ID is your asset. We can create a smart contract and ask voters to stake certain amount of tokens into the smart contract by certain time, and be eligible to vote. After round ends, the smart contract releases the tokens and users can claim them back.

For example, we require each account to stake 1 ETH to be eligible to vote, if someone wants to create 10 voting addresses, 10 ETH need to be staked. This naturally prevents sybil attack of some extent.

However, staking requirements increase the entry barrier. If someone does not have 1 ETH, the mechanism stops the person from voting at all.

Another problem, obviously, is that people can still borrow tokens to stake in order to have more addresses to vote.

1. Use centralized IDs

The easiest way to solve this problem is probably to introduce some centralized control. For example, the GitCoin grants require users to login with GitHub, and GitHub provides an identity for voters.

Since HackerLink Grant is running on a smart contract, even if we require user login from frontend, savvy players can still sybil attack by directly interacting with the smart contract.

1. Use whitelist to register voters

Whitelisting is probably the most reliable method for on-chain voting. If voters get whitelisted before the voting period, sybil attack is automatically blocked.

The downside of this solution is that voters have to register themselves even before they see the projects. From operation's perspective, it will largely reduce the number of people who participate in donation and voting.

1. Progressive Tax

Another idea of depressing sybil attack is to use progressive tax (fee) system. More community donation will cause higher tax (fee). For example, if community donation is greater than 20 ETH for a single project, a 40% fee will apply, then cost of sybil attack will increase dramatically. The problem of this method is again — depressing votes when donation is above certain level. If we can estimate the max community donation of any project, we can design a reasonable progressive tax system. However, given that quadratic funding does not restrict entry of donors, it's pretty hard to estimate max donations

from community in the first place.

1. Grace Period

Because of the intensive competitions between project during the final days, we added a more restrictive grace period to future rounds. During the grace period, we will do two things:

1. Verify project identity
2. Conduct analysis of voting records within the round and detect sybil attack

The grace period allows us to identify scammers and projects conducting sybil attack. Once we find them, we can remove them from the registry, and cancel their rights to receive matching fund. In case of scammers, we have to return community donations back to users.

In practice, we added identity verification to only display verified projects on HackerLink's frontend.

1. Will Airdrop enhance the system or undermine the system?

Projects that promise airdrop to their donors are more successful in quadratic funding grants. We have seen this phenomena at DoraHacks' quadratic funding grant at ETH Hackathon Beijing, BSC Grant Round-1 on HackerLink, and GitCoin grants.

Promising airdrops will definitely help projects raise more community donation as long as they keep their promises. However, it actually conflicts the rule of quadratic funding. If airdrop is legitimized, then a quadratic funding grant will become a competition of airdrop campaigns, therefore violating the merit of community selecting projects they want to support "from heart".

There were some interesting discussions at the GitCoin GR9 Finale virtual event, people were actually talking about whether airdrop could bring a token model for platforms hosting quadratic funding grants.

The problem is — if airdropping is freely permitted, then quadratic funding becomes something else, maybe it's just more direct to build an airdrop donation platform.

Another proposal is to have projects choose between airdrop without taking matching fund and no airdrop with access to the matching fund. This might be a reasonable proposal, even though in practice it might be hard to prevent projects to secretly promise airdrop, because eventually it's hard to trace.

On the other hand, we can completely block airdrop by implementing MACI in quadratic funding, because all voting messages will be encrypted with an operator's public key and no further information will be revealed by the operator except a ZKP after round. But people might think about it — is this what we want?

Acknowledgements

We are inspired by Vitalik's paper "Quadratic Payments", and we appreciate all practices of quadratic voting and funding like GitCoin Grants, clr.fund, and so on.