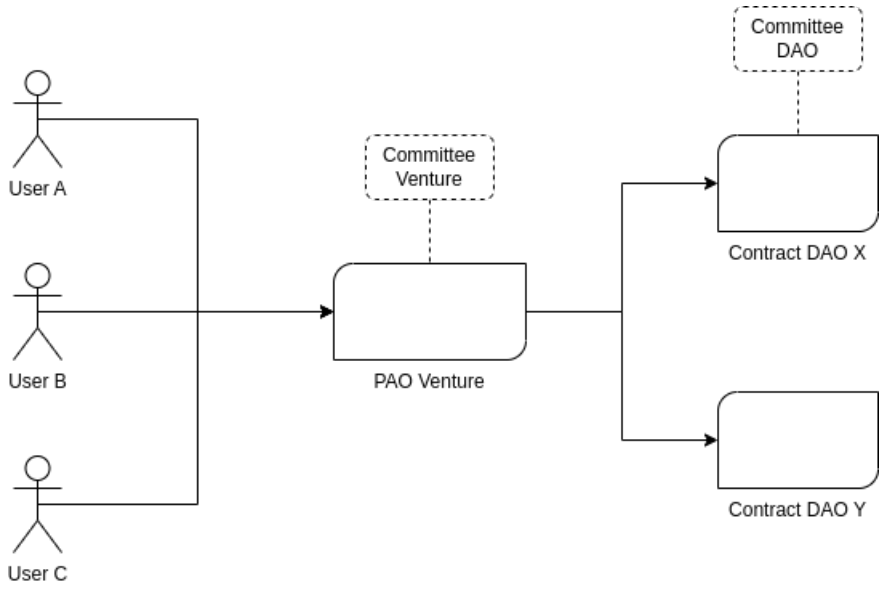
Overal Description

The PAO protocol is a solution to address the challenges of private funding and voting in decentralized autonomous organizations (DAOs). The protocol ensures that the investment details of the individual investors are kept confidential, while also allowing them to vote on proposals without revealing their stance. In other words, with this proposed protocol, no one knows which project an investor has invested in, and no one knows how an investor has voted on a proposal (for, against, or abstain). Thanks to the power of Zero Knowledge Protocol (ZKP), Distributed Key Generation (DKG), and ElGamal encryption, the protocol called THE PAO has been successfully developed to serve this purpose.



Goals of the funding stage:

- Balance privacy, when someone sends funds to a PAO Venture managed by the master contract, an observer cannot tell which DAO the funds is going to.
- User funds with a single submission of a transaction / message signature / proof.
- Avoid centralized dependencies; any off-chain computation should run open source code and allow multiple parties to participate, to optimize for censorship resistance and redundancy.

- The output must be the balance tally of each DAO.
- Impossible to tally before funding period ends.

Achievements during the voting stage:

- Privacy is preserved forever; e.g. designs where voter identity is exposed when counting votes, or at any point post vote casting are not acceptable.
- Double voting must not be possible.
- Should be impossible to tally a proposal's votes before the voting period ends; the reason is that if tallying is possible, whale votes become easily identifiable.
- Not every token holder should be required to vote, and if a voter registration round is required, not every registered voter should be required to show up to vote.
- The output must be the vote tally of each proposal: # for votes, # against votes and # abstain votes, with abstain being nice-to-have. The final decision whether a proposal passes or fails should be left to the DAO's smart contract
- Should not require an admin to trigger steps in the voting process.
- Avoid centralized dependencies; any off-chain computation should run open source code and allow multiple parties to participate, to optimize for censorship resistance and redundancy.

Technology

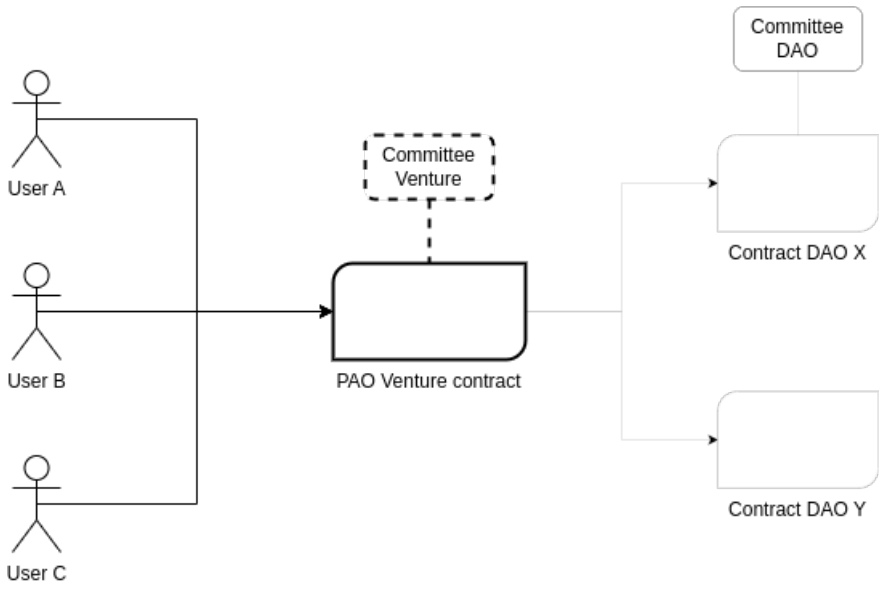
Throughout the protocol, we utilize Distributed Key Generation (DKG) to reduce dependence on a single address acting as the DAO manager. In other words, the committee now fulfills the role of a DAO manager, and the secret key of the DAO manager is managed by the members instead of a single individual. Therefore, assuming there are fewer than t people out of a total of n people who do not agree with each other, the problem of dependence on the DAO manager will be resolved.

Next, we utilize ElGamal encryption. This enables each voter to encrypt their ballot using the committee's public key. Only those holding the secret key of the entire committee can determine what choice was made. However, we employ a minor technique to ensure that this does not happen, as long as there are fewer than t individuals out of a total of n committee members who do not agree with each other (DKG phrase). Finally, we will tally the total number of votes for each option using a brute-force approach, as the total number of votes is finite.

Last but not least, Zero Knowledge plays an extremely important role in the protocol. With the ability to prove correct knowledge without disclosing information, we utilize Groth16 with circom language for describing circuits and verify proofs on solidity contracts. With the Snark protocol, we have achieved our goal of private funding and voting for investors while still maintaining security on the Ethereum platform.

Furthermore, we employ other significant techniques such as the Merkle Fixed Tree, double spending prevention, Poseidon hash, and the Baby Jub curve.

2.1 Venture Raise Fund



Hình 2.1: Funding Phase

Venture committee members need to perform a one-time Distributed Key Generation (DKG) to set up a public key Pk and n_c secret keys sk_i. The public key is public. Each secret key is owned by a committee member and is used for the final vote on-chain tally.

Once investors have the public key, they can use it for private funding with Threshold Homomorphic Encryption.

Once the funding phase ends, we require t out of n_c committee members to submit an on-chain transaciton.

On the venture contract, it also stores a Merkle fixed tree. The purpose is for investors to later prove that they have voting power during the voting round.

2.1.1 Distributed Key Generation (DKG)

Round 1

1. Each committee member i samples t random value $(a_{i,0}, \cdots, a_{i,t-1}) \leftarrow \mathbb{F}_r$ and define a degree $t-1$ polynomial $f_i(x) = \sum_{j=0}^{t-1} a_{i,j} * x^j$.
2. Every participant computes a public commitment:

   $$\boldsymbol{C_i} = \langle C_{i,0}, \cdots, C_{i,t-1} \rangle, \text{ where } C_{i,j} = a_{i,j} \cdot G, 0 \leq j \leq t-1$$

   and submits this public commitment to an onchain contract to verify that all points are valid.

Round 2

1. Each committee member keeps $(i, f_i(i))$ as a secret

2. Each committee member i posts every other ℓ-th committee member's public key encrypted secret share $ENC(f_i(\ell), C_{\ell,0})$ and a ZKP to prove:
   - $f_i(\ell) \cdot G = \sum_{k=0}^{t-1} \ell^k \cdot C_{i,k}$
   - Encryption is valid
3. Each committee member i calculates their secret key $sk_i = \sum_{\ell=1}^{n_c} f_\ell(i)$ and stores sk_i securely.
4. Each committee member computes the public key $Pk = \sum_{i=1}^{n_c} C_{i,0}$, which is shared with all user.

As a result, each committee member holds a secret key $sk_i \in \mathbb{F}_r$ and we have public key $Pk \in \mathbb{G}$ which voters will use when submitting their private vote on-chain.

$$\begin{pmatrix} \boldsymbol{C}_1 \\ \boldsymbol{C}_2 \\ \vdots \\ \boldsymbol{C}_{n_c} \end{pmatrix} = \begin{pmatrix} C_{1,0} & C_{1,1} & \cdots & C_{1,t-1} \\ C_{2,0} & C_{2,1} & \cdots & C_{2,t-1} \\ \vdots & \vdots & \ddots & \vdots \\ C_{n_c,0} & C_{n_c,1} & \cdots & C_{n_c,t-1} \end{pmatrix}$$

2.1.2 Private Funding By Investors

Investors use the shared public key Pk for private voting protected by ZKP and threshold hommomorphic Encryption.

During funding phase, investors only need to send a transaction to the governance contract. Assume that the investor want to fund an amount of v_i. The investor i first samples a vector of random number $\boldsymbol{r}_i \in \mathbb{F}_r^m$ (m is the number of DAOs available for funding.) and calls the governance contract signing a transaction containing $(\boldsymbol{R}_i, \boldsymbol{M}_i) \in (\mathbb{G}^m, \mathbb{G}^m)$ + a ZKP that:

1. public input: $Pk, v_i,$ commitment, $\boldsymbol{R}_i, \boldsymbol{M}_i, \boldsymbol{id}_{DAO}$
2. proving knowledge of $\boldsymbol{r}_i$ such that $\boldsymbol{R}_i = \boldsymbol{r}_i \cdot G$
3. Proving knowledge of vote $\boldsymbol{o}_i$ in 1-hot encoding. $M_{i,k} = (o_{i,k} * v_i) \cdot G + r_{i,k} \cdot Pk$ and k is index of the bit in the 1-hot encoded vote.
4. commitment = $H(null, id, v)$
5. $id = \langle \boldsymbol{id}_{DAO}, \boldsymbol{o}_i^T \rangle$

The contract receives the message and

1. Verifies the proofs
2. Accumulates $\boldsymbol{R} = \sum_{i=1}^{n_u} \boldsymbol{R}_i$ and $\boldsymbol{M} = \sum_{i=1}^{n_u} \boldsymbol{M}_i$
3. Inserts $H(null, id, v)$ and computes new root.

Denoting $\boldsymbol{r} = \sum_{i=1}^{n_u} \boldsymbol{r}_i$, $\boldsymbol{v} = \sum_{i=1}^{n_u} v_i * \boldsymbol{o}_i$, we have $\boldsymbol{R} = \boldsymbol{r} \cdot G \in \mathbb{G}^m$ and $\boldsymbol{M} = \boldsymbol{v} \cdot G + \boldsymbol{r} \cdot Pk \in \mathbb{G}^m$. During vote tally, committee member will reveal $\boldsymbol{v}$ as the result.

$$\begin{pmatrix} \boldsymbol{r}_1 \\ \vdots \\ \boldsymbol{r}_{n_u} \end{pmatrix} = \begin{pmatrix} r_{1,0} & \cdots & r_{1,m-1} \\ \vdots & \ddots & \vdots \\ r_{n_u,0} & \cdots & r_{n_u,m-1} \end{pmatrix}$$

$$\boldsymbol{o}_i = \begin{pmatrix} 0 & \cdots & 1 & \cdots & 0 \end{pmatrix}$$

2.1.3 Funding Tally

Funding allocation phase requires only t out of n_c committee members to participate, denote as $I \subset \{1, 2, \cdots, n_c\}$. Committee members interact with the governance contract and do not require secure p2p communication channels.

During funding allocation phase, each committee member queries $\boldsymbol{R}$ from the contract and sends a message $(i, \boldsymbol{D}_i)$ to the contract along with a ZKP proving that the decryption is valid and sk_i is generated correctly:

- $f_\ell(i) = dec(enc(f_\ell(i)), k_\ell(i)$
- $sk_i = \sum_{\ell=1}^{n_c} f_\ell(i)$
- $\boldsymbol{D}_i = sk_i \cdot \boldsymbol{R}$.

The contract

- Verifies the proof
- Receives and accumulates the first t messages from the committee, i.e. $\mid I \mid = t$ $\boldsymbol{D} = \sum_{i \in I} \lambda_i \cdot \boldsymbol{D}_i$ where $\lambda_i = \Pi_{j \in I, j \neq i} \frac{j}{j-i}$ is the Lagrange coefficient
- Reveals $\boldsymbol{v} \cdot G = \boldsymbol{M} - \boldsymbol{D}$
- Uses a lookup table to extract $\boldsymbol{v}$ from $\boldsymbol{v} \cdot G$

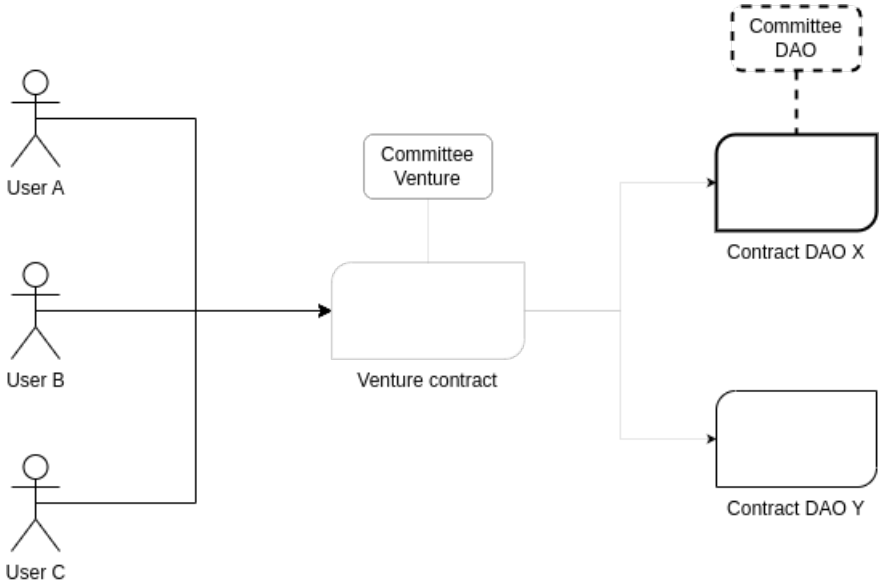
Note: Order matters! $i \in I$ must be the same index used during Distributed Key Generation.

Note: Suppose $\lambda_i(x) = \Pi_{j \in I, j \neq i} \frac{x-j}{i-j}$, we have $\lambda_i(0) = \lambda_i$.

Intuition: We use Lagrange interpolating to reconstruct the $\sum_{i=1}^{n_c} f_i(0) \cdot G = \sum_{i=1}^{n_c} a_{i,0} \cdot G = Pk$.

2.2 Voting

2.2.1 Description Voting Phrase



Hình 2.2: Voting Phrase

Likewise funding pharse, this phrase has two types of participants: n_c committee members who tally the ballots and n_u users who cast votes.

Committee members need to perform a one-time Distributed Key Generation (DKG) to set up public key Pk and n_c secret keys sk_i. The public key is public. Each secret key is owned by a committee member and used for the final vote on-chain tally.

Once user have the public key. they can use it for private voting with Threshold Homomorphic Encryption.

Once the voting ends, we require t out of n_c committee members to each submit an on-chain transacion.

The difference compared to the funding phase is that we will also conceal the voting power of the users using zero-knowledge proofs (ZKP).

2.2.2 Distributed Key Generation

Similar to DKG in the venture funding phase.

2.2.3 Private Voting By Users

User use the shared public key Pk for private voting protected by ZKP and threshold homomorphic Encryption.

At the beginning of each proposal, the DAO needs to store the root Merkle fixed tree in the funding phase.

During voting, users only need to send 1 message to the governance contract. Suppose each user has voting power v_i according to the snapshot. We have a $root$ storing the voting power of users. In particular, a user i samples a vector of random number $\boldsymbol{r}_i \in \mathbb{F}_r^3$ and calls the governance contract signing a transaction containing $(\boldsymbol{R}_i, \boldsymbol{M}_i) \in (\mathbb{G}^3, \mathbb{G}^3)$ + a ZKP that:

1. public input: $Pk, root, id_{DAO}, id_{proposal}, nullHash, \boldsymbol{R}_i, \boldsymbol{M}_i$
2. private input: $\boldsymbol{r}_i, v_i, null$ and path to root of commitment.
3. $H(null, id_{DAO}, v_i)$ is a node in Merkle tree that stores the voting power of user that satisfies the $root$.
4. $H(null, id_{proposal}) = nullHash$ for prevent double spending.
5. proving knowledge of $\boldsymbol{r}_i$ such that $\boldsymbol{R}_i = \boldsymbol{r}_i \cdot G$
6. Proving knowledge of vote $\boldsymbol{o}_i$ in 1-hot encoding where 100 means yes, 010 means no and 001 means abstain. $M_{i,k} = (o_{i,k} * v_i) \cdot G + r_{i,k} \cdot Pk$ and k is index of the bit in the 1-hot encoded vote.

The contract receives the message and

1. verifies the proofs
2. checks the account has not voted for the proposal before
3. accumulates $\boldsymbol{R} = \sum_{i=1}^{n_u} \boldsymbol{R}_i$ and $\boldsymbol{M} = \sum_{i=1}^{n_u} \boldsymbol{M}_i$

Denoting $\boldsymbol{r} = \sum_{i=1}^{n_u} \boldsymbol{r}_i$, $\boldsymbol{v} = \sum_{i=1}^{n_u} v_i * \boldsymbol{o}_i$, we have $\boldsymbol{R} = \boldsymbol{r} \cdot G \in \mathbb{G}^3$ and $\boldsymbol{M} = \boldsymbol{v} \cdot G + \boldsymbol{r} \cdot Pk \in \mathbb{G}^3$. During vote tally, committee member will reveal $\boldsymbol{v}$ as the result.

$$\begin{pmatrix} \boldsymbol{r}_1 \\ \vdots \\ \boldsymbol{r}_{n_u} \end{pmatrix} = \begin{pmatrix} r_{1,0} & r_{1,1} & r_{1,2} \\ \vdots & \ddots & \vdots \\ r_{n_u,0} & r_{n_u,1} & r_{n_u,2} \end{pmatrix}$$

2.2.4 Vote Tally

Similar to funding tally in the venture funding phase.

Approach

It is challenging to combine private funding and private voting while also being compatible with the PAO problem that we originally set out to solve, where the fundraising and voting processes are separate. Additionally, we prioritize convenience for users, where they can perform a single transaction for voting or investing.

We will begin with the private voting process. We approach user convenience through ElGamal encryption and brute-force synthesis. The idea is that we will need a DAO manager. Users will encrypt their choices using the public key of the DAO manager. Then, the DAO manager will tally the total number of votes from all users. Two issues arise: firstly, the DAO manager owns the private key and can decrypt the choices of each user. Secondly, the DAO manager needs to be "online" to ensure the voting process runs smoothly. There is too much "power" here. However, the advantage of this protocol is that it creates convenience for users, as mentioned earlier, eliminating the need for two transactions if a commitment scheme were used.

The next step is to reduce dependency on the DAO manager. This suggests the use of a distributed key generation (DKG) algorithm. The DAO manager will now be a committee, and the security level of the protocol will depend on t people rather than one individual. Moreover, we can use a technique to ensure that even the committee members do not know the shared secret key, making it impossible to decrypt the user's choices.

The fundraising process has become much easier if we apply the aforementioned private voting process to this phase. For each fundraising round, there will be a list of invested DAOs. We will consider investment choices as a type of vote. Then, we will tally the balances of each DAO. The security of investment information will depend on the Venture committee.

A new issue arises, where we need to prove the voting power of investors in the voting phase. Thanks to Zero knowledge and Merkle trees, we were able to solve this problem. Moreover, Zero knowledge also helps ensure that committee members cannot cheat in the contribute DKG and tally processes.