FROSTDAO: Collective Ownership of wealth using FROST

Rahim Klabér

April 27, 2023

1 Introduction

1.1 TODO REWRITE

The introduction of Bitcoin allowed anyone to send and receive money anonymously and without government oversight[CITE]. Subsequent Blockchains introduced the concept of ”smart-contract”, code that is stored on a Blockchain and runs depending on certain conditions. Smart-contracts led to the concept of Decentralized Autonomous Organizations (DAOs). These are leaderless organizations where the members can collectively determine what to do. These DAOs use Blockchain smart-contracts to allow for the collective ownership of wealth.

The behavior of certain smart-contracts can be emulated with cryptography instead of code. Most importantly, a smart-contract for collective decision-making can be replaced with threshold cryptography[CITE]. In this case the members of a DAO would jointly control a Blockchain account that serves the account of the organization.

In this paper, we introduce a DAO framework based on cryptography, instead of Blockchain smart-contracts. We substitute smart contracts with threshold signatures to create a DAO framework that is Blockchain agnostic and does not require the underlying Blockchain to support smart-contracts.

2 Problem

The goal is to create a DAO framework that can scale to hundreds or even thousands of members for Blockchains where DAOs cannot be created using smart-contracts. In particular, Bitcoin. Our framework should work in a fully decentralized and peer-2-peer setting where the only participants are smartphones and there are no servers. This is especially challenging given the unreliability of smartphone networking.

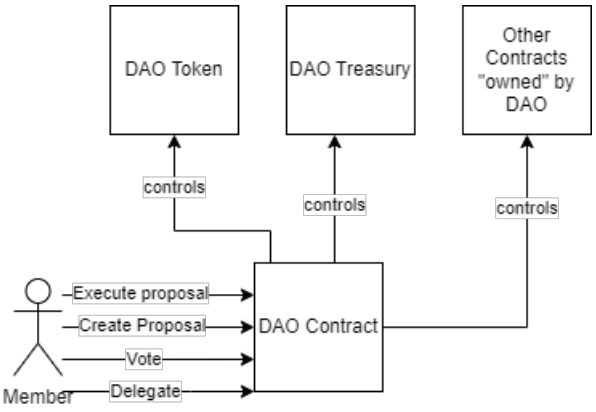

Figure 1: Traditional DAO architecture.

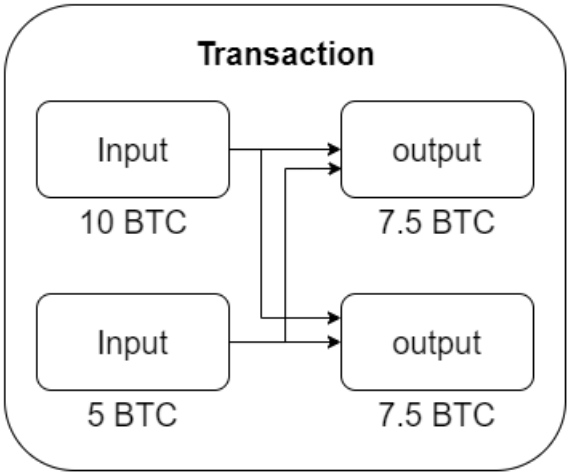

Figure 2: Simplified Bitcoin transaction. Shows inputs with Bitcoin values being consumed to create new inputs.

3 Background

3.1 DAO

A Decentralized Autonomous Organization (DAO) is a collectively-owned, blockchain-governed organization[cite]. DAOs are leaderless organizations where decisions must be made through voting on proposals that are created by its members. Traditionally, DAO consists of a number of components. First, the DAO smart contract, which handles proposals and voting. Second, a DAO token represents a state in the DAO. Third, the DAO treasury contract holds the assets of the DAO. Lastly, various contracts or applications are governed by the DAO.

The DAO smart contract is responsible for registering proposals and allowing members to vote. Proposals include instructions that are executed by the smart contract if the proposal receives enough votes. Often, creating proposals requires a certain amount of voting power.

The DAO token is used to show membership in the DAO and represent voting power within the DAO. The DAO controls the DAO token smart contract, which can be used to issue more tokens or remove existing tokens from circulation.

The DAO treasury holds the assets of the DAO. This can include DAO tokens, but also other tokens on the Blockchain. The treasury is used to pay members who contribute to the DAO and is used to pay for other expenses.

A DAO may have control over other smart contracts. This enables the DAO members to control and make changes to those contracts when they deem it necessary. For example, AAVE, a blockchain lending platform, is controlled by a DAO that can adjust the risk parameters of the lending smart contracts.

DAOs often have a small number of members who are active and a large number who do not actively participate. This means that a large portion of power within the DAO is inactive, which can lead to not enough participation and proposals failing. A solution to this problem is delegation, members can delegate their voting power to other members who they think are trustworthy. Effectively delegating temporarily gives someone your voting power.

3.2 Bitcoin

Compared to newer Blockchains like Ethereum, Bitcoin uses an interesting transaction model. Bitcoin transactions consist of a number of inputs and outputs. Each input is an output of another transaction.

Outputs have a Bitcoin value attached to them, which can be used to pay other Bitcoin accounts. Outputs can be created by ”spending” an input which renders that input unusable anymore. Each input has a small program that must be satisfied to spend the input. This allows for

some interesting constructions. Figure 2 shows a simplified view of a Bitcoin transaction

As mentioned previously, Bitcoin inputs include a program that must be satisfied to spend the input. When attempting to spend an input, the user provides inputs to satisfy the input program. When doing a normal payment the input to the program is a signature proving that the user owns the address that is allowed to spend the input.

The spending rules can allow for more than just sending payments. One example related to the problem statement of this paper is requiring an input to be approved by multiple users to be spent. This would allow a basic DAO to exist on Bitcoin where the execution of proposals creates outputs with arbitrary rules. However, this solution only works for extremely small DAOs as Bitcoin transactions shave a size limit and Bitcoin fees are dependent on the size of a transaction. Therefore, even for small DAOs this solution is not optimal as the fees would be high.

3.3 Threshold Signatures

Threshold signatures Schemes is a method for creating signatures where multiple users are required to collaborate to create a signature. A Threshold signature scheme has 2 parameters. The number of participants n and the threshold t where $t \leq n$. Only t participants are required to create a valid signature.

To create a Threshold signature scheme a private key must be split up and each member must have a share of the private key. This can be done by relying on a trusted party that would generate a private key and split it up for each of the members. This does not work for peer-2-peer systems as there is no trusted party. Instead, a distributed key generation protocol is used, which generates the key shares in a distributed fashion such that no party learns the actual private key. Compared to using a trusted dealer, distributed key generation is much costlier and takes multiple network rounds.

To create a signature t participants need to sign the message. The t signed messages can then be combined to create the final signature.

3.3.1 Flexible Round-Optimized Schnorr Threshold Signatures (FROST)

FROST is a Threshold signature scheme that is able to create signatures that are compatible with Bitcoin. FROST consists of two protocols, one for key generation and one for signing.

The key generations protocol consists of 2 rounds, which both require every participant to broadcast a message to every other participant. The first round is particularly problematic as the size of the message depends on the number of participants.

The signing protocol is much lighter than key generation. It also consists of two rounds. However, the first round does not require knowledge of the unsigned data. Therefore, the output of the first round can be batched, resulting in a much quicker signing protocol. In the second round of signing, each participant signs the data with their own key share. The signatures are then combined to create the final signature.

3.4 IPV8

IPV8 is a peer-2-peer networking library. It allows for the creation of applications that do not require a central server. One example, is MusicDAO[cite], a Spotify alternative aiming to give a more significant share of the revenue to music artists.

IPV8 works by keeping a list of peers that are periodically checked for liveness. Peers are introduced to new peers which they can add to their list to keep track of. IPv8 is particularly useful because it supports hole-punch and therefore works behind WIFI, where peer-2-peer applications normally wouldn't work.

On top of the networking layer, IPV8 has the concept of Communities. These can be seen as protocols that live on top of IPV8. Members of a Community can communicate with each other using Community specific messages which are handled in Community specific ways. For example, A Community named Torrent Community could implement torrent functionality.

3.5 System Design

4 Implementation

4.1 Bitcoin specifics

Class / Package	Line coverage	Lines of code
FrostManager	93%	404
SchnorrAgent	94%	106
FrostCommunity	65%	141
FrostViewModel	0%	156
ui	0%	980

Table 1: Code coverage of the FROSTDAO application.

5 Evaluation

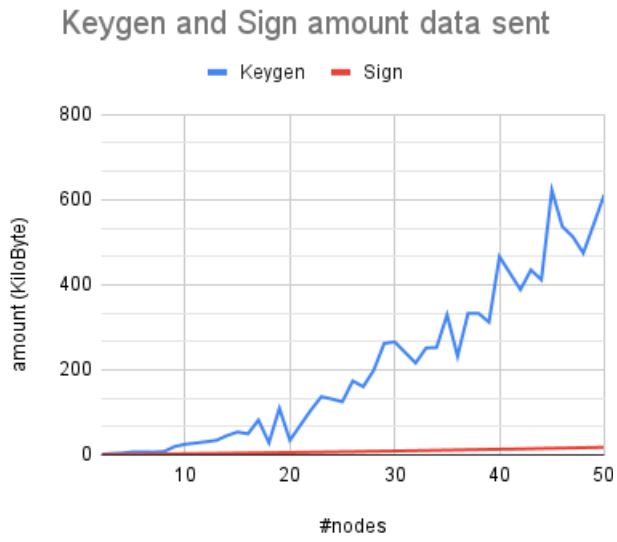

Figure 3: Amount of data in Kilobytes sent during Key generation and Signing

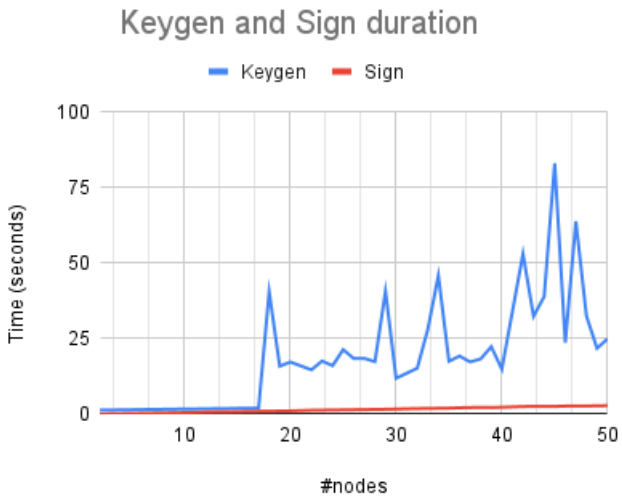

Figure 4: Duration of Key generation and Signing.

6 Conclusion