SingletonDAO: Multi-party Ownership System For A Single Asset

Christopher Hughes

February 12, 2023

Contents

1 Introduction

SingletonDAO is open-source project that makes the management of a single asset owned by multiple parties simple. It consists of a set of Ethereum smart contracts, a web interface, a mobile wallet, and the corresponding back-end that is based largely on the Gnosis Safe multi-sig wallet and mobile-client. SingletonDAO smart contracts and corresponding interfaces allow a Gnosis Safe's signing functions to be assigned to holders of an NFT. This is achieved by augmenting the signing relays in a way that checks for the presence of an NFT in a signers account, if present, allows the account holder to sign transactions. Soon, EIP-4337 also will bring the ability for Ethereum accounts to be abstracted in interesting ways to support new signing paradigms. Either of these use-cases are supported in the SingletonDAO mobile wallet, it's a white-label build of Gnosis Safe for iOS and it includes a set of tools that allows for the management of a single asset held in treasury. Proposals for the sale of the asset in treasury will be periodically presented to the user, and the tools for funding the treasury will allow stakeholders to send value to Safe using a crowdfunding like interface.

The Gnosis team has made this possible by implementing the functional primitives in a project they call ClubCard, this is part of the technology we are employing in this mono-asset DAO tool-set. SingletonDAO also includes the crowdfunding portal code, as well as process for presenting various bulletins for disclosure or voting notification services.

Simply put, SingletonDAO allows for the group ownership of an asset, and the life-cycle of the asset's management long term. This includes governance voting on subjects like liquidation or acquiring additional assets in treasury, sale between participants, participant capital calls and other group ownership decisions are made simple. All of this is achieved my leveraging the module system of Gnosis Safe, coupling it with the issuance of NFT software to manage access.

SingletonDAO will have a commercial implementation, this software is capable of being packaged in a way to help others with digital asset fractionalisation. This is the use-case it's being built for but is a worthy piece of software for simple asset DAOs.

2 Ethereum Account Abstraction

Ethereum account abstraction is a design pattern that allows developers to create more flexible and complex smart contracts by separating the ownership of assets (e.g., Ether) from the control of those assets. With account abstraction, a contract can be set up as the owner of an account, allowing it to receive and send funds and interact with other contracts on the network.

In practical terms, this means that developers can create more sophisticated and modular contracts that can interact with multiple assets and contracts without having to worry about the security of the underlying account. Instead of having to manage the account themselves, developers can delegate ownership to their contracts, which can then perform various actions on their behalf.

Account abstraction is implemented through the use of contract wallets, which are smart contracts that act as the interface between the account and the contract. These wallets can have their own logic and functionality, allowing them to perform various tasks and interactions with the blockchain.

Overall, account abstraction is a powerful tool that allows developers to create more complex and secure contracts on the Ethereum network, while also providing more flexibility and control over the assets involved.

To achieve this, a new standard EIP-4337 the Ethereum Improvement Proposal that aims to enhance the security of Ethereum transactions by introducing standardized multi-signature capabilities. Multi-signature transactions require more than one person to authorize a transaction before it can be executed, providing an extra layer of security that can help prevent fraud, theft, or other malicious activities. EIP-4337 proposes a standardized implementation for multi-signature transactions that can be used by any application or contract on the Ethereum network.

The core idea behind EIP-4337 is to introduce a new smart contract that acts as a multi-signature wallet, with ownership of the wallet tied to the smart contract rather than any specific individual or entity. This means that the wallet can be controlled by multiple signers who collectively authorize transactions, rather than relying on a single private key. The smart contract also includes additional features such as time-based or event-based rules that can be used to automate certain types of transactions.

One of the main benefits of EIP-4337 is that it provides a standardized implementation for multi-signature transactions, making it easier for developers to incorporate this functionality into their applications or contracts. The proposal includes a detailed technical specification that outlines how the smart contract should be designed and how the multi-signature functionality should be implemented. This can help ensure that all applications and contracts on the network use the same security standards, making it easier to manage and secure the network as a whole.

Finally, EIP-4337 has the potential to improve the security and reliability of the Ethereum network by reducing the risk of fraud, theft, or other malicious activities. Multi-signature wallets provide an extra layer of security that can help prevent unauthorized transactions or protect against other types of attacks. By providing a standardized implementation for this functionality, EIP-4337 can help ensure that all applications and contracts on the network use the same high standards of security, making the Ethereum network a more secure and reliable platform for decentralized applications and services.

3 Differences between Gnosis Safe and EIP-4337

EIP-4337 and Gnosis Safe are both solutions that aim to improve the security of Ethereum transactions by introducing multi-signature capabilities.

Overall, both EIP-4337 and Gnosis Safe offer valuable solutions for improving the security of Ethereum transactions, and the choice between the two will depend on the specific needs and priorities of the user.

However, there are some key differences between the two:

1. Implementation: EIP-4337 is a proposed Ethereum Improvement Proposal that has not yet been fully implemented in the network. In contrast, Gnosis Safe is a popular and widely-used multi-signature wallet solution that has been live on the network for several years.

2. Customizability: Gnosis Safe is highly customizable, allowing users to specify the number of signers required for each transaction, set up different access levels for different users, and create custom rules for each transaction. EIP-4337, on the other hand, proposes a standard implementation for multi-signature transactions that does not offer the same level of customization.

3. Interoperability: EIP-4337 is designed to be compatible with other Ethereum-based applications and contracts, making it easier to integrate with other services on the network. Gnosis Safe also supports integration with other applications, but may require more customization and development work to achieve the desired level of interoperability.

4. Ownership: With EIP-4337, the ownership of the wallet is tied to a specific smart contract, whereas Gnosis Safe wallets are owned by specific individuals or entities. This means that with EIP-4337, ownership can be transferred more easily if necessary, but also requires a greater level of technical expertise to set up and manage.

EIP-4337 is in it's infancy with respect to software support, there are some wallets that are being built to support the standard and the Gnosis Safe team is actively working with the standards designers to support the new paradigm of ownership. Once this standard is better adopted by users, the expectation is that wallets like Metamask can be modified to allow the advanced security features to roll out.

4 What is ClubCard?

ClubCard introduces three new signing paradigms to the Gnosis Safe Programmable ownership, it's a newly developed module of granular governance that can greatly improve the security of a Gnosis Safe by changing the crude N of M signing schemes to allow alternate signing behavior. This is achieved by using another digital asset to verify the ability to sign. The mere presence of either the correct quantity or the correct asset in signers wallet would allow the signer to effect a transaction of a Gnosis Safe without being one of the explicit owners of the safe.

Transferrable ownership is the simplest form of NFT governance, this allows a holder of an ERC-721 based NFT to sign transactions against the Gnosis Safe.

- ClubCardERC721.sol: allow the holder of a designated ERC-721 NFT to sign transactions

Threshold ownership is an ownership scheme where a quantified level of assets must be in the signers wallet to sign a transaction. This allows for granular management of how Safe transactions work. Threshold Signature Schemes resemble voting governance where a score must be above a threshold to enact a gonvernance action. This can allow for the expression of ownership on a pro-rata basis of contribution. This can be done with either an NFT based on the 1155 standard, or any available ERC-20 based token on Ethereum.

- ClubCardERC20.sol: allow holders of a minimum balance of ERC-20 tokens to sign transactions
- ClubCardERC1155.sol: allow holders of a minimum balance of ERC-1155 tokens to sign transactions

  Programmable ownership

- ClubCardBase.sol: implement custom ownership terms by extending this abstract contract

EIP-1271 signatures ClubCardBase implements the EIP-1271 interface. It validates that a given ECDSA signature is from the expected account where the expected account is derived using the isCardHolder that inheriting contracts must implement.

Additionally, it supports validation of EIP-1271 contract signatures, which are expected to be given in the following format based on ECDSA r, s, v components with v = 0 as the recovery identifier: