Dagon: Governance Abstraction

Ross Campbell
ross@kali.gg

Abstract

Dagon is a DAO singleton smart contract architecture, simplifying core governance primitives for digital asset management. Chiefly it acts as an abstraction layer to allow any account to become a group account without requiring new interfaces or deployments. The alpha implementation[1] supports weighted voting with arbitrary tokens, as well as a native token creation method. Accounts can initially opt in and out of Dagon governance by calling its installation function or deleting their storage. Dagon can also be given programmatic custody of accounts that support ERC-1271[2] contract signatures and ERC-173[3]. Dagon will serve as a first and demonstrative example of how governance abstraction can work with AA[4] to streamline group signature verification and lead to an improved blockchain UX.

Background

Owning something on the internet has been made vastly more accessible through blockchain technology. Users do not need to request permission to hold a digital asset if they have access to basic cryptographic tools like private keys. *Being-your-own bank* has some obvious drawbacks, however, such as losing your funds permanently in cases of mistakes.

These risks are not acceptable for most property regimes. Thankfully, smart contracts allow users to program how they interact with blockchains, such as

[1] *Dagon.* https://github.com/Moloch-Mystics/dagon

[2] Giordano et al., *ERC-1271: Standard Signature Validation Method for Contracts.* Ethereum Improvement Proposal, 2018. https://eips.ethereum.org/EIPS/eip-1271

[3] Mudge et al., *ERC-173: Contract Ownership Standard.* Ethereum Improvement Proposal, 2018. https://eips.ethereum.org/EIPS/eip-173

[4] Account Abstraction or *AA* is a popular term used to describe the technical and social steps to make blockchains and smart contracts mostly abstract or invisible to end users.

splitting custody risk by granting other accounts the right to approve sending tokens or other digital transactions from a shared address.

So far, smart contract developers have provided several templates for such multi-owner blockchain accounts. For example, a multi-signature contract (multisig) typically allows users to program and set up joint custodians with an approval threshold, such as a "2-out-of-3" configuration for group sign-off. Similarly, accounts with diverse or transient ownership may opt to allow users that hold balances of certain so-called "governance tokens" to vote onchain to make collective decisions, such as adjusting the issuance rate of, or making developer grants in, their own token.

Such tools for group accounts have allowed more users to enjoy the benefits of blockchain technology and digital assets with far less risk. Conveniently, it is also no longer a question of technical expertise to add governance rights to a blockchain account. Multisig and DAO frameworks have thus at this point established so far the requisite building blocks to make governance even more accessible through a singleton architecture.

Singleton designs have been increasingly more common because they offer very acceptable efficiency gains compared to their risk of funnelling interactions to a single instance of code. Specifically, they reduce the amount of duplicative bytecode needed to service shared business logic, as seen by the NFT marketplace OpenSea shifting to a singleton (*Seaport*)[5] to broadly cater to users outside their platform, as well as Uniswap[6], the `ERC-4337`[7] EntryPoint [8], and others.

Notably, a singleton can be built upon much more easily than a more complex series of contracts, and in a development context that is particularly unforgiving given the financial risks, this is an important feature to consider. Further, web applications have a much simpler time tracking contract state changes and recognizing user balances in a singleton by avoiding the need to make multiple calls to the blockchain.

For these reasons Dagon is proposed and deterministically deployed with `create2` in alpha (`DV0`) as a singleton to a canonical instance, recorded at *`0x0000000000001D4B1320bB3c47380a3D1C3A1A0C`*.

---

[5] *Seaport.* https://github.com/ProjectOpenSea/seaport

[6] *Uniswap.* https://github.com/Uniswap/v4-core

[7] Buterin et al., *ERC-4337: Account Abstraction Using Alt Mempool.* Ethereum Improvement Proposal, 2021. https://eips.ethereum.org/EIPS/eip-4337

[8] *EntryPoint.* https://github.com/eth-infinitism/account-abstraction

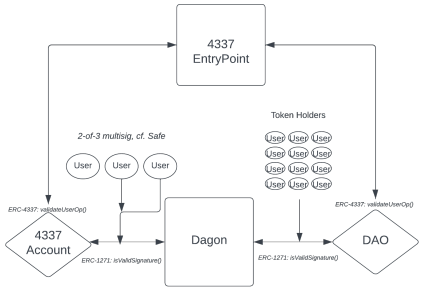

Figure 1: Dagon Singleton

Users

Accounts that integrate with the Dagon singleton may be externally-owned (EOA) or another smart contract, like a DAO. Execution of proposals will require that the user is a smart contract account, until the time that EVM proposals like `EIP-3074`[9] may be adopted to allow delegation of EOA contract calls.

Installation

Dagon exposes a public installation method (`install()`) that receives an `Ownership` struct of owners (`address`) and their shares (`uint96`), a `Settings` struct including a `token` (`address`), voting `threshold` (`uint88`) and `Standard` (`enum`).

[9] Wilson et al., *EIP-3074: AUTH and AUTHCALL opcode.* Ethereum Improvement Proposal, 2020. https://eips.ethereum.org/EIPS/eip-3074

For Existing Tokens

In *DV0*, the `Standard` enumerates the following token interfaces for account settings: `ERC-20`[10], `ERC-721`[11], `ERC-1155`[12], and `ERC-6909`[13]

For New Tokens

Users can provide a struct of `Metadata` that adheres to the `ERC-6909` metadata extension reference if they want to store additional context and use Dagon for tokenizing their account ownership.

Doing Offchain Vote

Dagon incorporates `ERC-1271`. It performs a signature collection loop,[14] verifying signer balances according to the calling account's `Settings`, and checks if the `threshold` weight is met. If the `threshold` is met, the call to `isValidSignature()` returns the expected `ERC-1271` magic value (`0x1626ba7e`).[15]

Doing Onchain Vote

For cases where required signatures exceed simple loops,[16] Dagon allows users to call the `vote()` function. Signatures are posted onchain, and the run-

[10] Fabian Vogelsteller, Vitalik Buterin. *ERC-20: Token Standard.* Ethereum Improvement Proposal, 2015. https://eips.ethereum.org/EIPS/eip-20

[11] Entriken et al., *ERC-721: Non-Fungible Token Standard.* Ethereum Improvement Proposal, 2018. https://eips.ethereum.org/EIPS/eip-721

[12] Radomski et al., *ERC-1155: Multi Token Standard.* Ethereum Improvement Proposal, 2018. https://eips.ethereum.org/EIPS/eip-1155

[13] JT Riley et al., *ERC-6909: Minimal Multi-Token Interface.* Ethereum Improvement Proposal, 2023. https://eips.ethereum.org/EIPS/eip-6909

[14] In the primary validation flow, if no signature is provided to a call to `isValidSignature()`, an onchain vote will be assumed, and the function will skip to check the stored tally against the `hash` argument.

[15] With this view return flow, Dagon is likely best suited for smart contract accounts that support `ERC-1271` contract signatures. For example, a smart account that wants to validate an `ERC-4337` `userOp` would make Dagon its `ERC-173`-compliant owner, and install governance settings for membership tokens and voting weight threshold.

[16] There are also composability benefits to posting votes onchain, where peripheral smart contracts might trigger new updates based on voting actions, or distribute rewards to incentivize more governance participation.

ning tally is stored in the `votingTally` mapping.[17]

Signatures Supported

Supported signature methods are ECDSA (elliptic curve) and `ERC-1271` (contract). New cryptographic curves or verification rules should be implemented through `ERC-1271`-compatible updates.[18]

Execution Guard

A contract address for authorization requests is included in the Dagon `Metadata` struct, as an account's `authority`. This administrative account validates transfers of tokens created on Dagon and guards against invalid calls made by group accounts, following the design of *DSAuth*[19].

RWA Use Cases

Dagon can be extended for tokenizing assets with more fluid or customizable rules. Membership tokens in a DAO LLC, for example, can be given governance rights over a contract account by installing these settings into Dagon. For security tokenization, an agent or another DAO can manage and authorize transfers using the Dagon `IAuth` interface to establish a compliance plan onchain.[20]

[17] These votes may be posted in a batch, as well, to save on network fees, but must follow the signature formatting conventions (see below, *Signatures Supported*).

[18] One current limitation in `DV0` is the preference towards packing group signatures into blocks of 85 bytes for its validation loops. This follows the standard Ethereum convention of building ECDSA signatures into 65 bytes (`v, r, s`), where in this case, the signature owner address is also placed into the first 20 bytes and extracted in each loop run. The benefits of allowing compact signatures (64 bytes) via `ERC-2098` to reduce calldata costs is currently being explored for `DV1`. The drawback of this approach is simply lack of adoption and rigorous developer tooling.

[19] *DSAuth.* https://github.com/dapphub/ds-auth

[20] Specifically, and for example, an agent or another DAO or onchain service company can be given the right to manage and authorize transfers using the Dagon 'IAuth' interface. This configuration, and the ability to more granularly control transfers beyond just simple "pause and unpause" will allow simplifications around issuing equity that can now be programmed to respect regulations, holistically. Notably, and unlike other proposals around security tokenization, the Dagon system doesn't just propose a method for self-regulated tokens using existing token interfaces, but also self-regulation of digital organizations themselves: users can actually house their shareholder governance and onchain ownership into the Dagon singleton.

Philosophy

The Dagon singleton should be simple, optimized, and a public good in the larger project of meta-governance. It is free-to-use, licensed under AGPL, and aims to encourage cryptographic proof of ownership and decision-making for everyone on the internet.