Imperial College London

IMPERIAL COLLEGE LONDON

DEPARTMENT OF COMPUTING

Decentralized Autonomous Organizations for Funding and Monitoring Infrastructure in Developing Countries

Author:
Gus Levinson

Supervisor:
Will Knottenbelt

Submitted in partial fulfillment of the requirements for the MSc degree in Computing of Imperial College London

August 2022

Abstract

Traditional fundraising routes for financing infrastructure projects in developing countries are typically slow and inefficient. Capital raised must pass through multiple unnecessary layers of bureaucracy, whether that be financial institutions or government officials, before reaching the intended beneficiary. This pathway makes it very difficult to make fast impactful change and enables corruption, resulting in reduced investment to the areas that need infrastructure the most.

In this thesis, we propose an alternative financing pathway. We present a technical architecture that enables the proposal, fundraising and monitoring of infrastructure projects on the Ethereum blockchain. The architecture is intended to be implemented by decentralized autonomous organizations with a focus in financing infrastructure development in the Global South.

Our eco-system of smart contracts allows capital to be raised in a cryptocurrency chosen by the entity that proposes the project. Consequently, capital raised bypasses third party institutions and reaches the desired entities much faster. Investors are granted access to data associated with projects that they have invested in. This data enables investors to monitor a successfully-funded project's progress and, upon completion, maintenance.

Our architecture has been well received by experts in smart contracts, in applying blockchain and in building infrastructure in underdeveloped nations respectively. We also demonstrate that our smart contract gas fees are low considering each transaction's particular use case.

Acknowledgments

I sincerely thank my supervisor, Professor Will Knottenbelt, for his unmatched enthusiasm and support throughout the duration of this thesis. Will's ability to seemingly fit me in anytime (and anywhere) has been incredibly kind and reassuring.

Thank you to the rest of the Spring DAO team for providing such a stimulating work environment. I would especially like to thank Rob Hygate and the eWater team (namely Seibo, Jainaba and Alieau) for allowing me to visit their projects in The Gambia and being so hospitable in the process. I express my gratitude to Imperial for financing a substantial part of the trip.

Rob, Catherine Mulligan and Jamie Anson were extremely generous with their time reviewing and discussing this project.

Finally, thank you to my friends and family for their unwavering support, without which this thesis would not have been possible.

Contents

Chapter 1

Introduction

1.1 Motivation

There are significant barriers to fundraising infrastructure projects in developing countries using traditional financing pathways. Such routes are typically slow, inefficient and over-complicated due to unnecessary layers of bureaucracy. Even projects proposed by the World Bank can take up to 4 years for the tender winner to receive funding and be able to begin construction [1]. Once sufficient capital has been raised, it must often pass through multiple unnecessary financial institutions and government officials before reaching the intended destination. This makes it incredibly difficult to make fast, impactful change and consequently limits investment.

Not only is the tardiness of this process problematic, but the lack of accountability and general security throughout it enables corruption. It is estimated that over a trillion dollars that is intended for poverty-stricken nations is siphoned off each year by corrupt government officials [2]. Corruption causes a fundamental lack of trust for all parties involved, disrupting charitable causes and, as one would expect, having a negative impact on investment [3].

Figure 1.1: Image taken on the author's research trip to The Gambia. The aim of the trip was to increase understanding around the context of implementing infrastructure projects in developing countries and how such project's stakeholders operate. Further details in chapter 3.

We propose an architecture for funding infrastructure projects in developing countries. The architecture aims to harness many of the benefits of blockchain, cryptocurrency and smart contract technology whilst being seamlessly integratable with the practices of building infrastructure in less technologically advanced locations. Blockchain technology offers the Global South unprecedented security and transparency. Blockchain provides public, immutable records that ensure accountability which instils trust.

Within the architecture is a fundraising system that raises capital in specific cryptocurrencies (mainly stablecoins). As a result, there will be no involvement from unnecessary third-party financial institutions and much more limited involvement, if any, from governments. Payments will reach the desired entities much faster (almost immediately) and will be significantly more traceable due to the transparent nature of blockchain. The system is functional for fundraising infrastructure projects that expect a financial return, such as funding solar panel farms, and for those that are purely charitable like financing the construction of a water system.

Figure 1.2: 'Jalambang lower basic and nursery school', suburbs of Banjul, The Gambia. The headmistress is desperately trying to raise capital to build a new teaching block and make enough space for all the students. The proposed architecture could be ideal for making such a raise. (The author, 2022)

The proposed architecture expects to be implemented by decentralized autonomous organizations (DAOs) focused on financing infrastructure projects in developing countries. An example of such an organization is the "Spring DAO", which the author has been assisting to create throughout the duration of this report. The Spring DAO, currently under development, aims to provide a decentralized fundraising platform for proposing, funding and monitoring infrastructure projects in developing countries.

DAOs that implement our system will require that data be fed back from a successfully funded project to ensure that it is being implemented correctly and maintained. In the example of a funded water system, as construction begins this data might prove a borehole has been drilled by showing the presence of running water at the desired location. By the end of construction, the data may be the live water consumption from the water system. Not only is this data necessary as evidence, but it is also valuable to users involved in the project.

We therefore propose an authentication mechanism within our architecture that allows investors to view data associated with the projects that they have contributed towards. This allows users to monitor their projects and therefore retain an active interest. It also means they can see first-hand the tangible benefits of their contribution. Our mechanism will ensure that only those who have invested sufficiently in a project have access to its data. For the scope of this thesis, we can assume the existence of a mechanism that ensures the collection, integrity and availability of such data.

1.2 Objectives & Aims

Our primary research question is:

"Do Decentralized Autonomous Organisations provide a route for funding and monitoring of infrastructure projects in developing countries".

To that end, I have laid out the following objectives:

1. To represent infrastructure projects using smart contracts/tokens.
2. To build a mechanism for on-chain fundraises using smart contracts.
3. To create an authentication mechanism that allows users to access data associated with projects that they have invested in.

1.3 Contributions

We highlight the main contributions of this thesis below.

1. **'ProjectMarketplace': Smart Contract for creating infrastructure projects and their associated fundraise**

   We have designed and implemented an ERC-721 compliant smart contract in Solidity called 'ProjectMarketplace' for project proposers to interact with. The contract's main function is called 'createProject'. The entity that calls the function specifies the details of the project in the function's parameters, such as the name of the project, the amount it is aiming to raise, the cryptocurrency that funds are raised in and the 'data access threshold' (minimum amount of

investment required to view a project's data). When 'createProject' is called, a smart contract that represents the project's fundraise is deployed and a NFT minted which is owned by the fundraising contract. The NFT represents the project and its ownership. 'createProject' uses 2,440,000 units of gas which is estimated to cost approximately 0.0853 ETH according to the average daily gas prices since June 2022.

2. **'Fundraise': smart contract for raising a project's required funds on-chain**

   We have developed an ERC-20 smart contract in Solidity called 'Fundraising' which is deployed when a new project is created. The contract is interacted with by investors and holds the capital raised until the project's target amount is reached, at which point the funds are immediately sent to the proposing entity. The contract's functions 'invest' and 'withdrawInvestment' allow users to add and withdraw their capital from a project's fundraise. As shown in section chapter 5, functions gas fees are estimated to be 0.00476 ETH and 0.00171 ETH respectively which prices out the very small investors (below $\approx$ \$50) but is insignificant to the remainder.

   In return for investment, users receive the Fundraising contract's ERC-20 tokens in a 1:1 ratio in the value to their contribution. If a user withdraws funds, a corresponding amount of these token are removed from their possession and burned. The Fundraising contract tokens are referred to as "re-fungible tokens" (RFTs) as they represent a share of the project NFT's ownership and, thus, the project's ownership itself. If a user invests the full amount a project is aiming to raise, they can call the function 'redeemNFT'. This results in the user receiving the project NFT itself and burns their corresponding RFTs. The entity that proposed the project can stop the fundraise temporarily with the function 'Pause' or stop it permanently with 'kill'. 'redeemNFT', 'pause' and 'kill's transaction fees are all negligible considering their use cases.

3. **An authentication mechanism that uses project tokens as access tokens**

   We have built an authentication mechanism in JavaScript by using the Express.js framework and Ethers library. As investment into a project is represented by ownership of the project's RFTs or NFT, the authentication mechanism uses these tokens to grant access to the project's data. The mechanism is passed the user's signature of a message and the message's contents as param-

eters. The mechanism subsequently determines the user's address. By interacting with the blockchain that the 'ProjectMarketplace' contract is deployed on, the mechanism iterates through all the created projects and grants access to a particular project's data if the user holds at least the data access threshold, specified when the project is created, of corresponding RFTs or the project NFT itself. We have built a web application that allows users to view the project data authorised to them by the authentication mechanism.

1.4 Ethical & Legal Issues

As our project is focussed on developing countries, it is important to consider potential benefit sharing actions. In the case of a charitable project that is funded through the proposed architecture, the benefit is purely to those local to the project. If a project expects a financial return however, it is important that there is a benefit sharing action that is satisfactory to the inhabitants local to the project infrastructure. This will be left to the platform that uses the proposed technical architecture, such as the Spring DAO, to ensure.

It will be necessary for platforms that use the proposed architecture to check a project's details before it is created. This is to ensure that the project is not raising funds for malicious reasons. The DAOs should also stagger access to the funds raised by a successful project and only give access to the next portion if sufficient data is supplied proving progression of the project's construction. This would ensure that funds are used for the project itself rather than for other, potentially malicious, purposes.

1.5 Thesis Roadmap

Chapter 2 describes and explains concepts that are fundamental to the understanding of this report. Chapter 3 explores the author's research into the context of constructing infrastructure in The Gambia; an underdeveloped country. In chapter 4, we explore the proposed system architecture, its design and implementation. Chapter 5 contains an evaluation of the system and in chapter 6, we conclude our work, summarising achievements and suggesting topics of future work.

Chapter 2

Background

In this chapter we introduce many of the fundamental concepts of decentralization that are core to our technological architecture. We first give an overview of Blockchain technology followed by smart contracts. We combine this knowledge to explain the basis of a Decentralized Autonomous Organisation (DAO). We then explore ERC standards, the distinction between fungible and non-fungible tokens and introduce re-fungible tokens. Finally, we look at stablecoins; a particular type of cryptocurrency.

2.1 Blockchain

2.1.1 Overview

A blockchain is a distributed digital ledger that was initially introduced as the technology behind the first cryptocurrency, Bitcoin [4]. It is a collection of transactions that are verified and distributed amongst the participants (connected computers) of the blockchain. These participants are often referred to as “peers” or “nodes”. The distribution of a blockchain is possible due it operating as a peer-to-peer network, meaning that peers can communicate with one another without the involvement of a central server [5]. As every node has a copy of the blockchain’s state, the system is decentralized, transparent, immutable and secure. A blockchain’s consensus algorithm is a mechanism through which all the peers come to a common agreement over the state of the blockchain [6]. This enables blockchains to be “trustless”; the integrity of transactions can be ensured without the need for participants to trust or know one another [7]. As a result, blockchain eliminates the need of many centralised third parties such as financial institutions overseeing transactions.

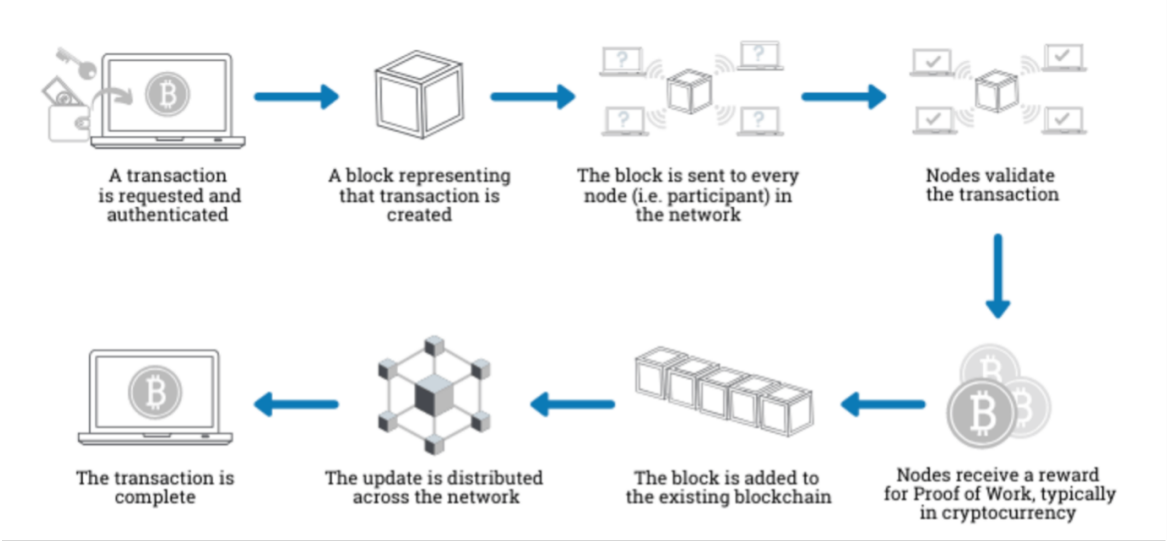


Figure 2.1: How a transaction is processed by blockchain. (Image sourced from [8])

There are two main types of blockchain: public and private [9]. Anyone can become a participant of a public blockchain whereas they must be accepted into a private blockchain by a centralised management system [10]. In this report we will be building on Ethereum which is a public blockchain and stores information regarding transactions within its ecosystem. At the time of writing, Ethereum's cryptocurrency Ether (ETH) is the second largest by market capitalization behind Bitcoin with values of almost $200 billion and over $400 billion respectively [11].

2.1.2 Security and Immutability

We now explore the reasons behind a blockchain's immutable and secure nature. A blockchain is made up of a series of connected blocks, with each block corresponding to a group of verified transactions. Each block is identified by its hash and contains data including a value called a "nonce", details about its multiple different transactions (their timestamp, amount, recipient, sender, etc...) and the hash of the previous block [12; 13]. A block's hash is a deterministic mapping of its contents to a unique, encrypted string of fixed length [14] and can be thought of as its digital fingerprint [15]. If any of the data within a block is changed, the hash of the block also changes.

When a block is added to the chain, its transactions are validated (eligibility checked) and it is "mined". "Mining" involves a node repeatedly using different values for a prospective block's nonce until it results in a "successful" hash [16]. A successful

hash is one that has a sufficient number of leading zeros (number specified by the blockchain). The block is now said to be "signed" [16]. The hash is cross-checked by other nodes to ensure that it is valid and the block added to the chain. The updated blockchain state is distributed to the network's nodes. The node that signs a block is rewarded according to the consensus algorithm.

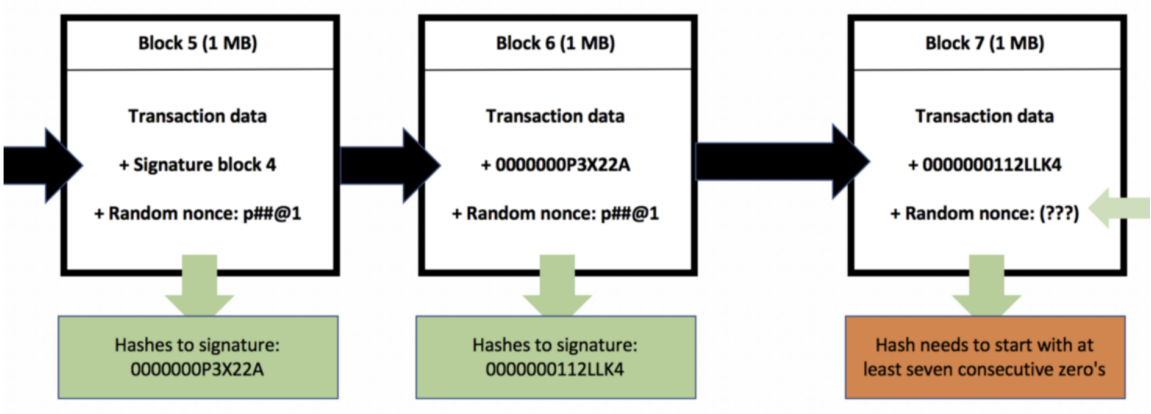


Figure 2.2: In this example, we are mining "Block 7" in order to add it to the blockchain. In this case, a nonce must be found that results in a hash of at least 7 leading zeros in order for the block to be signed. (Image adapted from [17])

If a bad agent manipulated the data of a previously validated block, its hash would subsequently change and the block would need to be re-mined. This would result in the block having a new hash which in turn would change the hash of the following block (as each block's data contains the previous block's hash). The bad agent would therefore need to re-mine every block thereafter resulting in a different record of block hashes. As all the participants in a blockchain have a copy of its state, the bad agents re-mined state should be recognised as different by the consensus mechanism and discarded.

This is the case unless for a "51% attack". A 51% attack is when a malicious entity gains at least 51% of the blockchains mining power, allowing them to manipulate the blockchain and add invalid blocks to the chain [18]. However, the larger the blockchain, the more difficult it is to gain 51% of its mining power. For a blockchain as large as Ethereum, the probability of a 51% attack is very low.

2.1.3 Ethereum

In Ethereum, the blockchains state is known as the "Ethereum Virtual Machine" (EVM). The EVM is distributed amongst the network's nodes. A "transaction" is an

action that results in the change of the EVMs state and is irreversible [19]. A transaction costs a "gas fee" (in ETH) to be validated and then completed. The value of the gas fee is determined by the congestion of the network and the speed at which the user wants the transaction executed. Only "Ethereum accounts" can send transactions on Ethereum [20], of which there are two types: "Externally Owned Accounts" (EOA) that represent users and "Contract Accounts" that correspond to smart contracts. Accounts are allocated an "address"; a unique 160-bit hexadecimal string. Ether and other Ethereum tokens are created in a process known as "minting". We chose to implement the Spring DAO on Ethereum as it has a diverse decentralized application ecosystem, enabled by its support of smart contracts. We will explore smart contracts and this ecosystem later in this chapter.

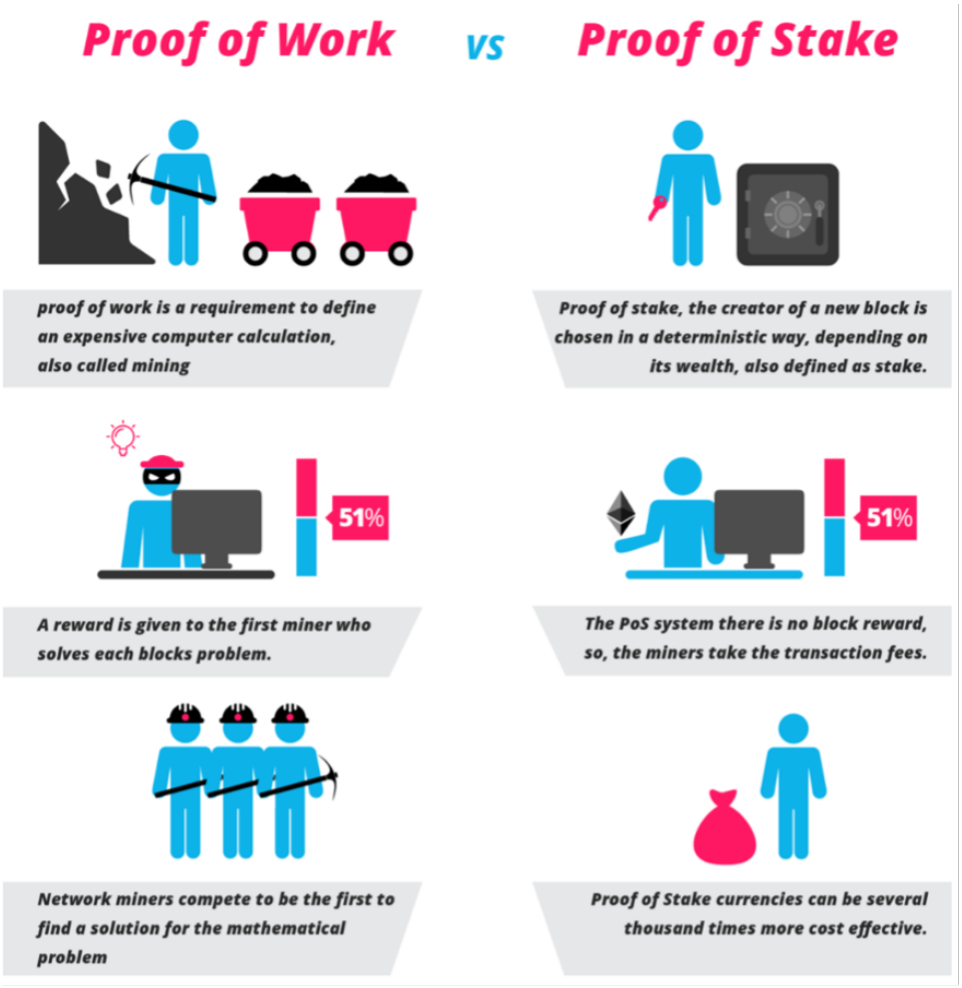


Figure 2.3: (Image sourced from [21])

Ethereum currently, like Bitcoin, uses the "proof-of-work" consensus algorithm, but is soon expected to evolve into "Ethereum 2.0". This marks a change to the "proof-of-stake" consensus algorithm in a transition that is known as "The Merge" which is

expected to reduce Ethereum's energy consumption by approximately 99.5% [22]. In proof-of-work, nodes compete against one another to mine a new block with the first to find a suitable nonce rewarded in the blockchains cryptocurrency [23]. The competitive nature of proof-of-work results in it consuming a massive amount of energy. For example, Bitcoin would rank as the 40th larger consumer of electricity by country [8].

In proof-of-stake miners are known as "validators" and are chosen to "forge" (equivalent to mine) a new block depending on a "stake" that they put forward [23]. The stake is a quantity of the blockchain's cryptocurrency (ETH for Ethereum) and is proportional to the likelihood of the node being chosen. The stake is lost if the node approves fraudulent transactions which acts as a safeguard against malicious intentions [24]. Due to only chosen nodes validating new blocks, proof-of-stake results in a significantly lower energy consumption by a blockchain network.

2.2 Smart Contracts

The phrase "smart contract" was first coined by Nick Szabo in the early 1990s, using it to refer to "a set of promises, specified in the digital form" [25]. Today, we more specifically consider smart contracts to be programs that are stored on a blockchain and whose functions can be called so long as any conditions that are written into the code are satisfied [26]. Smart contracts allow for trusted agreements and transactions to take place between anonymous entities without the need to be overseen by a mutually trusted third party [27]. When Ethereum launched in 2015 [28], it became the first platform to support smart contracts.

We consider smart contracts on the Ethereum blockchain. The most popular programming languages for developing smart contracts are Vyper and Solidity, the latter being created by Ethereum developers [29]. Publishing smart contracts on Ethereum is permissionless; anyone can write and deploy them to the network [30].

When a smart contract is deployed, it is published to the Ethereum Virtual Machine (EVM) and is allocated an address corresponding to its Ethereum account. The entity that deploys the contract is charged a gas fee which depends on the size and complexity (number of operations) of the contract [31]. Depending on access modifiers, a deployed smart contract can be interacted with by externally owned ac-

counts (users/wallets) and other smart contract accounts. As the EVM is distributed amongst the network's nodes, published smart contracts are open-source and thus transparent.

Smart contract deployment is an example of a transaction. As a result, a smart contract is permanent. Once it is deployed, it can neither be removed nor edited, only built on top of. This design has led to some high-profile losses due to bugs in smart contracts. In 2016, "The DAO" was hacked and lost $60 million due to a smart contract bug [32]. In April 2022, a release of NFTs belonging to the popular Aku NFT series resulted in $34 million being locked away in smart contracts due to another bug [33]. Due to this, it is important that smart contracts are rigorously tested.

Smart contracts provide the basis for decentralized applications, otherwise known as "dApps". A dApp is an application that combines smart contract(s) with a frontend user interface [34]. Ethereum has a rich and well supported eco-system of open source dApps which is why we have chosen to build on it for our technical architecture.

2.3 Decentralized Autonomous Organization (DAO)

A decentralized autonomous organization (DAO) is a type of decentralized application (dApp). A DAO can be thought of as a "internet-native company" whose members own and manage it [35]. It is built on smart contracts that encode the fundamental rules of the DAO. As a result, DAOs are typically opensource and their activity public which offer transparency. DAOs allow users to interact with one another in a trustless manner, enabling decentralized communities to form around a common goal [36]. Any example DAOs we consider will be on the Ethereum blockchain.

A DAO operates on a flat hierarchical structure, is fully democratized [35] and offers a trusted, transparent voting system without the involvement of a centralised authority. Members of a DAO are entitled to vote on decisions regarding the DAO and suggest initiatives themselves. This contrasts with traditional organisations where important decisions are finalised by a small group of individuals such as executives and/or board members. For example, a vote may be held to decide which project the DAO should fund using its treasury (a smart contract that holds the DAOs collective funds) or to make changes to the DAO's governance rules [37]. Votes are typically

held on-chain, however there are DAOs that rely on off-chain voting and others that combine on-chain and off-chain depending on the importance of a decision [38]. The weight associated with a member's voted depends on the membership model adopted by the DAO.

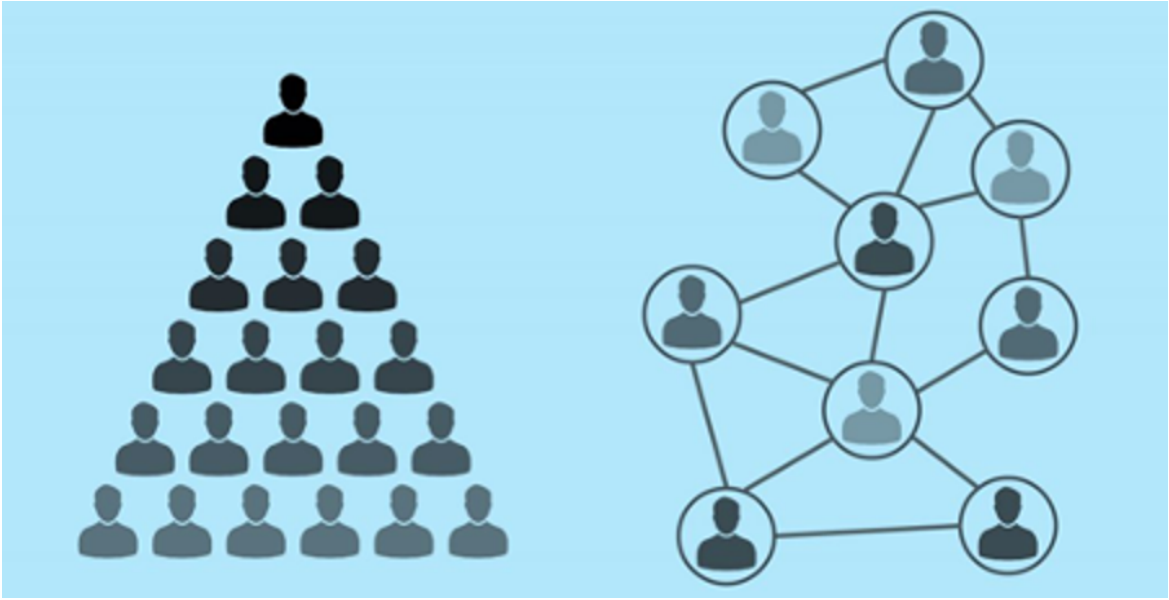


Figure 2.4: Left: traditional "top down" organisation structure. Right: Decentralized autonomous organisation structure. (Diagram adapted from [39])

We will consider three membership models: token-based membership, share-based membership and reputation-based membership [35]. In a token-based membership model, users that hold the DAOs "governance tokens" are eligible to vote [37]. Tokens are digital assets that reside on a blockchain. Governance tokens can be held in cryptocurrency wallets and traded on decentralized exchanges which means that access to the DAO is permissionless; anyone can buy into it [35]. Otherwise, governance tokens may be earned by offering the DAO liquidity or doing some work (proof-of-work) [35]. The weight of a member's vote is directly proportional to the number of governance tokens they own. An example of a well-known DAO that applies a token-based membership model is the MakerDAO. The MakerDAO allows users to lend and borrow cryptocurrencies, using its own cryptocurrencies "MKR" and "DAI" to regulate loan values [40].

In a share-based membership model, prospective members apply and offer a tribute to the DAO community in order to become part of it [35]. As a result, they are more permissioned (not anyone can join) than token-based membership. The shares of a DAO represent "direct voting power and ownership" [35]. A popular exam-

ple of a DAO that uses a share-based membership model is the "MolochDAO". The MolochDAO is a community that finance projects on the Ehereum blockchain. Users must apply to join the MolochDAO, demonstrating that they have sufficient expertise and capital to both be involved in the DAO and in its decision-making processes [35].

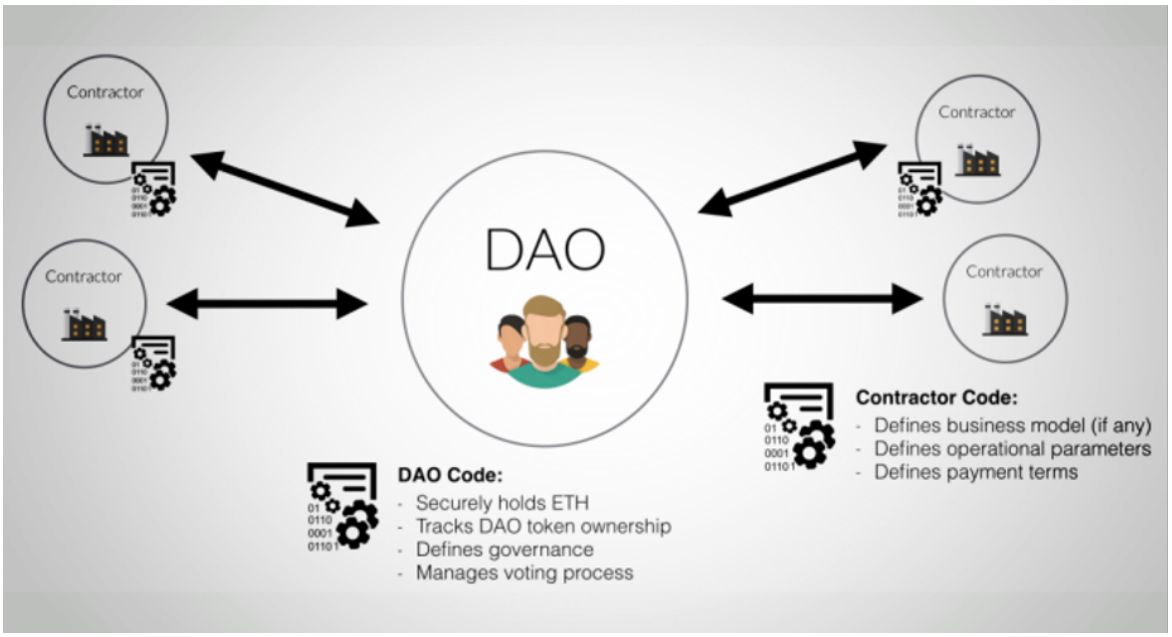


Figure 2.5: Image displays a DAO and how it is interacted with by contractors . There are smart contracts for the DAO itself and other smart contracts that separate contractors interact with. The latter should clearly state the details of the agreement. (Image sourced from [41])

A reputation-based membership model determines a member's voting powers by "reputation". A member earns reputation by participating [35] in the DAO's decisions, whether that be voting on proposals or initiating them. Reputation can neither be purchased nor can it be transferred, only earnt [35]. Users can request to join the DAO and may also ask for an increase of their reputation and/or tokens in return for some contribution they have made [35]. An popular example of a reputation-based DAO is "DXdao" that "develops, governs, and grows" decentralized finance protocols and products [42].

2.4 Tokens

Tokens are digital assets that reside on a blockchain. There are numerous types of tokens that each offer different functionality. From "security tokens" that represents

investment to “utility tokens” that allow holders access to blockchain-based products and services [43]. We shall focus on tokens on the Ethereum blockchain. When a token is created, we say that it is “minted”.

Tokens can be divided into two main types: fungible tokens and non-fungible tokens. We shall first explore token standards and their role in token development, followed by fungible tokens and then non-fungible tokens. We then examine “refungible” tokens that attempt to bridge the gap between fungible and non-fungible tokens. Finally, we consider “stablecoins”, a type of fungible token.

2.4.1 Token Standards

Token standards are standardised smart contract interfaces for token development on the Ethereum blockchain. Token standards ensure that newly minted tokens remain compatible with previously minted tokens and can therefore be traded on existing decentralized exchanges [44]. Token standards are commonly accessed through OpenZeppelin’s smart contracts. ‘OpenZeppelin’ is a company that offers thoroughly tested and audited open-source token standards which provide a secure foundation for developers to build off. Developers can straightforwardly implement these contracts and create standardised tokens to their specification. Some of the most popular Ethereum token standards are “ERC-20” for fungible tokens, “ERC-721” for non-fungible tokens and “ERC-1155” for utility tokens or non-fungible tokens [21].

2.4.2 Fungible Token

The original interface for minting fungible tokens on Ethereum is the ERC-20 token standard [44], however the ERC-223, ERC-777 or ERC-1363 token standards can also be used. Each standard has different advantages and disadvantages that depend on the use case for the fungible token. Though fungible tokens minted by different contracts are distinct, those minted from the same contract are interchangeable and equivalent [45].

Fungible tokens are divisible. A fungible token contract has a “decimal value” which denotes the smallest unit, $1\mathrm{e}\{$*-decimal value*$\}$, possible of the corresponding token that can be held or transferred. In the case of Ether, and numerous other cryptocurrencies, the decimal value is 18 which means that the smallest value that can be transferred is $(1\mathrm{e}{-}18)$.

Example use cases of fungible tokens are cryptocurrencies or governance tokens which can be used for voting. Once the relevant liquidity pools have been setup for a newly deployed fungible token contract, its tokens can be traded for other fungible tokens on decentralized exchanges such as 'Uniswap' or 'SushiSwap'.

2.4.3 Non-Fungible Token (NFT)

A non-fungible token (NFT) is a unique digital asset that is indivisible [43]. The original, and most commonly used, token standard for NFTs is the "ERC-721" token standard. Every NFT that a NFT contract mints is allocated a unique token ID (integer) and typically metadata. A NFT's metadata is the data that describes the NFT, such as its name, a description about it and details of the asset it represents. Metadata is normally stored in JSON file format. As it is very relatively expensive to store data on-chain, a pointer to a NFTs metadata is usually stored on-chain and the metadata itself off-chain. This pointer is called a 'Uniform Resource Identifier' (URI). A popular approach is to store the metadata itself in the 'InterPlanetary File System' (IPFS) and a link to that metadata as the URI. IPFS is decentralized off-chain storage and is much more cost effective than storing on-chain.

NFTs have made significant headlines in recent years for their use in digital art and the large-scale investment that has attracted. In 2021, 'The Merge' by Pak was sold for $91.8 million [46] and the famous 'Bored Ape Yacht Club' collection of 10,000 NFTs has been valued at $1.16 billion at the time of writing [47]. However, NFTs offer much more than digital art. For example, NFTs can be used for identity verification, patents, supply chain provenance and to represent medical records [48].

2.4.4 Re-Fungible Token (RFT)

Re-fungible tokens represent making non-fungible tokens, fungible again. The concept is to have an ERC-721 NFT that is owned by a fungible token contract such as an ERC-20 [49]. If the fungible token supply is limited, these fungible tokens represent the shared ownership of the NFT held by the fungible token contract. The fungible tokens themselves are referred to as re-fungible tokens (RFTs). A token standard for RFTs named "ERC-864" was suggested in 2018 for what was described as "divisible NFTs" [50]. This was formally proposed as the Ethereum Improvement Proposal (EIP) "ERC-1633" [51]. However, having failed to gain enough support,

the proposal became stagnant and has subsequently not become a token standard [52].

The notion of 're-fungibility' has recently gained traction again under a new name: 'fractionalization'. Fractionalization applies the same approach as re-fungible tokens in order to make ownership of a minted NFT or, more commonly, collection of NFTs divisible and therefore liquid [53]. As a result, owners are able to sell portions of their NFTs which consequently enables more people to be involved in owning higher value "blue chip" NFTs [54]. Improved liquidity allows investors to further diversify their NFT portfolio [55] and ensures they get more accurate prices for their investments [56].

There are several decentralized applications that apply variations of fractionalization. For example, 'NFTX' allows a user to deposit their NFT into an index fund containing other NFTs of the same value [57]. In return, the user receives a 'vToken' (fungible token) that is redeemable for any NFT in the same fund [57]. One of the benefits NFTX offers is that users can instantly sell a NFT by trading it for a vToken which they can swap on a decentralized exchange [58]. 'Fractional.art', meanwhile, allows a user to "fractionalize" their NFT(s) into a supply of fungible tokens [59] which the platform offers various auction services for.

2.4.5 Stablecoins

A stablecoin is a type of fungible token and cryptocurrency. A stable coin bridges the gap between stable fiat currencies and volatile cryptocurrencies [60]. Stablecoins are "pegged" to an external reference such as "another currency, commodity or financial instrument" [61]. In the case of a stablecoins being pegged to the US dollar, that means a holder of one corresponding stablecoin should be able to exchange it for $1. This exchange rate is ensured by two main different approaches. 'Collateralized stablecoins' are backed by a reserve of liquid assets that can be traded quickly in response to the stablecoin supply and demand [62]. Alternatively, 'algorithmic stablecoins' use smart contracts and algorithms to manage the supply of stablecoins in circulation and provide price stability [63].

The four biggest stablecoins by market capitalization are all pegged to the US dollar. Tether's USDT, Circle's USDC, Binance's BUSD and DAI have market capitalizations ranging from $67.5 billion to 7 billion respectively [64]. However, there are sta-

blecoins pegged to other fiat currencies that have significant values in circulation. Circle, the makers of USDC, have a stablecoin pegged to the euro called “Euro Coin” that has more than $54 billion in circulation [65]. Tether, the company behind USDT, is looking to launch a stablecoin pegged to the GBP called “GBPT” [13]. There are also popular stablecoins pegged to the Chinese Yuan [66] and the Mexican Peso [67].

2.5 Signatures

Digital signatures are fundamental to blockchain and are primarily used for authentication [68]. Signatures are used to prove that a user owns an account that they are making a transaction or signing a message from, thus preventing forgeries. A signature also ensures that a transaction/message has not been changed in transit, providing integrity [69].

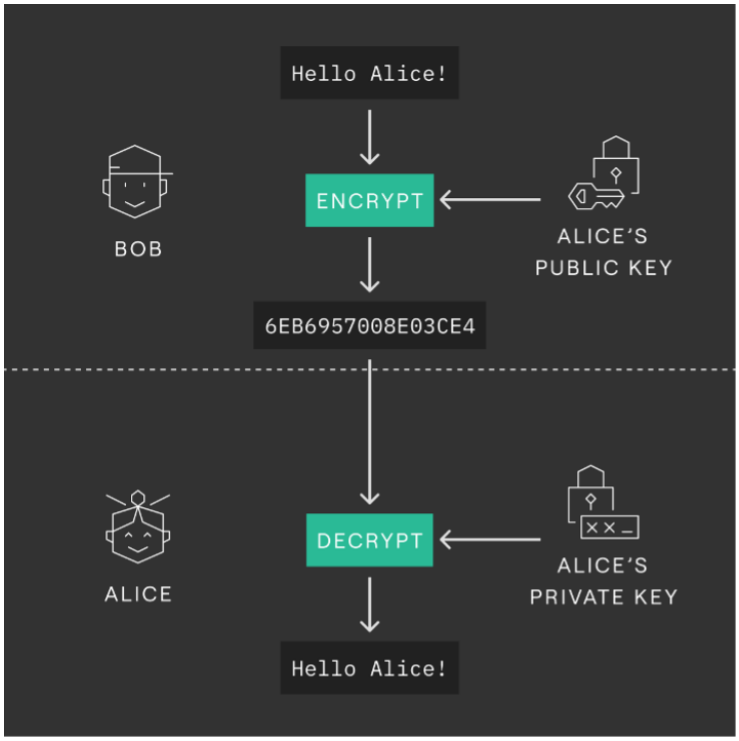


Figure 2.6: The user Bob can access another user Alice’s public key and send them an encrypted message that can only be decrypted by Alice’s private key. (Image sourced from [68])

.

Signatures in blockchain rely on public-key cryptography. Every account has a pair of keys: a private key and a public key. A public key, intuitively, is visible for everyone

to see whereas a private key is kept secret by the account holder as it proves ownership of its public key [68]. It is not possible to derive a private key from its public key pair. In the Ethereum blockchain, an account's address is its public key and its private key will typically be stored in a cryptocurrency wallet [69], such as MetaMask. A signature scheme is a system for generating keys, signing and correctly verifying the keys associated with a signature [70]. Ethereum uses 'Elliptic Curve Digital Signature Algorithm' (ECDSA) as its signature scheme [68]. Any data that is encrypted by a private key can only be decrypted by its public key pair and vice versa.

To authenticate the ownership of a public key (an account), a user signs a message (string) with a private key to create a signature on that message [71]. Using the public key of the account whose ownership we are verifying, we can convert that signature back into message format. If the message is the same as the original, it proves that the signer holds the public key's private key pair (they own the account) [71]. Similarly, if a message has been signed with a private key and we have both the message and signature, we can deduce the signers account address (public key).

Chapter 3

Contextual Field Research

It can be easy to imagine the benefits that advanced technologies could bring to an underdeveloped society without truly understanding whether those technologies are suitable and practical for that society. The author felt it necessary to visit a location that the proposed architecture is intended to benefit. Due to the author's work with the Spring DAO, they were introduced to 'eWater'; an internationally awarded company that builds water systems in The Gambia and throughout western Africa. eWater is the first company expected to use the Spring DAO and its projects provide good examples of the type that might use our architecture to gain funding.

The author travelled to The Gambia to conduct on site field research with eWater's team, allowing them to gain a deeper understanding in the context surrounding a use case of our funding system. During the trip, the author visited numerous remote Gambian villages in which eWater had either previously installed water systems (and were now maintaining) or were planning to soon. This allowed the author to see various stages of a project's lifecycle and the funding requirements within that. It also allowed the author to observe how all the stakeholders involved in a project operate and hear their opinions about the fundraising nature of the DAO. This research will significantly affect our design choices, described in the following section, for the better.

Figure 3.1: The author has included numerous photographs that they took on their research trip to The Gambia. (author, 2022)

3.1 Explaining Our Technology

It very quickly became apparent to the author that the locals were naturally very sceptical and distrustful of newly introduced foreign technologies such as cryptocurrencies, smart contracts and blockchain. It is completely understandable for people to be naturally hesitant of putting faith in unknown and seemingly inexplicable technology to fund such desperately needed infrastructure. However, once the benefits of the same technologies had been described and explained to the locals, they would become very interested and even enthusiastic.

eWater had a similar experience when beginning operations in The Gambia. eWater's systems rely on 'smart taps' that release water on contact with a 'fob' (an electronic tag). Users can top up their fob for credit at a local shop which can then be used at a 'smart tap' in exchange for water. Though locals were initially sceptical of this technology, it became widely accepted once it had become clear that it was not malicious and was in fact a far more reliable and superior alternative to pre-eWater systems. The author believes that the adoption of our fundraising system would receive a

similar reception. If the benefits it can provide are clearly explained and there are some successful use cases to evidence then it will be accepted.

Figure 3.2: eWater's plumber repairing a 'smart tap'. Due the extreme heat the wiring connecting the tap's solar panel to its fob reader had melted. (The author, 2022)

This technology makes eWater's projects prime use cases for our fundraising system. When water is dispensed at a tap, eWater records the transaction data. They also have sensors throughout a water system that measure water levels. This enables the company to infer invaluable statistics regarding the health of a system that are updated minute by minute. For example, they can view the percentage of a water supply that is working, whether any of their smart taps are not operational (and

if so which) and how many litres have been consumed or people served within a specified time-period. Though this information is currently just used for identifying problems quickly and responding effectively, it is ideal for feeding back to any DAO that implements our architecture. The data could then be visible on the DAO to those that had contributed to the specific eWater project, allowing them to monitor the project as intended.

3.2 Centralized Entity

When visiting different villages, the author met the respective decision-making committees and discussed how they currently raise capital for development projects. There were no computers in these villages and very few smart phones. It became clear that inhabitants would not have the means to interact directly with a DAO themselves. Rather, there would need to be some trusted centralized entity that represents a project and has the technological capability to propose it on the DAO. In the context of the author's research, this entity would be eWater.

Figure 3.3: Left image is a traditional water well which are relatively ineffective and very dangerous to use. Right image is a water tank installed by eWater. (The author, 2022)

Such an entity would also be necessary due to the complex reality of implementing projects in the Global South. There are only four construction companies in the whole of The Gambia that have the capacity to build every aspect of an eWater

project. These companies will often not leave Banjul, the capital city of The Gambia, let alone travel to the remote villages that require infrastructure the most. As a result, eWater are forced to find numerous smaller independent companies or individual contractors that are responsible for different aspects of a project (they are required to sign guarantees for the quality of their work). For example, one contractor may deal with digging the water system's borehole and another with laying its concrete, etc... These contractors are sourced primarily through the recommendations of respected villagers followed by general word of mouth.

Not only is this nature of project management far too complex for a DAO to organise, but investors should not have a say in decisions as low level as these. Provided the cost and construction time remain as anticipated, it is a better approach to allow those who have experience of the environment to make decisions such as selecting the contractors. This means leaving the organising of construction down to a collaboration of locals and the centralized entity that proposed the project.

Figure 3.4: An independent contractor that the author interviewed in Boiram, Southwestern The Gambia. (The author, 2022)

Furthermore, the entity should be responsible for disbursing the capital raised. Though raising funds in cryptocurrency has its benefits of very fast transactions and bypass-

ing both financial institutions and governments, it is by no means a form of payment to contractors. No project stakeholder that the author spoke to in The Gambia had a bank account and therefore none could accept digital payment, let alone cryptocurrency. The Gambia has a cash economy; contractors were paid in cash (Gambian Dalasi) or physical cheque that they could take to a bank branch in exchange for cash. It can be assumed that other similarly underdeveloped countries that make use of our fundraising system will also have cash economies.

For our fundraising system to be successfully adopted in a developing country, it is crucial that it is seamlessly integratable with the country's current payment practices. In eWater's example, that means payments being made in cash. If it is not easily integratable, we can envisage workers not wanting involvement in projects that are funded through our architecture and subsequently communities opting not to use it. USD and GBP were accepted everywhere in The Gambia, with the former being the most traded currency in the world [72]. Therefore, to ensure payment practice compatibility, funds should be raised in stablecoins that are pegged to the value of a fiat currency that is most convenient for a project. This means giving the proposer the choice of what stablecoin to raise in. In an eWater project example, funds might be raised in USDT which, once the target has been reached, eWater can straightforwardly convert to USD and then to Gambian Dalasi if necessary.

Figure 3.5: Construction site located in a village the author passed through. Any type of infrastructure project should be possible to fundraise using our architecture. (The author, 2022)

Having a centralised entity that proposes a project, is responsible for construction decisions, and distributes funds raised requires a significant amount of trust. A DAO that implements our architecture would need to have a rigorous vetting process about whether a proposer is a bad agent and malicious or benevolent and trustworthy. This most likely would be judged by the DAO's community. Such a vetting process is a topic of future work in section 6.2.

3.3 Staggered Payments

As a precaution against bad agents, access to funds should be staggered and granted upon the receipt of the data that proves a project's progress. The author's experience in The Gambia suggests projects do not typically require the full raised funds straight away. While visiting different eWater projects, the author was exposed to the different funding requirements of the eWater project lifecycle. Although it could vary depending on the project, a contractor would typically receive fifty percent of their quotation at the start of construction and the remaining upon inspection of their work. It is likely that the majority of implementation of other types of construction projects would be similar. If all the funds are required up front, it should be clearly documented why this is the case on the project's proposal.

Figure 3.6: A primary school in which eWater have built a water system for the students to use. (The author, 2022)

In the example of eWater's projects, data might involve initially confirming the consistent presence of water at the correct location of the water system's borehole. Once the data is deemed to be sufficient evidence of development, funds would be released for the following stage of construction. By the project's completion, that data might be the percentage of smart taps that are operational and how many litres have been consumed or people served within a specified time-period. A mechanism for staggering access to funds is a focus for future work, as described in section [insert section].

3.4 Sustainability

Proposed charitable projects should be encouraged to be sustainable. We specify charitable because a project that promises a financial return would expect to be. Our fundraising system should welcome financing upgrades to infrastructure, but it would be unwise for projects to constantly rely upon it for their maintenance. The main reason behind eWater's success in Western Africa is their model's self-sufficiency. eWater charges users for the water they consume. This revenue goes towards system maintenance and means no further fundraises, subject to circumstances not drastically changing, should be required for a project unless it is added to. This means that eWater can supply their beneficiaries with consistent, clean water year-round which is something previous charities/aid in The Gambia have not been able to provide.

Our architecture is attempting to assist funding permanent change rather than short term benefits. Project fundraises created should attempt to emulate approaches that are similarly self-sufficient to eWater's. To encourage this, any DAO that implements our architecture should make a charitable project listed on the DAO clearly state whether its model is sustainable and, if it is, precisely explain how so. If it is not, it should be made clear to potential donors that the project will rely on consistent donation which may affect their willingness to contribute.

3.5 Summary

In answer to our primary research question "do Decentralized Autonomous Organisations provide a route for funding and monitoring of infrastructure projects in developing countries", we conclude from the contextual research in The Gambia

that they can provide a route for funding infrastructure projects. This is subject to the technology behind our architecture being explained coherently to the would-be beneficiaries and that there is a trusted centralized entity to propose, manage and disburse funds raised for a project. Projects providing data such as eWater's will also ensure that investors can monitor their projects. The eventual success of a project will be determined by how sustainable it is, which should be clearly explained to potential investors/donors. However, it will ultimately be up to their discretion as whether to contribute.

Chapter 4

Design & Implementation

4.1 Technical Architecture

Having considered the approaches of related work listed in chapter 2, our contextual research documented in chapter 3 and the general requirements of such a system, we present the high-level architecture displayed in figure 4.1.

The architecture can be broken down into three main components:

1. Creating a project on the DAO.
2. Investing in a project on the DAO.
3. User authentication for project data access.

We discuss the design and implementation of each of these components in the remaining sections of this chapter.

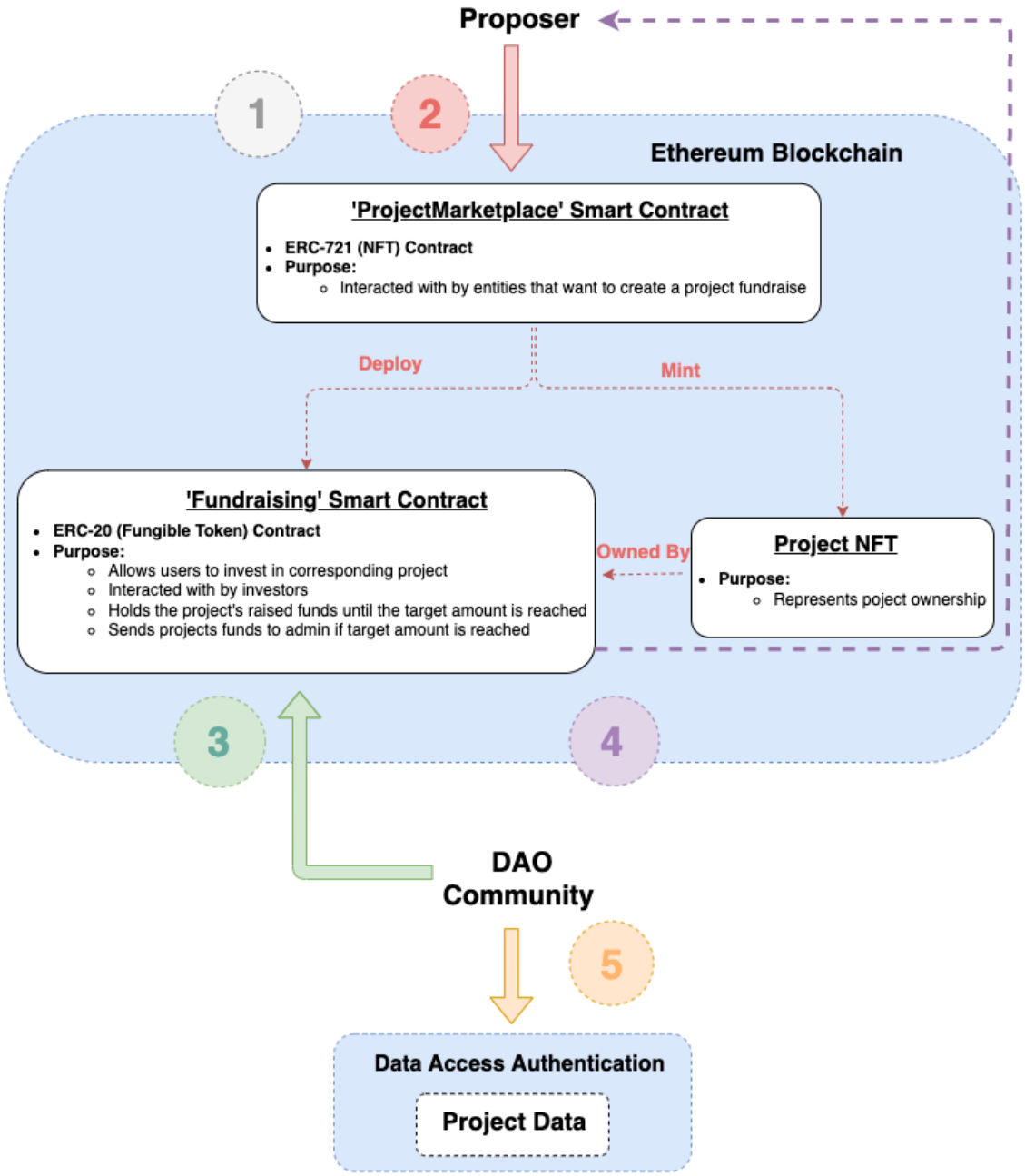


Figure 4.1: Our technical architecture:

1. DAO deploys 'ProjectMarketplace' smart contract.
2. Proposer interacts with 'ProjectMarketplace' to create project; 'Fundraising' smart contract is deployed and project NFT minted (owned by fundraising contract).
3. Community can invest in project by interacting with 'Fundraising' smart contract.
4. If target amount is raised, funds are transferred to the proposer.
5. Users given access to the project's data if they have invested sufficiently (quantity specified by proposer).

4.2 Creating projects: 'ProjectMarketplace'

There are two smart contracts that form the core of the technical architecture: 'ProjectMarketplace' and 'Fundraising'. They are written in Solidity for the Ethereum blockchain.

'ProjectMarketplace' uses OpenZeppelin's ERC-721 token standard and should be deployed once by each DAO that implements our architecture. The contract's purpose is to be interacted with by proposers. Its main function is called 'createProject' and allows users to create a fundraise for an infrastructure project. When createProject is called, a Fundraising contract is deployed which represents the projects fundraise and an NFT is minted which is owned by the Fundraising contract. The function offers the proposer significant choice in the fundraise details. It takes, as parameters, the fundraises name, symbol, target it is aiming to raise and the address of the cryptocurrency that the funds will be raised in. These are all passed to the 'Fundraise' contract constructor when it is deployed. The function also takes as parameters a URI, which is used as the minted NFTs metadata, and a value for the 'data access threshold' which will be explained further in section 4.4

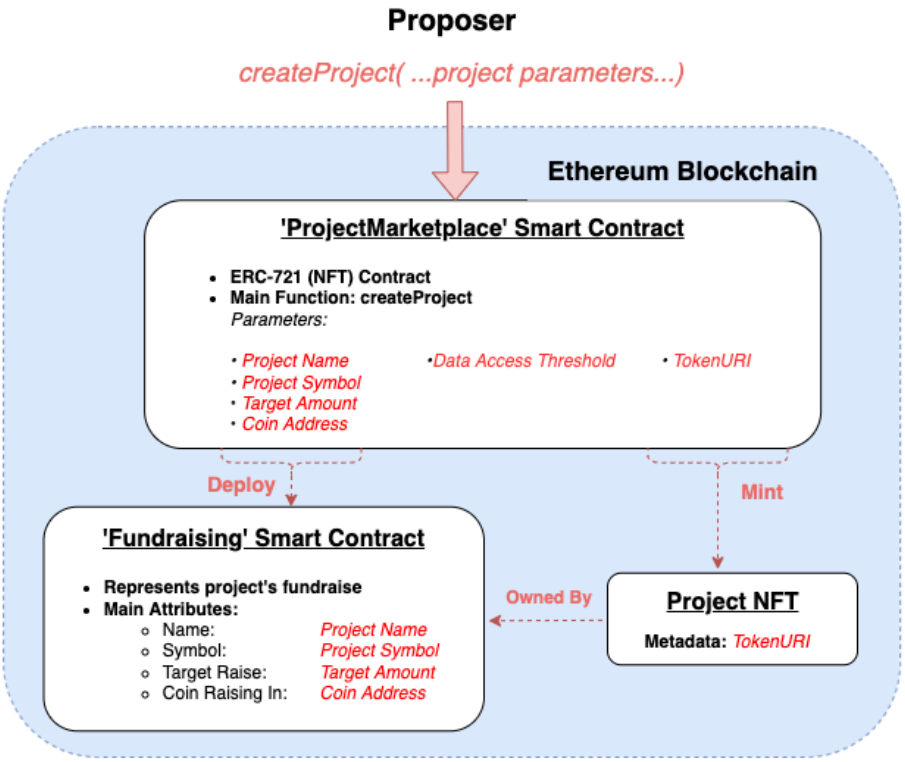


Figure 4.2: 'createProject' deploys a 'Fundraising' smart contract and mints a project NFT that is owned by the Fundraising contract.

The minted NFT is a unique digital representation of the project's infrastructure and its ownership. The entity that owns the NFT would be able to make certain decisions afforded to investors regarding the project and be entitled to any financial returns that a project might produce. Though these voting and financial return features have not been implemented in our technical architecture, they are a topic of future work (section 6.2) and have significantly impacted our design choices. As the NFTs are all minted by the ProjectMarkeplace contract, projects created on a DAO that implements our architecture will all be in the same collection. This will enable investors to easily distinguish projects legitimately listed on a DAO that they support from potential scams. Furthermore, by having projects represented by NFTs, a DAO's funded infrastructure will be easily integratable to the metaverse if stakeholders so wish.

As previously mentioned, ProjectMarketplace's createProject allows the proposing entity to decide on the cryptocurrency that funds are raised in. Though this feature does enable funds to be raised in any type of cryptocurrency, non-stablecoins should be strongly discouraged as they are generally very volatile. We originally considered hard coding the token that funds are raised in to being one of the more popular stablecoins pegged to the US Dollar such as USDT, USDC or DAI. This was because they have the greatest market capitalization and are subsequently the most trusted, exchanged and, thought to be, reliable stablecoins available. However, there are popular stablecoins pegged to the euro [65], Chinese Yuan [66], Mexican Peso and soon to the British pound [67]. The entity proposing a project should be able to choose a stablecoin that is pegged to the most convenient currency for their project.

As an optimization, when createProject is called, the minted token's token id is set to the integer format of the corresponding Fundraising contract's address. Contract addresses are 160-bit hexadecimal strings. By converting the string to a 256-bit unsigned integer and then to the decimal radix system, we have a value that is suitable for a token id. A token id can conversely be converted to the corresponding Fundraising address. This allows a project's NFT to always be linked to its Fundraise contract and vice-versa, without the need to store the information that links them on the blockchain. This is beneficial due to the relatively huge costs associated with storing data on chain.

4.2.1 Testing

As smart contracts cannot be edited, it is essential that they are rigorously tested before deployment. We used 'Truffle Suite', an ecosystem for developing decentralized applications [73], to write extensive unit and integration tests in JavaScript. We then deployed 'ProjectMarketplace' on the Rinkeby testnet to conduct system tests. As a result, we conclude that the contract initialises correctly and createProject behaves as described above.

When testing createProject, we had particular trouble ensuring that a Fundraising contract had been correctly deployed by createProject. This is because smart contracts interact asynchronously with external systems [74]. When a smart contract function that modifies the state of the blockchain such as createProject is called in Truffle, it returns the transaction information as opposed to the actual value returned by the function. createProject returns the token id of the minted project NFT which is needed to verify that the NFT has subsequently been minted and, once converted to a 160-bit hexadecimal string (an address), that the Fundraising contract has been deployed. In order to obtain the intended return value, the function must first be called with the '$.call(\ )$' suffix. This runs a simulation of the function, returning the intended value but not saving the changed state variables on chain; the state of the blockchain is not modified [75]. By calling createProject with the '$.call(\ )$' suffix and then again without it, we were able to first obtain the project token ID and then actually run the transaction that the token ID was associated with. This enabled us to verify that the project NFT and Fundraising were minted and deployed correctly respectively.

4.3 Investing in projects: 'Fundraising'

The 'Fundraising' smart contract implements OpenZeppelin's ERC-20 token standard and, intuitively, represents a project fundraise. The contract combines NFT re-fungibility/fractionalization with crowdfunding. It is interacted with by potential investors. Funds raised are held by the contract and sent to the entity that created the project once the target, specified in createProject, is reached.

The 'Fundraising' contract has five main functions centred on investing: 'Invest', 'withdrawInvestment', 'redeemNFT', 'pause' and 'kill'. In order to interact with the first three, a user must go to the contract of the coin that funds are being raised

in and approve the Fundraising contract. These three functions are public whereas 'pause' and 'kill' can only be called by the admin of the contract; the entity that originally created the project.

'Pause' enables the admin to set a binary flag called 'paused' which is by default false. By calling the function, the flag is set to true and by calling it again it is set back to false, and so on. When the contract is paused ('paused' equals true), users cannot invest but can still withdraw their investment. An admin may want to call this function if they have encountered an unexpected problem that may jeopardise the project but it is not yet certain. This function enables the admin to temporarily stop the fundraise without necessarily losing all the funds they have so raised so far. It is intended to be used only in emergencies and briefly at that. The more frequently and longer a project is paused, the more we can expect investors to withdraw.

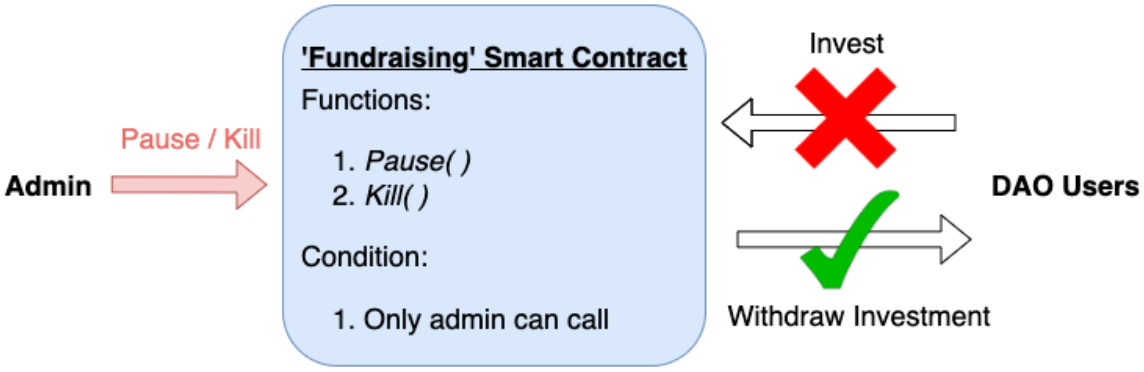


Figure 4.3: The functions 'pause' and 'kill' are used to prevent further investment temporarily and permanently respectively. They can only be called by the project 'admin' (proposer).

If an admin needs to stop a fundraise permanently, they can call the function 'kill'. 'kill' sets the flag "killed" from false to true which prevents the contract from receiving any further funds and cannot be undone. Users can still withdraw their investment and should strongly be encouraged to, as the fundraise is effectively defunct. Reasons for "killing" a fundraise may be if a bug is found in the code that could be exploited by bad agents or if the project is deemed to be no longer feasible. However, we anticipate it would most commonly be used in the case of human error; when a proposer has entered a project's details incorrectly. As smart contracts cannot be edited or taken down, the incorrect fundraise needs to be stopped ("killed") and another, with correct details, created.

The 'invest' function allows users to send the contract funds in the desired coin with the amount being specified in the function's single parameter. In return for investment, the contract mints and sends the user tokens in a 1:1 ratio to the value of investment; if a user invests 'X' amount of the desired coin, they will receive 'X' amount of that Fundraising contract's tokens. As 'Fundraise' owns the project NFT, the tokens that the contract mints represent a share of the project NFT's ownership and, thus, the project itself. Tokens minted by a Fundraising contract can therefore be referred to as re-fungible tokens (RFTs) as they enable the ownership of a successfully fundraised project NFT to be divisible.

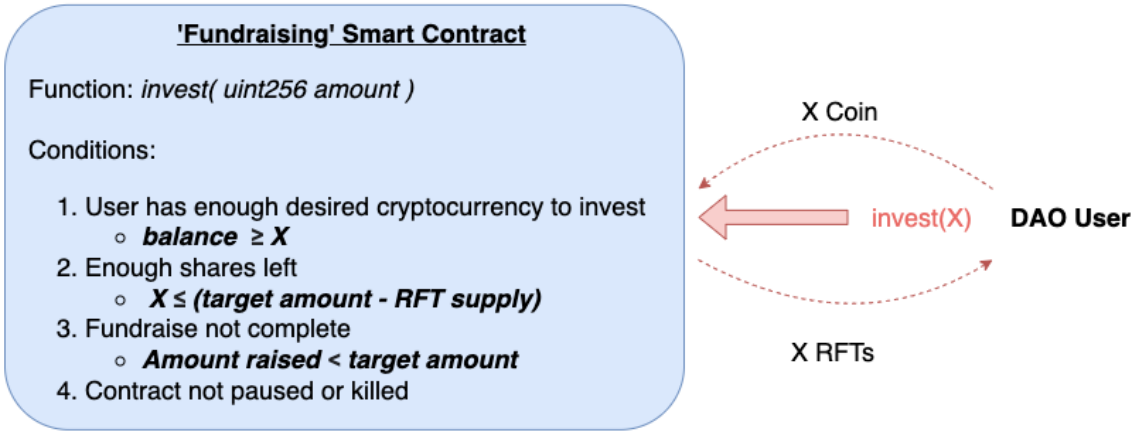


Figure 4.4: Subject to the conditions being met, a user on the DAO can invest in a project. In return for the specified investment amount of the cryptocurrency that funds are being raised in, they receive the same of quantity the project's RFTs

The value of a project's RFTs is only realised if its fundraise reaches its target amount and the project can consequently be implemented. When this occurs, no more investment can take place. Once the features have been added, the share of RFTs that a user holds in comparison to the total supply, which equals the target amount, will entitle the user to a proportionate voting power in decisions afforded to the project investors and percentage of the capital that the project might return. For example, if a project has been successfully fund-raised and a user owns 20% of the corresponding RFT supply, then they own 20% of the project; they will have 20% of the vote on investor decisions and will receive 20% of any funds returned.

In order to ensure a project's RFTs are distributed in a 1:1 relationship to the coin invested, both tokens must have the same decimal value. We have therefore set a project's Fundraising contract to be the decimal value of the coin that funds are being raised in. This means that all numerical values associated with a Fundraise

contract are in that particular decimal value, including the target amount to raise, the amount a user invests and withdraws.

A 'Fundraising' contract interacts with two types of tokens: the cryptocurrency tokens that funds are being raised in and the tokens it mints. Both are ERC-20 tokens. In order to avoid confusion, we shall refer to the former tokens as "coins" and the later as "RFTs" when distinction would otherwise be unclear.

Four conditions must be met in order to call 'invest': the user must have enough funds in the desired cryptocurrency, the contract must not be paused or killed and there must be enough shares left. Requiring enough shares remaining means that if a contract is a value of 'Y' away from reaching its target, a user cannot invest more than 'Y'. If the target amount has been raised, there are no more shares available which means that one of the conditions is no longer satisfied. As a result, this function will revert if it is called preventing further investment.

A Fundraise's 'withdrawInvestment' function can be interacted with by users who have a positive net investment in the contract. It has one parameter which allows the user to specify the amount of their investment that they would like to withdraw. That value can be range from a unit of the contract's decimal value up to their net investment. When called, the function returns the value of coin specified but takes the same value of corresponding RFTs from the user and burns them. Therefore a 1:1 ratio of RFTs to investment made is maintained. The function cannot be called if the Fundraising contract's target amount has been raised.

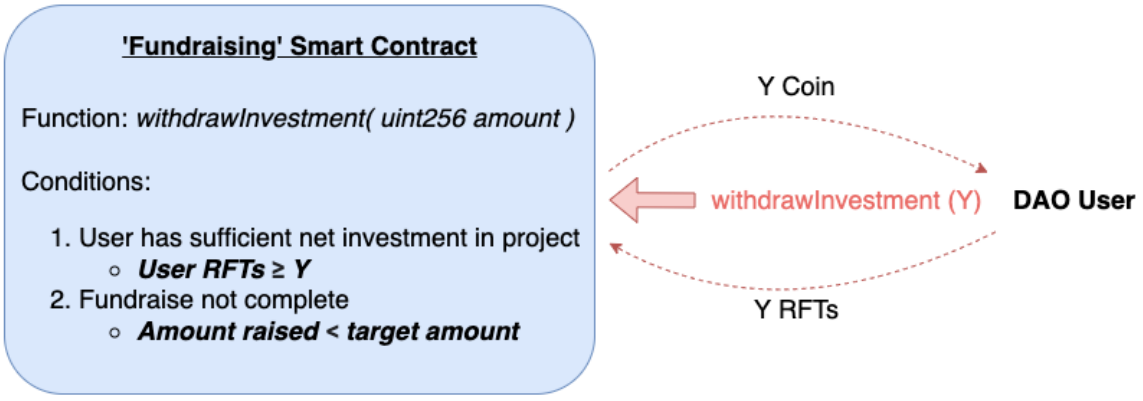


Figure 4.5: Subject to function conditions being met, a user can withdraw up to their net investment in return for the corresponding value of project RFTs

The final investment function is 'redeemNFT'. If a user contributes the total amount

that a project is raising, then they can call this function and exchange their RFTs for the project NFT itself. The RFTs are burned; the contract's RFT supply is zero. Instead of making this exchange compulsory, we leave it as an option for the user. That is because they may rather keep the project in a more liquid state by retaining their RFTs rather than holding the single project NFT asset.

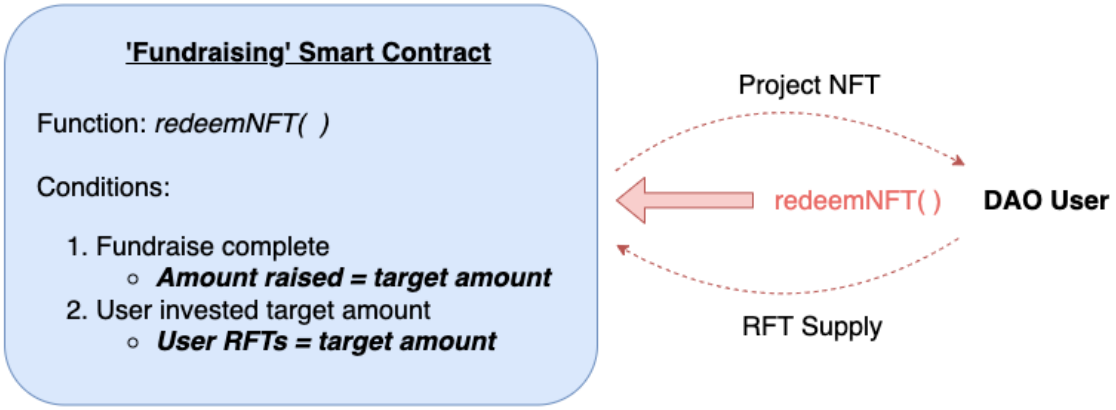


Figure 4.6: If a user has invested the project's total target amount, they can exchange their RFTs for the project NFT

4.3.1 Testing

We wrote 32 unit and integration tests that assessed the Fundraising contract's deployment and functions. In order to conduct system tests, we deployed a ProjectMarketplace contract on the Rinkeby testnet, created numerous projects and interacted with their respective Fundraising contracts. As a result, we can be assured that Fundraising initialises correctly and that all of its functions behave as intended.

4.4 User authentication for project data

We have built an authentication mechanism that uses a project's tokens as access tokens. A project's 'data access threshold' is the minimum amount a user must have invested in a successfully fundraised project in order to access its data. The ProjectMarketplace smart contract's function createProject has a parameter that takes an unsigned integer and sets the value of the data access threshold. Its value is therefore decided by the entity that proposes the project and is stored in the corresponding Fundraising contract. We allow the proposing entity to choose the threshold as the suitability of its value will vary from project to project. For example, if

a project expects a financial return, the proposer may allow everyone that has invested, regardless of the amount, to have data access so that they can monitor their investment. Whereas for a charitable project, the threshold might be set at $100 to encourage reasonable donations.

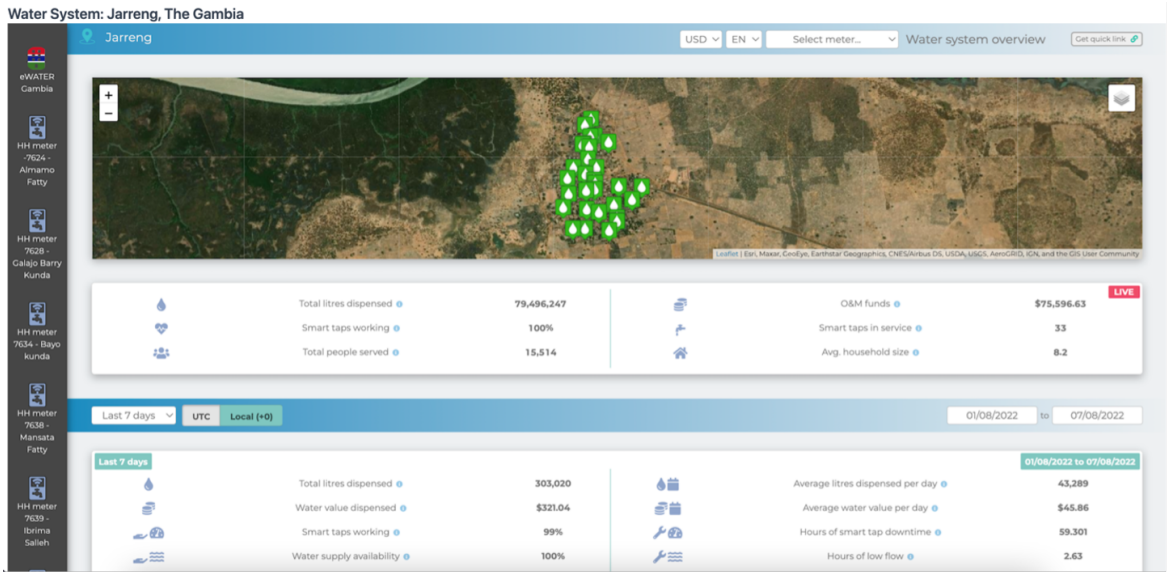


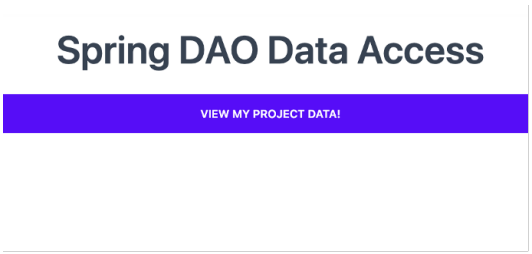
Figure 4.7: The sort of impactful data that an investor may have access to. It would ideally be as close live data as possible. This data has been kindly provided by eWater.

We programmed the authentication mechanism in JavaScript. This is to provide consistency as it will reside in a web application designed for data access that most likely will have been written in JavaScript. Every DAO that implements our infrastructure must have its own authentication mechanism that is connected to that DAO's ProjectMarketplace contract. The mechanism uses Ethers, a library for interacting with the Ethereum blockchain [76], to interact with ProjectMarketplace and the different project Fundraising contracts. The mechanism is passed the user's signature of a message and the contents of that message as parameters.

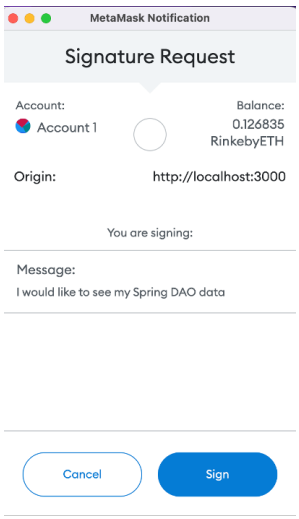
When the mechanism is called, it first passes the user's signature and its message's contents into the Ethers function 'verifyMessage' which returns the account address of the signer (user). Then it finds the total number of infrastructure projects on the DAO by interacting with the ProjectMarketplace contract. The mechanism iterates through all the created projects. For each project, it determines the corresponding Fundraising contract address. By interacting with the Fundraising contract, the mechanism finds the project's data access threshold and then the balance of the project's RFTs that the user holds. If the user owns the threshold value of RFTs or

more, then the project's name is added to an array that is initialised empty. Alternatively, if the user owns the project NFT itself (i.e., they have donated the full amount and redeemed the NFT) then the project's name is also added to the array. Once the mechanism has iterated through the all the projects, it returns the array. The DAO's infrastructure can then show the user any data associated with projects in the list; the data that they are authorised to view.

In order to demonstrate the authentication mechanism, we have created an app for accessing mock project data for the Spring DAO use case. The app consists of a frontend where a user can view their project data and a backend that contains our authentication mechanism. For the purpose of this proof of concept, we used screenshots of impactful information collected by eWater from a few of their active projects, as displayed in figure 4.7, for our mock data. The screenshots were stored on the image hosting website 'ImgBB' [77] andand a mapping was created from each project name to its corresponding mock data URL.



(a) User can click button to request their project data.

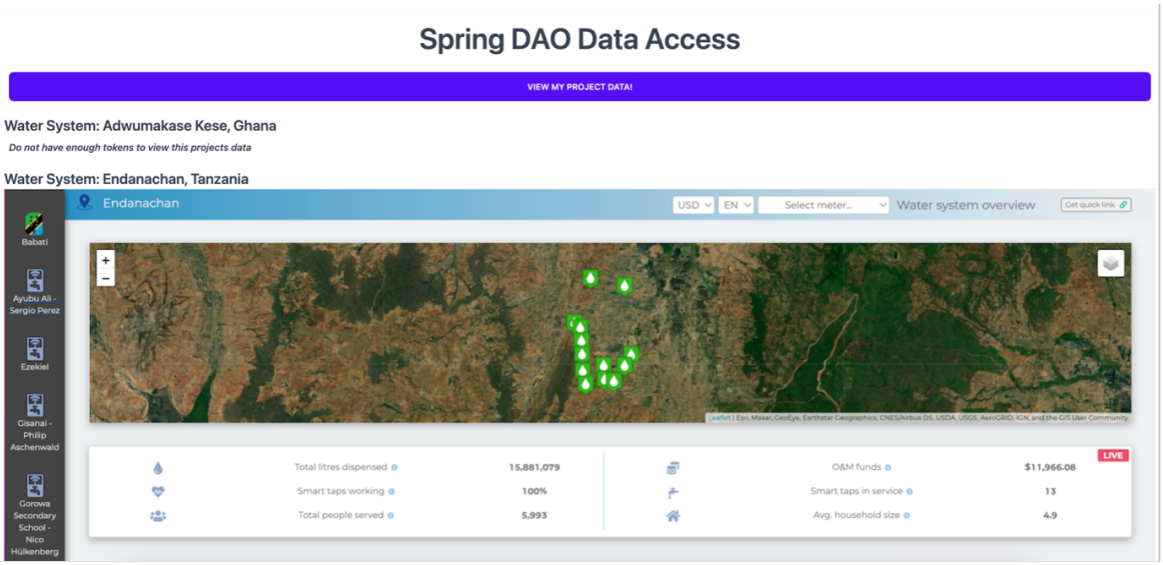


(b) User prompted to sign message displayed using a cryptocurrency wallet such as MetaMask.

Figure 4.8: Screenshots of app's user interface.

We built the frontend using the JavaScript library 'React'. Its main page is called 'Access Data' and has a button that, when clicked, calls a function called 'handleSignature'. This function checks whether the user has a Ethereum compatible cryptocurrency wallet installed on their browser, throwing an error if they do not, and prompts them to sign a message. As our app is for the Spring DAO use case, the message is "I would like to see my Spring DAO data" as displayed in figure 4.3 b). The associated signature is then passed as a parameter in a GET request to the backend.

(a) User owns no RFTs for any project.



(b) User owns insufficient RFTs for one project's data access but sufficient for anothers.

Figure 4.9: The possible outcomes when a user requests to see their project data using our App.

The backend is written in JavaScript and uses two main packages: Express and Ethers. It was built using Express which is a backend web app framework for Node.js. We used Express firstly because it is in JavaScript which enabled straightforward full stack development. Secondly, Express is very good for scaling as it allows developers to break functionality into very small modules. Developers can easily add

further modules, and therefore functionality, at a later date. When the frontend's GET request is received, the backend passes the signature and the message's contents into the authentication mechanism that returns an array containing the names of the projects that the user has access to. The project names, if there were any, were then mapped to the mock project data (screenshots) and sent back to the frontend in a JSON formatted response to the GET request. The corresponding data is subsequently displayed on the user interface.

Chapter 5

Evaluation

5.1 Smart Contract Gas Performance

It is important that we consider the transacction costs of our smart contracts, and their functions, to ensure that the various costs of execution are appropriate for respective purposes. We first consider the gas usage of our smart contracts and their respective functions. We then translate that usage to approximate costs in ETH followed by USD to give context to the gas performance.

$$Transaction fee = \textit{gas price per unit} \times \textit{units gas consumed}$$

In order to approximate the gas used by our smart contract, we executed the main transactions displayed in figure 5.1 10 times each on the Rinkeby testnet and calculated their average gas usages. For functions that required parameters, the values passed were varied. Though different arguments did impact the gas usage of a transaction, the variation was minimal.

Smart Contract	Transaction	Gas Used
ProjectMarketplace	Deploy smart contract	5,357,320
	'createProject'	2,441,850
Fundraising	'invest'	136,205
	'redeemNFT'	94,590
	'withdrawInvestment'	48,989
	'pause'	26,699
	'kill'	26,568

Figure 5.1: Main transactions and their corresponding average gas usages.

We shall convert the gas used to approximate transaction fees. Bar outliers, over the last 12 months [ftnote] the average gas fee on a particular day has been between 12 Gwei and 220 Gwei [78]. 1 Gwei is 1e-9 ETH. As this presents a significant range, we calculated the average of the average gas fees over the last 90 days[1] which we found to be 33.91 Gwei [78]. Using 12 Gwei and 220 GWei as lower and upper bound gas prices respectively and 33.91 Gwei as a more realistic present gas price, we produced the results in figure 5.2 a).

We then produced figure 5.2 b) by converting these approximations to USD by assuming the exchange rate of 1 ETH to $1750. Like the price of gas, the value of Ether is also very volatile; over the past year[2], ETH has had a low of $883.62 and a high of $4864.13 [79]. At the time of writing, the price of ETH is $1688.52 [79]. As the value of Ether is expected to improve over the remainder of 2022 while accounting for industry scepticism, we have assumed the exchange rate of $1750 to 1 ETH.

Smart Contract	Transaction	Trnx Fee LB (ETH)	Trnx Fee Av (ETH)	Trnx Fee UB (ETH)
ProjectMarketplace	Deploy smart contract	0.064287	0.187024	1.178610
	'createProject'	0.029319	0.085295	0.537522
Fundraising	'invest'	0.001634	0.004755	0.029965
	'redeemNFT'	0.001135	0.003302	0.020810
	'withdrawInvestment'	0.000588	0.001710	0.010778
	'pause'	0.000320	0.000932	0.005875
	'kill'	0.000319	0.000927	0.005845

Smart Contract	Transaction	Trnx Fee LB (USD)	Trnx Fee Av (USD)	Trnx Fee UB (USD)
ProjectMarketplace	Deploy smart contract	112.50	327.29	2062.56
	'createProject'	51.31	149.27	940.66
Fundraising	'invest'	2.86	8.32	52.43
	'redeemNFT'	1.99	5.77	36.41
	'withdrawInvestment'	1.03	2.99	18.86
	'pause'	0.56	1.63	10.28
	'kill'	0.56	1.62	10.22

Figure 5.2: Main transactions and their associated fees in ETH, upper table, and USD, lower table. "Trnx Fee LB", "Trnx Fee AV" and "Trnx UB" correspond to transaction fees with gas prices of 12Gwei, 33.91 Gwei and 220 Gwei respectively.

[1] 26/05/2022 to 24/08/2022
[2] 24/08/2021 to 24/08/2022

As we can see in figure 5.2 a), deploying a ProjectMarketplace contract is the most expensive transaction but, as a DAO that implements our architecture only needs to deploy the contract once, this is not a concern. Even in the case that createProject requires the upper bound transaction fee to process, it is still very affordable. If a project aims to raise a target amount of $20,000, which is very low in comparison to the amount that we expect projects to raise using our architecture, the transaction fee would be at most only 4.7% of the target raise.

The invest function is more expensive than anticipated as it is a function that we expect a user might call multiple times for the same project. If we were to assume that users are willing to forfeit a maximum of 5% of their investment on transaction fees, they would need to invest at least $57.20, $166.40 and $1048.60 in the lower bound, average and upper bound scenarios respectively. These costs price out the smaller investor. It also suggests that the data access threshold is pointless being set at any value below $50 if any exclusivity of the project data is intended, as almost all investors will have contributed over that value. However, these prices will do little to phase the larger investors of a few thousand dollars upwards.

'withdrawInvestment' requires 35% of the gas needed by 'Invest'. If we were to apply the same logic of a user accepting a loss of 5% on the funds that they are withdrawing, they would need to withdraw at least $20.60, $59.80 and $377.20 in lower bound, average and higher bound scenarios respectively to make the transaction worthwhile. This is again relatively expensive for the smaller investors but insignificant to the remainder.

The transaction fees associated with 'redeemNFT' are inconsequential given a user can only call the function if they have invested the full amount. The upper bound of $52.46 is minimal in comparison to the target amount of the fundraise. 'Pause' and 'kill's transaction fees are insignificant, especially given that the functions are intended to be used only in emergency.

Overall, we determine that the potential gas fees associated with our main transactions are positive for the purpose of our smart contract eco-system. Although 'invest', and 'withdrawInvestment' to an extent, discourage very small investors, the remaining are not adversely affected. It is not expected for The Merge to impact transaction fees [22]. However, the Ethereum Foundation say that the The Merge is a "critical precursor" to realizing "rollups" [80]; a layer 2 technology that com-

presses and processes numerous transactions together off-chain and commits them as one to the blockchain [8]. As a result, gas fees could decrease to between $0.002 and $0.05 [8] which would have a very positive impact on our smart contracts gas costs and make very small investments also cost-effective.

5.2 Limitations

5.2.1 Authentication Mechanism

The evaluation uncovered limits in the scalability of the authentication mechanism. After testing, we found that the average time to authenticate a particular project is approximately 1.77 seconds which results in an unreactive user experience as the number of projects created increases. Assuming the average time to authenticate a project remains consistent, 34 projects would result in an approximately 1-minute wait. This is because on-chain function calls are relatively very slow and the mechanism interacts with the blockchain between $(4n + 1)$ and $(5n + 1)$ times where n is the number of projects created by the DAO in question that implemented our architecture.

The first interaction is to find the number of projects created. The mechanism then iterates through all the projects and has either 4 or 5 interactions with the blockchain for each: locating the project's fundraising contract, finding its data access threshold, checking the user's balance of RFTs, checking whether they own the project NFT and finally, if the user is deemed to have data access, getting the project name. However, if the number of projects and each project's name, data access threshold and Fundraising contract address were stored in a database off-chain, we could reduce the number of interactions of the authentication mechanism with the blockchain to 2n. That means, per project, just one on-chain function call to check the users RFT balance in that project and one to check whether they own the project NFT.

Let $f_1(n)$ be the number of times that our current mechanism interacts with the Ethereum blockchain, where n is the number of projects created on the DAO. We have that:

$$(4n + 1) \leqslant f_1(n) \leqslant (5n + 1) \tag{5.1}$$

$$(4n) <= f_1(n) \leqslant (5n) \text{ as } n \to \infty \tag{5.2}$$

Let $f_2(n)$ be the number of function calls to the Ethereum blockchain by our newly suggested approach. As explained above, we have $f_2(n) = 2n$. Let $q := \frac{f_2(n) - f_1(n)}{f_2(n)} \times 100$ be the percentage decrease of functions calls having changed to our newly suggested approach. We have that:

$$\frac{(4n - 2n)}{4n} \times 100 \leqslant q \leqslant \frac{(5n - 2n)}{5n} \times 100 \tag{5.3}$$

$$\frac{2n}{4n} \times 100 \leqslant q \leqslant \frac{3n}{5n} \times 100 \tag{5.4}$$

$$50\% \leqslant q \leqslant 60\% \tag{5.5}$$

Therefore, assuming that each on-chain function call has the same execution time and that the execution time of the authentication mechanism's remaining code is negligible, we could reduce the time to authenticate each project by between 50% and 60% to a range of [0.88, 1.062] seconds. This would mark a significant improvement to our model but would still result in a slow response time as the number of projects created grows. Addressing the authentication mechanism's scalability is a topic of future work.

5.2.2 Smart Contracts

The proposed technical architecture assumes that the entity proposing a project is benevolent and trustworthy. This assumption has impacted our design choices but is not necessarily true. As a result, there are some aspects of the architecture that would need to be modified by a DAO that implements it.

The first is that 'ProjectMarketplace's function 'createProject' is public. In reality, entities that want to propose a project should go through some vetting process before they can propose. This is to prevent bad agents who have no intention of implementing a project from scamming investors and to ensure that the purpose of the project is aligned with the DAO. Once an entity has been cleared as 'safe' to propose a project, they should then be enabled by the DAO to call createProject. This is raised as a topic of future work.

Furthermore in our 'Fundraising' smart contract, if a project successfully raises its target amount then all its funds are immediately sent to the proposer. This is again under the assumption that the proposer can be trusted to implement the project. In reality, access to the funds should be staggered. Access to each "stage" of funds

should be granted upon the receipt of data that proves a project's progress. A mechanism for how the funds are staggered is subject of future work in chapter 6.2.

When a project is created, we give the entity creating it the option of which cryptocurrency funds are raised in. Though this feature does enable funds to be raised in any type of cryptocurrency, we expect successful raises to be predominantly in stablecoins pegged to fiat currency. If funds are raised in a cryptocurrency whose price significantly fluctuates frequently, the target amount originally specified in the Fundraising contract may no longer be equivalent to the required amount in fiat currency. Due to the permanent nature of smart contracts, the target amount would not be changeable and therefore the desired value in fiat currency potentially not raised.

We therefore anticipate that any fundraises that choose types of cryptocurrencies other than popular stablecoins will consequently put off potential investors due to the reasons listed above. Such a project would therefore be less likely to successfully raise the target amount. We expect this to provide a sufficient natural barrier to raising in non-stablecoins.

5.3 Industry Feedback

We presented our architecture to experts from areas of industry that our project overlaps with. We have summarised their opinions below.

We discussed our approach with Professor Catherine Mulligan. Catherine is the European Research Area Chair in Blockchain and a member of the World Economic Forum 'Crypto Impact and Sustainability Accelerator' (CISA). Catherine was very supportive of the architecture and particularly liked the use of project NFTs. This is because not only do project NFTs represent ownership of a project funded on a particular DAO, but they also provide a "form of organizational structure within the DAO itself" [81].

We spoke to Jamie Anson, NFT strategy advisor and the co-organiser of 'Ethereum London Meetup'; the world's largest Ethereum meetup community. Through his role in deciding which projects are promoted at the Ethereum London meetups, Jamie has considerable experience of evaluating blockchain projects from an end user perspective and assessing whether they will be well received by the industry.

Jamie was very positive about of our architecture. He particularly liked the 'data access threshold' feature and that the entity that proposes a project can decide its value. He thought that the threshold provided "extra value to people who put in a substantial amount" and will "encourage users to commit more money" [82]. He also liked how the feature ensured data access can be restricted to only the most committed investors. These committed investors can consequently form opinions about the progress/operation of a project which Jamie believes could prove useful for investor communications. He suggested that users with project data access could also have access to a communication channel, such as on Discord, that includes the project's founders.

Finally, we demonstrated our system to Rob Hygate: co-founder of both eWater and the Spring DAO. As a result of his time with eWater, he has significant experience of implementing infrastructure projects in developing countries. His opinion is therefore important for assessing whether our architecture is truly practical for financing infrastructure projects in underdeveloped nations.

Rob thought that our approach made good logical sense and that it can "provide the speed and liquidity to mobilise funds directly" [1] to a project. He understood our stance on the need for a centralized entity to propose, manage and disburse funds raised for a project, but in the long term would like that responsibility to become decentralized when technology in less developed locations allows.

Chapter 6

Conclusion

6.1 Summary of achievements

In response to our main research question: "Do Decentralized Autonomous Organisations provide a route for funding and monitoring of infrastructure projects in developing countries", we conclude that, assuming they follow the implementation advice outlined in this report, they can. By combining tokenized infrastructure, smart contract crowdfunding, NFT re-fungibility/fictionalization and access token authentication, we made the achievements summarised below.

1. **Contextual field research on how proposed architecture should be implemented**

   The author travelled to The Gambia to better understand the context around projects that might be funded using the proposed architecture and increase understanding of how their stakeholders operate. It is clear that for the proposed fundraising mechanism to be widely adopted, there would need to be a level of centralization on the end of the beneficiary. The system would need to be interacted with by trusted, centralized entities that propose, manage and disburse the funds of respective projects. This ensures that a project can be implemented in a way that is suitable to its location. The technology must also be explained coherently to prospective proposers and beneficiaries by any DAO that implements our architecture. Access to funds should be staggered and granted upon the receipt of data that proves a project's progress, ensuring that it is being correctly implemented.

2. **ProjectMarketplace: smart contract for creating projects**

   We designed and implemented a ERC-721 compliant smart contract written in Solidity called 'ProjectMarketplace'. The contract's function 'createProject' allows users to create a project fundraise. The entity that calls the function specifies, among other parameters, the projects name, the amount that the project requires to be implemented, its data access threshold and the cryptocurrency the raise is made in. Once called, a contract that represents the project's fundraise is deployed and a "project NFT" minted which is owned by the fundraise contract. The NFT represents the project and its ownership. 'createProject' requires 2,440,000 units of gas to call, which equates to an estimated transaction fee of 0.085295 ETH [1] ($149.21 [2]). As projects are expected to raise $50,000 upwards, this is a relatively low transaction fee.

3. **Fundraise: smart contract for on-chain fundraises**

   We created an ERC-20 compliant smart contract in Solidity called 'Fundraise' that represents a project's fundraise. The purpose of the contact is to be interacted with by investors and hold the funds raised until the target is reached. Users can call the contracts functions 'invest' and 'withdrawInvestment' to contribute to or remove funds from the project's fundraise. In return for investment, 'Fundraise' contract tokens, referred to as re-fungible tokens (RFTs), are minted and sent to the investor. As 'Fundraise' owns the project NFT, the RFTs represent a share of the project NFT's ownership and, therefore, the project itself. RFTs are minted or burned in a 1:1 ratio to the amount invested or withdrawn respectively

   The entity that proposed the project can stop the fundraise temporarily or permanently using the function's 'pause' or 'kill' respectively. If a user has invested the full target amount that the project is raising, they can burn their RFTs in return for the project NFT by calling the function 'redeemNFT'. Once the fundraise target amount is reached the funds are sent immediately to the entity that proposed the project and neither 'invest' nor 'withdrawInvestment' can be successfully called.

[1] Based on the average of average gas prices each day since the 1st of June 2022 [78].
[2] Assuming the price of 1 ETH is $1750.

In the context of their use cases, all function transaction fees are negligible apart from 'invest' and 'withdrawInvestment' which are relatively expensive for very small investors but insignificant for the remainder. We expect the former function to have a transaction fee of 0.004755 ETH ($8.32[3]) and the latter a fee of 0.001710 ETH ($2.99[4]).

4. **Authentication mechanism that uses project tokens as access tokens**

   Project NFT and RFTs act as access tokens in the project data access authentication mechanism. The entity that creates a project must specify a 'data access threshold'; the minimum investment required for a user to access a project's data. The authentication mechanism was built in Express.js and takes both the user's signature of a specific message and the contents of that message as parameters. The mechanism then determines the user's account address. By interacting with the blockchain that the 'ProjectMarketplace' contract is deployed on, the mechanism iterates through all the created projects and grants access to a particular project's data if the user holds at least the data access threshold of corresponding RFTs or the project NFT itself.

5. **Application for viewing project data**

   We have created a web app that allows users to view the data associated with projects they have invested in. The frontend was built using the React JavaScript library and the backend with Express.js. The authentication mechanism is included in the backend. When a user requests to view their project data, they are prompted to sign a message with a cryptocurrency wallet (such a wallet must be installed). The signature is passed as a parameter to the backend in a GET request. The signature and contents of the message signed are then passed to the authentication mechanism. The data associated with the project names outputted by the authentication mechanism is returned to the frontend and displayed on the user interface.

[3]Assuming the price of 1 ETH is $1750.
[4]3

6.2 Future Work

We highlight the following topics of future work:

1. **Add voting and financial return utility to project tokens**

   In our technical architecture, a successfully fundraised project's ownership is represented by its NFT and subsequently, unless the NFT has been redeemed, its RFTs. A possible approach to adding voting and financial return utility to project tokens could be that if a successfully fundraised project's RFT supply is zero (the NFT has been redeemed), then the owner of the NFT is allocated full voting power and, if there is any, sent the financial return. If the project NFT has not been redeemed, users could be allocated a voting power depending on the number of RFTs that they hold in comparison to the project's total supply and be entitled to an equivalent percentage of financial returns upon request to the DAO that is implementing our architecture. A project token's value will only be realised once this feature is implemented. Subsequently, liquidity pools could be setup to allow trading of RFTs on decentralized exchanges.

2. **Project integrity mechanisms**

   (a) **Vetting mechanism for project proposer**

   In the proposed technical architecture, it is assumed that the entity that creates a project is benevolent and trusted. As a result, the 'createProject' function on the 'ProjectMarketplace' contract is public. In reality this trust cannot be assumed. A mechanism should be created to assess whether a proposer is a bad agent and whether the project's purpose aligns with the DAOs. A project should only be created once it is deemed acceptable by the mechanism.

   (b) **Mechanism to ensure project construction**

   As in our technical architecture a project proposer is assumed to be trusted to carry out their project, they immediately receive the project's funds if the target is raised. In reality, a mechanism is required that ensures that the proposer implements the project correctly. A possible approach is for access to funds to be staggered and granted upon the receipt of data that proves a project's progress. A mechanism would be required to assess the data.

3. **Address authentication mechanism scalability issues**

   Our evaluation uncovered limits in the scalability of the authentication mechanism, with each project taking approximately 1.77 seconds to authenticate. It would be beneficial to explore how the mechanism could be modified to improve its scalability. In section 5.2.1, we suggest a potential improvement that could reduce the time to authenticate each project to between 0.88 and 1.062 seconds. However, this may still result in a poor user experience as the number of projects increase.

Bibliography

[1] Hygate R. *Proposed Architecture Review*;. [Personal interview, 26th August 2022], London (Unpublished). pages 1, 48

[2] ONE. *The Trillion Dollar Scandal*;. [Accessed 19th August 2022]. [Online]. Available from: https://www.one.org/international/take-action/victories/the-trillion-dollar-scandal/. pages 1

[3] Department for International Development. *Why corruption matters: understanding causes, effects and how to address them*;. [Accessed 17th August 2022]. [Online]. Available from: https://www.gov.uk/government/publications/why-corruption-matters-understanding-causes-effects-and-how-to-address-them. pages 1

[4] Hayes A. *Blockchain Facts: What Is It, How It Works, and How It Can Be Used* ;. [Accessed 8st August 2022]. [Online]. Available from: https://www.investopedia.com/terms/b/blockchain.asp#:~:text=First%20proposed%20as%20a%20research,use%3A%20Bitcoin%2C%20in%202009. pages 7

[5] Vermaak W. *What Are Peer-to-Peer (P2P) Networks?*;. [Accessed 18th August 2022]. [Online]. Available from: https://coinmarketcap.com/alexandria/article/what-is-peer-to-peer-p2p. pages 7

[6] GeeksforGeeks. *Consensus Algorithms in Blockchain*;. [Accessed 18th August 2022]. [Online]. Available from: https://www.geeksforgeeks.org/consensus-algorithms-in-blockchain/#:~:text=A%20consensus%20algorithm%20is%20a,state%20of%20the%20distributed%20ledger. pages 7

[7] Bybit Learn. *Peer-to-Peer Blockchain Networks: The Rise of P2P Crypto Exchanges* ;. [Accessed 21st August 2022]. [Online]. Available from: https:

//learn.bybit.com/bybit-p2p-guide/peer-to-peer-blockchain-network/. pages 7

[8] Euromoney . *How does a transaction get into the blockchain?*;. [Accessed 18th August 2022]. [Online]. Available from: https://www.euromoney.com/learning/blockchain-explained/how-transactions-get-into-the-blockchain. pages 8, 11

[9] Chatterjee P. *Public vs private blockchains: How do they differ* ;. [Accessed 18th August 2022]. [Online]. Available from: https://analyticsindiamag.com/public-vs-private-blockchains-how-do-they-differ/. pages 8

[10] Paul T, Rakshit S. Blockchain-Based Internet of Things: Challenges and Opportunities. Blockchain based Internet of Things. 2022:23-45. pages 8

[11] Slickcharts. *Cryptocurrency Market Data* ;. [Accessed 26th August 2022]. [Online]. Available from: https://www.slickcharts.com/currency. pages 8

[12] Bitstamp. *What are the blocks in blockchain?* ;. [Accessed 15th August 2022]. [Online]. Available from: https://blog.bitstamp.net/post/what-are-the-blocks-in-blockchain/. pages 8

[13] Ethereum. *BLOCK EXPLORERS* ;. [Accessed 16th August 2022]. [Online]. Available from: https://ethereum.org/en/developers/docs/data-and-analytics/block-explorers/#:~:text=Standard%20data,miner%20who%20mined%20the%20block.. pages 8

[14] European Digital Assets Exchange (EDSX). *Blockchain Transactions: HASH Functions*;. [Accessed 15th August 2022]. [Online]. Available from: https://edsx.ch/blog-news/blockchain-transactions-hash-functions. pages 8

[15] Conor. *Why You Should Use Blockchain for Digital Fingerprints*;. [Accessed 20th August 2022]. [Online]. Available from: https://medium.com/web3labs/why-you-should-use-blockchain-for-digital-fingerprints-107d3d403c03. pages 8

[16] Blockchain 101 - A Visual Demo;. pages 8, 9

[17] S J. *How do blockchain mining and transactions work explained in 7 simple steps* ;. [Accessed 23rd August 2022]. [Online]. Available from: https://blog.goodaudience.com/how-a-miner-adds-transactions-to-the-blockchain-in-seven-steps-856053271476. pages 9

[18] Madappa TC. *Understanding Proof of Stake* ;. [Accessed 6th August 2022]. [Online]. Available from: https://metacept.com/understanding-proof-of-stake/. pages 9

[19] Ethereum . *TRANSACTIONS* ;. [Accessed 25th August 2022]. [Online]. Available from: https://ethereum.org/en/developers/docs/transactions/#:~:text=An%20Ethereum%20transaction%20refers%20to,takes%20place%20within%20a%20transaction. pages 10

[20] Ethereum . *ETHEREUM ACCOUNTS*;. [Accessed 25th August 2022]. [Online]. Available from: https://ethereum.org/en/developers/docs/accounts/. pages 10

[21] Rosic A. *Proof of Work vs Proof of Stake: Basic Mining Guide* ;. [Accessed 18th August 2022]. [Online]. Available from: https://blockgeeks.com/guides/proof-of-work-vs-proof-of-stake/. pages 10, 15

[22] Sun Z. *Ethereum Foundation clarifies that the upcoming Merge upgrade will not reduce gas fees* ;. [Accessed 25th August 2022]. [Online]. Available from: https://cointelegraph.com/news/ethereum-foundation-clarifies-that-the-upcoming-merge-upgrade-will-not-reduce-gas-fees. pages 11, 44

[23] Greenfield R. *Explaining How Proof of Stake, Proof of Work, Hashing and Blockchain Work Together* ;. [Accessed 20th August 2022]. [Online]. Available from: https://robertgreenfieldiv.medium.com/explaining-proof-of-stake-f1eae6feb26f. pages 11

[24] Real Vision. *What is Proof-of-Stake* ;. [Accessed 14th August 2022]. [Online]. Available from: https://www.realvision.com/blog/what-is-proof-of-stake#:~:text=Th%20requirement%20for%20coin%20owners,if%20they%20approve%20fraudulent%20transactions. pages 11

[25] Wikipedia. *Smart Contract* ;. [Accessed 7th August 2022]. [Online]. Available from: https://en.wikipedia.org/wiki/Smart_contract#:~:text=Smart%20contracts%20were%20first%20proposed,parties%20perform%20on%20these%20promises%22. pages 11

[26] IBM. *Smart contracts defined* ;. [Accessed 23rd August 2022]. [Online]. Available from: https://www.ibm.com/uk-en/topics/smart-contracts#:~:text=Smart%20contracts%20are%20simply%20programs,intermediary's%20involvement%20or%20time%20loss. pages 11

[27] Frankenfield J. *Smart Contracts* ;. [Accessed 7th August 2022]. [Online]. Available from: https://www.investopedia.com/terms/s/smart-contracts.asp. pages 11

[28] Shrimpy. *The Best Smart Contract Platforms*;. [Accessed 8th August 2022]. [Online]. Available from: https://academy.shrimpy.io/post/the-best-smart-contract-platforms. pages 11

[29] Wikapedia . *Solidity* ;. [Accessed 31st August 2022]. [Online]. Available from: https://en.wikipedia.org/wiki/Solidity. pages 11

[30] Ethereum. *INTRODUCTION TO SMART CONTRACTS* ;. [Accessed 25th August 2022]. [Online]. Available from: https://ethereum.org/en/developers/docs/smart-contracts/#:~:text=A%20%22smart%20contract%22%20is%20simply,a%20type%20of%20Ethereum%20account. pages 11

[31] LCX Team. *A Short Guide To Smart Contracts And Gas Fees* ;. [Accessed 26th August 2022]. [Online]. Available from: https://www.lcx.com/a-short-guide-to-smart-contracts-and-gas-fees/. pages 11

[32] del Castillo M. *The DAO Attacked: Code Issue Leads to $60 Million Ether Theft* ;. [Accessed 10th August 2022]. [Online]. Available from: https://www.coindesk.com/markets/2016/06/17/the-dao-attacked-code-issue-leads-to-60-million-ether-theft/. pages 12

[33] Coghlan J. *AkuDreams dev team locks up $33M due to smart contract bug* ;. [Accessed 10th August 2022]. [Online]. Available from:

https://cointelegraph.com/news/akudreams-dev-team-locks-up-34m-due-to-smart-contract-bug. pages 12

[34] Ethereum. *INTRODUCTION TO DAPPS* ;. [Accessed 20th August 2022]. [Online]. Available from: https://ethereum.org/en/developers/docs/dapps/. pages 12

[35] Ethereum. *Decentralized autonomous organizations (DAOs)* ;. [Accessed 20th August 2022]. [Online]. Available from: https://ethereum.org/en/dao/. pages 12, 13, 14

[36] Reiff N. *Decentralized Autonomous Organization (DAO)* ;. [Accessed 18th August 2022]. [Online]. Available from: https://www.investopedia.com/tech/what-dao/. pages 12

[37] coinbase. *Decentralized Autonomous Organization (DAO)* ;. [Accessed 6th August 2022]. [Online]. Available from: https://help.coinbase.com/en/coinbase/getting-started/crypto-education/glossary/decentralized-autonomous-organization--dao-. pages 12, 13

[38] Hedera. *Understanding the Basics of a Decentralized Autonomous Organization (DAO)* ;. [Accessed 5th August 2022]. [Online]. Available from: https://hedera.com/learning/decentralized-finance/decentralized-autonomous-organization#:~:text=The%20governance%20rules%20posted%20by,Governance%20Polls%20and%20Executive%20Polls. pages 13

[39] Moralis Academy. *DAO vs Traditional Organization* ;. [Accessed 18th August 2022]. [Online]. Available from: https://academy.moralis.io/blog/dao-vs-traditional-organization. pages 13

[40] Tran KC. *What is MakerDAO?* ;. [Accessed 18th August 2022]. [Online]. Available from: https://decrypt.co/resources/makerdao-guide-learn-explained-decrypt-3-minutes. pages 13

[41] Okaformbah C. *Governance in a Decentralized Autonomous Organization* ;. [Accessed 18th August 2022]. [Online]. Available from: https://justcharles.medium.com/governance-in-a-decentralized-autonomous-organization-425f56b3e8bb. pages 14

[42] DXdao. *DXdao* ;. [Accessed 23rd August 2022]. [Online]. Available from: https://dxdao.medium.com/. pages 14

[43] Cointelegraph . *Fungible vs nonfungible tokens: What is the difference?* ;. [Accessed 10th August 2022]. [Online]. Available from: https://cointelegraph.com/nonfungible-tokens-for-beginners/fungible-vs-nonfungible-tokens-what-is-the-difference. pages 15, 16

[44] Ethereum . *TOKEN STANDARDS* ;. [Accessed 7th August 2022]. [Online]. Available from: https://ethereum.org/en/developers/docs/standards/tokens/#:~:text=Here%20are%20some%20of%20the,for%20artwork%20or%20a%20song. pages 15

[45] OpenZeppelin . *Tokens* ;. [Accessed 29th August 2022]. [Online]. Available from: https://docs.openzeppelin.com/contracts/3.x/tokens#ERC20. pages 15

[46] Block F. *PAK's NFT Artwork 'The Merge' Sells for $91.8 Million* ;. [Accessed June 30th 2022]. [Online]. Available from: https://www.barrons.com/articles/paks-nft-artwork-the-merge-sells-for-91-8-million-01638918205. pages 16

[47] intotheblock . *NFTs Analytics* ;. [Accessed 28th August 2022]. [Online]. Available from: https://app.intotheblock.com/insights/nft?group=general-activity&chart=total_volume_traded_by_nfts. pages 16

[48] Rean. *10 Practical NFT Use Cases Beyond Digital Artworks* ;. [Accessed 17th August 2022]. [Online]. Available from: . pages 16

[49] Rennekamp B. *Re-Fungible Token (RFT)* ;. [Accessed 27th August 2022]. [Online]. Available from: https://billyrennekamp.medium.com/re-fungible-token-rft-297003592769. pages 16

[50] devzl. *divisibleNFTs* ;. [Accessed 25th August 2022]. [Online]. Available from: https://github.com/devzl/divisibleNFTs/blob/master/contracts/ExampleDivisibleNFTs.sol. pages 16

[51] okwme. *ERC 1633 - RFT (Re-Fungible Token)* ;. [Accessed 27th August 2022]. [Online]. Available from: https://github.com/ethereum/EIPs/issues/1634. pages 16

[52] Ethereum . *Ethereum Improvement Proposals* ;. [Accessed 25th August 2022]. [Online]. Available from: https://eips.ethereum.org/erc. pages 17

[53] Quantix . *Fractionalization liquidity: What you should know when trading NFTs* ;. [Accessed 27th August 2022]. [Online]. Available from: https://medium.com/@quantixnft/fractionalization-liquidity-what-you-should-know-when-trading-nfts-f99ac31be371. pages 17

[54] Genç E. *How Can You Share an NFT? Fractional NFTs Explained* ;. [Accessed 27th August 2022]. [Online]. Available from: https://www.coindesk.com/learn/how-can-you-share-an-nft-fractional-nfts-explained/#:~:text=To%20fractionalize%20an%20NFT%20on,fractionalized%20asset%20through%20the%20platform. pages 17

[55] Mojtahedi A. *Fractional NFTs: Unlocking Liquidity Access to Small Investors* ;. [Accessed 27th August 2022]. [Online]. Available from: https://research.aimultiple.com/fractional-nft/. pages 17

[56] XP NETWORK . *NFT Fractionalization* ;. [Accessed 27th August 2022]. [Online]. Available from: https://blog.xp.network/nft-fractionalization-4bbfd433fbc9. pages 17

[57] Harris B. *An Overview of the NFT Fractionalization Landscape* ;. [Accessed 31st August 2022]. [Online]. Available from: https://medium.com/@bdharris/an-overview-of-the-nft-fractionalization-landscape-83487874926b. pages 17

[58] NFTX . *Introduction to NFTX* ;. [Accessed 31st August 2022]. [Online]. Available from: https://docs.nftx.io/. pages 17

[59] Fractional art . *Fractionalize* ;. [Accessed 31st August 2022]. [Online]. Available from: https://fractional.art/fractionalize. pages 17

[60] Livemint . *Stablecoin USDC: Benefits and use cases* ;. [Accessed 11th August 2022]. [Online]. Available from: https://www.livemint.com/market/cryptocurrency/stablecoin-usdc-benefits-and-use-cases-11660977004524.html. pages 17

[61] Hayes A. *Stablecoin*;. [Accessed 11th August 2022]. [Online]. Available from: https://www.investopedia.com/terms/s/stablecoin.asp. pages 17

[62] Bitcoinist . *Algorithmic vs Collateralized: Stablecoins Not So Stable* ;. [Accessed 16th August 2022]. [Online]. Available from: https://bitcoinist.com/algorithmic-vs-collateralized-stablecoins-not-so-stable/. pages 17

[63] Cryptopedia . *What Are Stablecoins?* ;. [Accessed 11th August 2022]. [Online]. Available from: https://www.gemini.com/cryptopedia/what-are-stablecoins-how-do-they-work. pages 17

[64] CoinMarketCap . *Top Stablecoin Tokens by Market Capitalization* ;. [Accessed 11th August 2022]. [Online]. Available from: https://coinmarketcap.com/view/stablecoin/. pages 17

[65] Circle . *A euro stablecoin backed by full reserves* ;. [Accessed 23rd August 2022]. [Online]. Available from: https://www.circle.com/en/euro-coin. pages 18, 32

[66] De N. *Tether Launches Chinese Yuan-Pegged Stablecoin on the Ethereum Blockchain* ;. [Accessed 24th August 2022]. [Online]. Available from: https://www.coindesk.com/markets/2019/09/09/tether-launches-chinese-yuan-pegged-stablecoin-on-the-ethereum-blockchain/. pages 18, 32

[67] Partz H. *Tether to launch GBPT stablecoin pegged to British pound sterling* ;. [Accessed 24th August 2022]. [Online]. Available from: https://cointelegraph.com/news/tether-to-launch-gbpt-stablecoin-pegged-to-british-pound-sterling. pages 18, 32

[68] coinbase . *Why digital signatures are essential for blockchains* ;. [Accessed 1st August 2022]. [Online]. Available from: https://www.coinbase.com/cloud/discover/dev-foundations/digital-signatures#:~:text=Digital%20signatures%20are%20a%20fundamental,other%20users%20from%20spending%20them. pages 18, 19

[69] Agrawal R. *Digital Signature from Blockchain context* ;. [Accessed 5th August 2022]. [Online]. Available from: https://ravikantagrawal.medium.com/digital-signature-from-blockchain-context-cedcd563eee5. pages 18, 19

[70] Wikapedia . *BLS digital signature* ;. [Accessed 21st August 2022]. [Online]. Available from: https://en.wikipedia.org/wiki/BLS_digital_signature. pages 19

[71] Wikapedia . *Public-key cryptography* ;. [Accessed 21st August 2022]. [Online]. Available from: https://en.wikipedia.org/wiki/Public-key_cryptography. pages 19

[72] IG . *What are the top 10 most traded currencies in the world?* ;. [Accessed 20th August 2022]. [Online]. Available from: https://www.ig.com/uk/trading-strategies/what-are-the-top-10-most-traded-currencies-in-the-world-200115. pages 25

[73] Howell J. *Know Everything About Truffle Suite* ;. [Accessed 22nd July 2022]. [Online]. Available from: https://101blockchains.com/truffle-suite/. pages 33

[74] Oh S. *Tips for Unit Testing Ethereum Smart Contracts in Solidity* ;. [Accessed 17th August 2022]. [Online]. Available from: https://betterprogramming.pub/a-few-tips-for-unit-testing-ethereum-smart-contract-in-solidity-d804062068fb. pages 33

[75] Block Heroes . *When to use call() for Solidity functions (Truffle)* ;. [Accessed 26th August 2022]. [Online]. Available from: https://blockheroes.dev/when-to-use-call-solidity-functions-truffle/. pages 33

[76] ethers . *Documentation* ;. [Accessed 20th August 2022]. [Online]. Available from: https://docs.ethers.io/v5/. pages 38

[77] ImgBB. *Homepage* ;. [Accessed 15th August 2022]. [Online]. Available from: https://imgbb.com/. pages 39

[78] YCHARTS . *Ethereum Average Gas Price* ;. [Accessed 29th August 2022]. [Online]. Available from: https://ycharts.com/indicators/ethereum_average_gas_price. pages 43, 50

[79] coinbase . *Ethereum to US Dollar* ;. [Accessed 29th August 2022]. [Online]. Available from: https://www.coinbase.com/converter/eth/usd. pages 43

[80] Ethereum . *Misconceptions about The Merge* ;. [Accessed 25th August 2022]. [Online]. Available from: https://ethereum.org/en/upgrades/merge/#misconceptions. pages 44

[81] Mulligan C. *Proposed Architecture Review*;. [Personal interview, 27th August 2022], London (Unpublished). pages 47

[82] Anson J. *Proposed Architecture Review*;. [Personal interview, 24th August 2022], London (Unpublished). pages 48