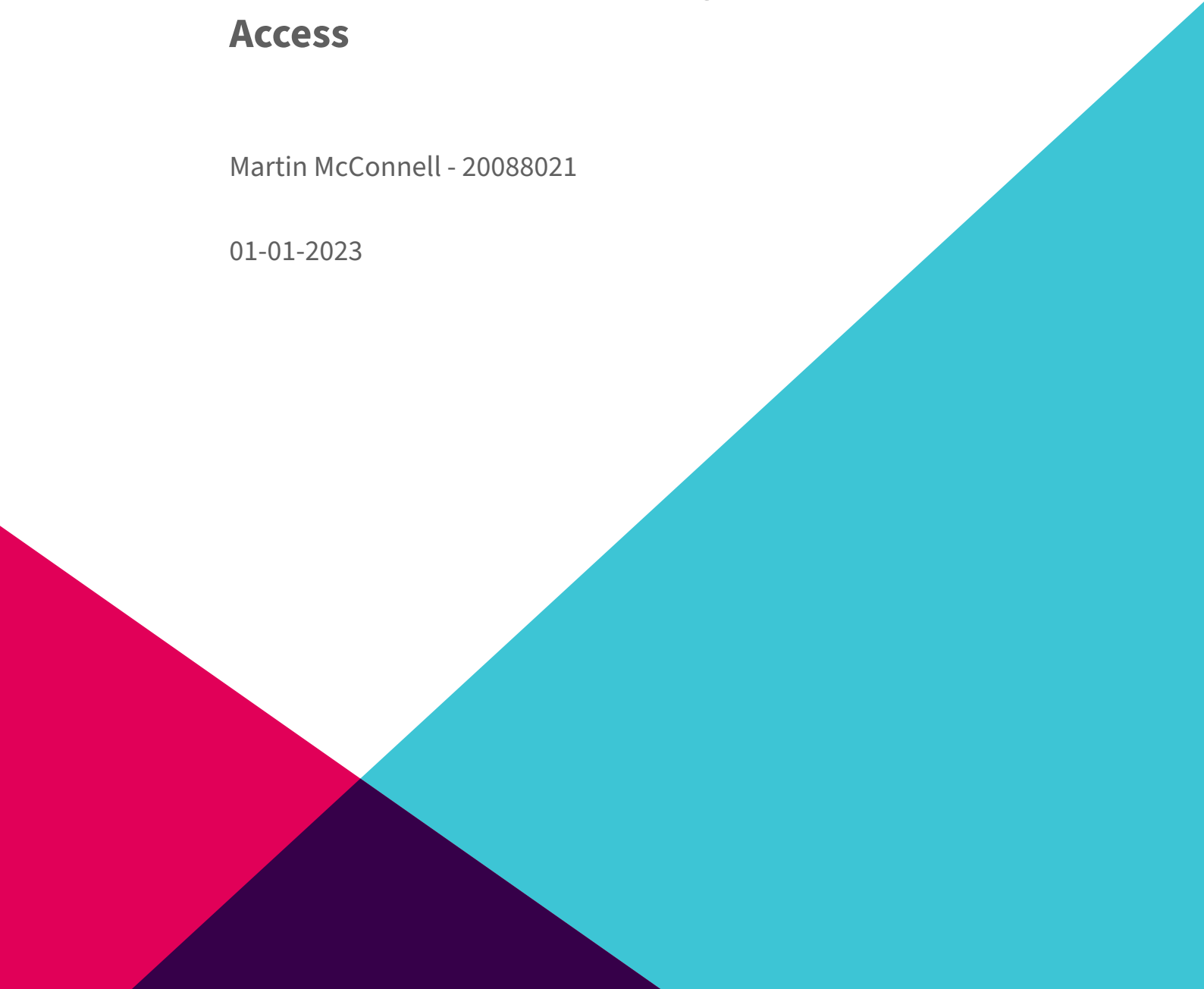# Democratising Knowledge, The Decentralised Opportunity for Trust and Access

Martin McConnell - 20088021

01-01-2023

Contents

Abstract

Introduction

"In economics, a public good is a good that is both non-excludable and non-rivalrous. For such goods, users cannot be barred from accessing or using them for failing to pay for them. Also, use by one person neither prevents access of other people nor does it reduce availability to others." **(Oakland, 1987)**

Non-excludability means it is made impossible to exclude any individual from consuming the good. It is possible to create excludability by means of pay-walls and membership only access.

Non-rivalrous, is the accessibilty of a product or good that in the consumption does not affect the availability for subsequent use, In this manner a digital good can be classified as such.

Purpose, Intended Use and Audience

Before the World Wide Web , distribution of academic articles to a global audience was extremely difficult. t required proof-reading, typesetting, printing and finally distribution. However since the ubiquity of the Internet the majority of tasks performed by publishers has shrunk enormously. In fact publishers now expect researchers to submit digital copies of their work that require no further type-setting or processing and because of digital distribution, printing has become unnecessary. Copying is now simple and free and worldwide distribution is instantaneous and online. **(Taylor, 2012)**

According to **Chow & Birdwell, 2022**, there is an increased distrust in scientific research in many fields of study. The main purpose of the proposed system is to remove the corporate and empirical structure of the current journal publication where in many cases corporations have been found to create and promote articles with bias towards certain priorities that suit the business opportunities of the corporation and not the actual scientific consensus.

For example in Coca-Cola and Mars sponsored research, publications appear to skew the evidence towards solutions that favour industry interests by focusing on food components that can be manipulated and marketed by food companies. Shaping the debate around scientific methods can be another strategy that corporations use for their benefit to raise doubts about the methods used in non-industry sponsored research **(Fabbri et al., 2018)**..

It has become necessary to provide an alternative to the profit driven publications which provide no tangible system to counteract manipulation by corporations.

"The European Universities Association (EUA) found that overall expenditure by 26 European countries was €597 million (£515 million) in 2017. But 75% of that – some €451 million – was spent on subscriptions

to journals published by the 'big five': Elsevier, Springer Nature, Wiley, Taylor & Francis and the American Chemical Society (ACS)" **(Mehta, 2019)**.

With all this in mind it is clear that there is a necessity for Academic Journal Publishing Reform. For better access, trust and incentive to contribute to the archive of journals for everyone's benefit.

Originally coined by Nick Szabo, "Smart Contracts" are electronic agreements that are immutable and transparent, and deployed on a decentralised blockchain. Meaning they cannot be altered, automatically execute everyone can analyse them. the terms of the agreement. By utilising the trust that smart contracts provide it will enable researchers to contribute, publish and peer-review articles removing the ability for manipulation and bias towards results, by moving the actions of reviewers and researchers to a proof-of-work model on a blockchain. Benefits of this is there will be no need for a subscription/fee based model for accessing research as the researchers themselves will own the rights to their own work by staking them on the blockchain and also researchers and Academics will be able to earn passive income off the favourable and positive contributions to the emerging consensus.

Currently, the peer-review process is controlled by publishers. If this process were to be decentralised, it would be fascinating to see the outcome.

The peer-reviewing process1

Peer review generally works like this;

The researcher writes a paper and submits it to a journal.

The editor who is put in charge of the paper selects a number of other researchers to offer the paper to. There is some say as to who the reviewers are (e.g. one can explicitly ask the editor not to have certain other authors review the work, and can also explicitly ask for certain reviewers), and in some cases the journal will explicitly ask for a list of potential reviewers, but final authority comes down to the editor.

The reviewers read and critique the paper, send their recommendations back to the editor, who then returns the information back to the authors along with their decision as to publication (typically, the editors request specific revisions prior to publication).

The researcher makes revisions, and returns the paper to the editor, who then will make the final call as to publication (typically, this is dependent on the addressing all of the various concerns of the reviewers).

This obviously has a lot of room for problems, and one has to trust the journals to do their due diligence and ensure the process works out correctly. In this manner the reputation of the journal is the key to judging the works published in said journal. Furthermore, many journals group themselves into one of

the bigger publishing houses (like Elseveir, Springer, Taylor & Francis, etc.), and those publishers like to keep their journals reputable.

Granted, many journals have issues and often can be a pay-to-publish journal. These journals have what is typically branded as "expedited peer review," which may just be the editor checking for grammatical mistakes. The authors then pay a fee to publish the paper, typically marketed as a fee to publish open-source.

In addition, many "open-source" journals (particularly in engineering fields) are patent and IP trolls (a pejorative term for an entity that enforces patent rights above and beyond the patent's actual value); you publish your work open-source with them, pay a lower fee than the pay-to-publish journals, but in the process you sign over all potential IP from the work, and the paper itself never really gets published in any major journals, but instead gets presented at a conference that you have to pay an exorbitant fee to attend.

Ultimately; it's all built on reputation. By checking the journal's impact factor against other journals in that field (while not comparing across fields). Also the location where the journal is published can be a deciding reputation factor. One can approximate the reliability of the source material.

The other good rule of thumb is the reference list of any given paper, and the journals that those references are published in.

Goals and Requirements

Goals

The aim of this project is to improve the process of publishing academic journals by utilising the internet for distribution and implementing decentralised applications to support the review process. Additionally, aiming to increase trust in the system through the use of "smart contracts" on the Ethereum blockchain.

Requirements

Some potential requirements for a decentralised application (dApp) for a scientific journal project are outlined below:

- The dApp should allow researchers to submit papers to the journal. This should include a mechanism for uploading the paper, along with information such as the title, author, and keywords.
- The dApp should allow reviewers to review submitted papers. This should include a mechanism for reviewing the paper, such as a rating system and a field for feedback.

- The dApp should allow the journal editor to review and approve papers for publication. This should include a mechanism for the editor to view submitted papers and reviews, and to decide which papers should be published in the journal.

- The dApp should allow users to view published papers in the journal. This should include a mechanism for browsing and searching published papers, as well as a way to view the full text of each paper.

- The dApp should use the Ethereum blockchain to store information about submitted papers, reviewers, and published papers. This should include using smart contracts to manage the submission, review, and publication process, as well as using decentralised storage solutions like IPFS to store the actual papers.

These are just some potential requirements for a dApp for a scientific journal project. Depending on the specific goals and requirements of the project as it evolves, there may be other requirements that are important to consider.

Exploratory Analysis

The first half of this section is research and an exploration of tech, leading to the developmental approach to be taken i.e. Agile, TDD etc.

Decentralised

According to **(IPFS, 2022)** Decentralisation is the downloading of a file or files from many locations that are not managed by a single organisation. The fundamental ethos behind decentralisation is the creation of a resilient internet where for instance if a service is under attack on the current centralised internet through a denial of service or ransomware attack the service could be disrupted, the modern internet relies on services like Amazon Web Services(AWS) to perform quick rerouting and load-balancing in such eventualities but again this is reliant on a single entity.

The driving force behind decentralised systems is to create a fast, more secure web **(Nnakwue, 2021)**. Typically a decentralised app relies on a distributed computing model where system components run using a peer-to-peer network. Where all files can be replicated or synced amongst other peers residing on the same network **(Nnakwue, 2021)**

This property of having caches of content distributed globally allows for a protocol where the content can be addressed from anywhere including remotely with little to no internet access and from a location geographically closer to the device retrieving said content.

There have been many protocols proposed to achieve these fundamental goals.

peer 2 peer, P2P

The concept of peer-to-peer technology dates back to the early days of networking. It was initially developed as a way to allow computers to connect and share resources without the need for a central server or authority.

In the early 1990s, the first peer-to-peer file sharing networks began to emerge, allowing users to share and download files directly from each other's computers. This proved to be a popular and efficient way to share files, but it also led to the widespread sharing of copyrighted material, which sparked controversy and legal battles **(Woolf, n.d.)**.

In recent years, peer-to-peer technology has continued to evolve and is now used in a variety of applications, including social networking, streaming video, and distributed computing. It is also a key component of blockchain technology, which allows decentralised networks to operate without a central authority **(White, 2018)**.

Overall, peer-to-peer technology has a rich history and continues to play a significant role in the development of networking and computing technology.

Interplanetary File System

The Interplanetary File System, or IPFS, is a decentralised, peer-to-peer protocol for sharing and storing files. It was developed by Protocol Labs, a research, development, and deployment laboratory, in 2015.

IPFS is based on the concept of distributed hash tables (DHTs), which allow nodes in a network to store and retrieve data based on a unique identifier known as a "hash." This allows IPFS to distribute and replicate data across multiple nodes, making it more resilient and efficient than traditional centralized systems **(Protocol Labs, 2015)**.

Since its launch, IPFS has gained a significant amount of interest and adoption in the blockchain and decentralised web communities. It has also been used in a variety of applications, including file sharing, content distribution, and distributed computing **(Protocol Labs, 2015)**.

Overall, the development of IPFS has been an important advancement in decentralised, peer-to-peer technology, and it continues to be an active area of research and development.

Content Identifier and Distributed Hash Tables

Content identifiers, or CIDs, are unique identifiers used by the Interplanetary File System (IPFS) to identify and locate data in a decentralised network. They are based on the concept of distributed hash

tables (DHTs), which allow nodes in a network to store and retrieve data based on a unique identifier known as a "hash." **(Protocol Labs, n.d.)**

CIDs are a key component of IPFS, as they allow users to access and share data without the need for a central authority or server. They are also used to ensure that data is stored and accessed in a consistent and reliable manner across the network **(Protocol Labs, n.d.)**.

CIDs are typically represented as a string of characters that begin with "Qm" followed by a series of numbers and letters. They are generated using a cryptographic hash function, which ensures that they are unique and cannot be easily tampered with **(Protocol Labs, n.d.)**.

Overall, CIDs are an essential part of the IPFS protocol and enable its decentralised, peer-to-peer nature.

Kubo (Golang on IPFS)

go-ipfs, now called Kubo, is an implementation of the Interplanetary File System (IPFS) written in the Go programming language. It is a decentralised, peer-to-peer protocol for sharing and storing files, and it is designed to be scalable, efficient, and securei **(Protocol Labs, n.d.)**.

go-ipfs is developed and maintained by Protocol Labs, the creators of IPFS. It is open-source and available on GitHub, allowing anyone to contribute to its development or use it in their own projects.

go-ipfs is a command-line tool, which means that it is typically used in a terminal or command prompt. It provides a range of commands for managing and interacting with the IPFS network, including commands for adding and retrieving files, running a local node, and connecting to the network **(Protocol Labs, n.d.)**.

Overall, go-ipfs is an important component of the IPFS ecosystem, providing a high-quality and well-supported implementation of the protocol in the Go programming language. **(Protocol Labs, n.d.)**

Databases

In a project such as this, a decentralised database solution could be used to store information about submitted papers, reviewers, and published papers. Decentralised databases, also known as distributed ledgers, are databases that are spread across multiple nodes or computers, rather than being stored on a single central server. This makes them highly resilient and secure, as well as transparent and easily auditable.

One potential decentralised database solution that could be used in a project like this is the Ethereum blockchain. The Ethereum blockchain is a decentralised platform that runs smart contracts, which are self-executing contracts with the terms of the agreement between buyer and seller being directly

written into lines of code. This makes it a powerful tool for storing and managing information in a transparent, secure, and decentralised manner.

Additionally, the Ethereum blockchain allows for the use of decentralised storage solutions like Inter-Planetary File System (IPFS), which is a distributed file system that allows for the storage and retrieval of data in a decentralised manner. This could be useful for storing the actual papers that are submitted to the journal, as well as other related information.

Furthermore, the Ethereum blockchain and decentralised storage solutions like IPFS are potential decentralised database solutions that could be used in a project like this. However, there are many other decentralised database solutions available, and the specific solution that is best for a given project will depend on the specific requirements and goals of the project.

OrbitDB

OrbitDB is a distributed, peer-to-peer database built on top of the Interplanetary File System (IPFS). It is designed to be scalable, efficient, and secure, and it uses a key-value data model to store and retrieve data in a decentralised network.

OrbitDB is developed and maintained by the team at 3box, a company that focuses on decentralised identity and storage solutions. It is open-source and available on GitHub, allowing anyone to contribute to its development or use it in their own projects.

OrbitDB has a number of unique features and advantages, including built-in conflict resolution and event logging, which allows for easy synchronization and collaboration. It also has support for various data types, including JSON, string, and binary, making it versatile and easy to use.

Overall, OrbitDB is a powerful and innovative database solution that is well-suited for decentralised and peer-to-peer applications. **(3box, n.d.)**

Blockchain

Blockchain is a distributed, decentralised, digital ledger that is used to record and verify transactions in a secure and transparent manner that uses a chain of cryptographic hashes to store and verify transactions. It is the underlying technology behind cryptocurrencies like Bitcoin and Ethereum, and it has a number of important characteristics and features.. It is typically implemented as a chain of blocks, where each block contains a number of transactions and a cryptographic "hash" that links it to the previous block in the chain. Each of these transactions are verified and validated by a network of nodes, which use consensus algorithms to ensure that the data in the blockchain is accurate and consistent.

The fundamental property of a blockchain is its immutability, which means that once a block has been added to the chain, it cannot be altered or removed. This is achieved through the use of cryptographic techniques, such as digital signatures and hash functions, which ensure the integrity and security of the data in the blockchain. **(Nakamoto, 2006)**

Each block in a blockchain is linked to the previous block through the use of a cryptographic hash, which ensures the integrity and security of the data in the chain. This allows for a tamper-evident and immutable ledger, as any attempts to alter or manipulate the data in a block would be easily detected.

Another key feature of blockchain technology is its distributed nature, which means that it is not controlled by any central authority or intermediary. Instead, it relies on a network of nodes, or participants, who maintain and validate the blockchain. This decentralised model allows for greater transparency, security, and resilience, as it is not dependent on any single entity **(Nakamoto, 2006)** .

Blockchain also uses public-key cryptography to secure transactions and protect user privacy. Each user in a blockchain network has a unique pair of keys, a public key and a private key, which are used to sign and verify transactions. This ensures that only the user with the corresponding private key can access and control their data in the blockchain.

Overall, blockchain is a powerful and innovative technology that has the potential to revolutionize a wide range of industries and applications. It is already being used in a variety of contexts, including finance, supply chain management, and digital identity.

Consensus algorithms

A consensus algorithm is a mathematical protocol that is used by nodes in a distributed network to reach agreement on the contents of a blockchain. It is an essential part of the blockchain, as it ensures that the data in the blockchain is accurate and consistent across all nodes in the network.

Consensus algorithms can take many forms, but they all share the same goal of allowing nodes in the network to reach agreement on the state of the blockchain in a decentralised and trustless manner.

One common example of a consensus algorithm is proof-of-work (PoW), which is used by the Bitcoin network. In a PoW-based blockchain, nodes compete to solve a mathematical puzzle by hashing a block of transactions and trying to find a solution that meets a certain criteria. The first node to find a valid solution is allowed to add the block to the blockchain and is rewarded with a certain number of tokens.

Another example of a consensus algorithm is proof-of-stake (PoS), which is used by the Cosmos and recently the Ethereum network. In a PoS-based blockchain, nodes are chosen to add blocks to the

blockchain based on their stake, or the amount of tokens they hold in the network. The higher the stake, the higher the probability that a node will be chosen to add a block.

Overall, consensus algorithms are a crucial part of the blockchain, as they allow nodes in the network to reach agreement on the state of the blockchain and ensure the integrity and security of the data.

The mathematics behind consensus algorithms varies depending on the specific algorithm being used. However, they all share the same goal of allowing nodes in a distributed network to reach agreement on the state of a blockchain in a decentralised and trustless manner **(Nakamoto, 2006)**.

The mathematical puzzle that is solved by the nodes in a PoW-based blockchain is typically a computational problem that is difficult to solve but easy to verify. For example, the Bitcoin network uses a problem called the "double SHA-256" hash, which requires nodes to find a number that, when hashed twice with the SHA-256 algorithm, produces a result that is less than a certain target value.

The mathematical calculation behind the selection of nodes in a PoS-based blockchain is typically a random number generation algorithm, which is used to determine the probability of a node being chosen to add a block based on its stake. This ensures that the selection process is fair and unbiased, and that nodes with a higher stake have a higher probability of being chosen.

The mathematics behind consensus algorithms is a crucial part of the blockchain, as it allows nodes in the network to reach agreement on the state of the blockchain and ensure the integrity and security of the data.

Smart Contracts and Solidity

Smart contracts are self-executing contracts with the terms of the agreement between buyer and seller being directly written into lines of code. They are a key component of the Ethereum blockchain and are used to facilitate, verify, and enforce the negotiation or performance of a contract.

Smart contracts are typically implemented using a high-level programming language called Solidity, which is designed specifically for writing smart contracts on the Ethereum platform. Solidity is a statically-typed, object-oriented language that is influenced by JavaScript and C++.

To write a smart contract in Solidity, the source code of the contract is compiled into bytecode, which is the binary representation of the contract that can be executed on the Ethereum Virtual Machine (EVM).

Once a smart contract is deployed on the Ethereum blockchain, it is stored and executed by nodes in the network. The contract can be called and interacted with by other contracts or by users, and its state and data can be queried and updated **(Buterin, 2014)**.

Solidity has a number of important features and characteristics that make it well-suited for writing

smart contracts. It has support for various data types, including integers, booleans, and arrays, and it also has support for inheritance and contract abstractions.

Smart contracts and Solidity are powerful tools for implementing and executing self-executing contracts on the Ethereum blockchain. They enable the creation of complex and decentralised applications that can automate a wide range of business processes and interactions.

Methodology

It is possible to create a scientific journal publication on the Ethereum blockchain using smart contracts in peer review. This would involve implementing a smart contract that allows researchers to submit their papers to the journal, and a peer review process that is managed and enforced by the contract.

To create a scientific journal on the Ethereum blockchain, the following steps could be followed:

- Develop a smart contract that defines the rules and requirements for submitting papers to the journal, such as the format and content of the papers, and the fees for submission and publication.

- Implement a peer review process in the contract, which allows reviewers to evaluate the papers and provide feedback and ratings. This could be done using a voting system, where reviewers cast their votes on the quality and relevance of the papers.

- Integrate the contract with a decentralised storage solution, such as IPFS (Interplanetary File System), to store and distribute the papers in a decentralised and secure manner.

- Provide a user-friendly interface, such as a web application or mobile app, that allows researchers to submit their papers to the journal and track the progress of the peer review process.

- Promote the journal and its benefits to the scientific community, and encourage researchers to submit their papers and participate in the peer review process.

Creating a scientific journal on the Ethereum blockchain using smart contracts in peer review would provide a decentralised, secure, and transparent platform for publishing and reviewing scientific papers. It could help to improve the efficiency and quality of the peer review process, and provide new opportunities for collaboration and dissemination of scientific knowledge.

Development cycle

Use Case Diagram

[Insert Diagram Here]

In this diagram, the main actors are the Researchers, who submit their papers to the journal, and the Reviewers, who evaluate the papers and provide feedback. The Ethereum blockchain and the smart contract are the main system components, and they are used to manage and enforce the submission and review process.

The main use cases in this diagram are:

- Submit paper: The researcher submits a paper to the journal, and the contract validates the submission and stores the paper in a decentralised storage solution.

- Review paper: The reviewer evaluates the paper and provides feedback and ratings, and the contract records the review and updates the status of the paper.

- Publish paper: After the review process is complete, the contract determines if the paper meets the requirements for publication and adds it to the journal.

The "Decentralised Storage" is an optional component that can be used to store and distribute the papers in a decentralised and secure manner. The "Interface" is another optional component that provides a user-friendly interface for researchers and reviewers to interact with the journal and track the progress of the peer review process.

The main use cases in the diagram are "Submit Paper", "Review Paper", and "Publish Paper". These represent the primary actions that researchers and reviewers can perform in the journal, and they are supported by the "Smart Contract" and other components of the system.

The use case diagram provides a high-level overview of the functionality and interactions in a scientific journal on the Ethereum blockchain using smart contracts in peer review. It can help developers and stakeholders understand the requirements and design of the system, and guide the development and implementation of the journal.

This use case diagram illustrates the main actors, components, and use cases involved in creating a scientific journal on the Ethereum blockchain using smart contracts in peer review.

An Agile Approach with Kanban

An agile approach with kanban can be considered for a scientific journal on the Ethereum blockchain using smart contracts in peer review, as it can help to improve the project's flexibility, transparency, and collaboration.

To use an agile approach with kanban for the journal project, the following steps could be followed:

- Define the project's goals and objectives, and identify the key features and requirements for the journal's contracts and user interface. This will provide a clear vision and direction for the project, and help to prioritize and plan the work.

- Create a kanban board to visualize and manage the project's workflow and tasks. The board can be divided into columns, such as "To Do", "In Progress", and "Done", to represent the different stages of the development process.

- Use agile principles and practices, such as iterative development, frequent feedback, and collaboration, to guide the project's development and deployment. This will allow the team to be flexible and adapt to changing requirements and priorities, and to deliver value to users quickly and iteratively.

- Use kanban tools and techniques, such as limit WIP (work in progress) and pull-based scheduling, to optimize the project's workflow and reduce waste and delays. This will help to keep the project on track and deliver high-quality results in a timely manner.

- Monitor and improve the project's progress and performance using agile metrics and techniques, such as velocity and burn-down charts. This will provide valuable insights into the project's progress and identify areas for improvement.

Using an agile approach with kanban for the scientific journal project can help to improve the project's flexibility, transparency, and collaboration, and provide a more effective and efficient platform for publishing and reviewing scientific papers.

Test-Driven-Development

Test-driven development (TDD) is a software development approach in which tests are written for a piece of code before the code itself is written. This allows developers to verify that their code is working as intended and identify any issues or errors early on in the development process.

If using a test-driven development approach for your project, there are a few key things to consider. These might include:

- Identifying the key functionalities and requirements of the project, and writing tests for each of these functionalities to ensure that they are working as intended.

- Writing the tests first, before writing the actual code. This will help you to focus on the desired behavior and functionality of the code, and ensure that the code is written in a way that satisfies the tests.

- Running the tests frequently during the development process, to ensure that the code is working as intended and identify any issues or errors as early as possible.

- Refactoring the code as needed, based on the results of the tests and any changes to the requirements or functionality of the project.

Continuous Integration/Development

Continuous integration (CI) and continuous development (CD) are software development practices that involve regularly integrating and testing code changes, and continuously deploying and delivering software updates to users. These practices can be useful for a scientific journal on the Ethereum blockchain using smart contracts in peer review, as they can help to improve the quality, reliability, and speed of the journal's development and deployment.

To use continuous integration and development for the journal project, the following steps could be followed:

- Set up a version control system, such as Git, to manage the source code for the journal and its contracts. This will allow developers to collaborate on the code, track changes, and revert to previous versions if needed.

- Configure a continuous integration server, such as Jenkins or CircleCI, to automatically build and test the journal's contracts whenever changes are pushed to the version control system. This will ensure that the contracts are always up-to-date and free of bugs and issues.

- Set up a continuous deployment pipeline, using tools such as the Hardhat CLI or Truffle, to automatically deploy and update the journal's contracts on the Ethereum network whenever changes are made and tested. This will ensure that the contracts are always available and accessible to researchers and reviewers.

- Monitor and optimize the continuous integration and deployment process using tools such as Grafana or New Relic. This will allow developers to track the performance and reliability of the journal's contracts and make improvements as needed.

Utilising continuous integration and development practices for the scientific journal project can help to improve the quality and speed of the journal's development and deployment, and provide a more reliable and user-friendly platform for researchers and reviewers.

Technologies

Ethers.js

Ethers.js is a JavaScript library for interacting with the Ethereum blockchain and decentralised applications (dApps). It is designed to be easy to use and provides a range of tools and features for developers, including support for signing and sending transactions, querying the blockchain, and interacting with smart contracts.

Ethers.js is developed and maintained by Ethers.io, a company that provides tools and services for the Ethereum ecosystem. It is open-source and available on GitHub, allowing anyone to contribute to its development or use it in their own projects.

To use Ethers.js, developers typically include the library in their project and use its APIs to interact with the Ethereum blockchain and dApps. For example, they can use the library to create and sign transactions, query the blockchain for information, or interact with smart contracts.

Ethers.js is designed to be lightweight and modular, allowing developers to include only the features and functionality they need in their projects. It also has support for multiple Ethereum networks, including the mainnet, testnets, and private networks, making it versatile and easy to use in a variety of contexts.

Ethers.js is a powerful and well-designed library for interacting with the Ethereum blockchain and dApps, and it is a popular choice among Ethereum developers.

Metamask

Metamask is a browser extension that allows users to interact with the Ethereum blockchain and decentralised applications (dApps). It is available for popular web browsers, including Google Chrome, Mozilla Firefox, and Microsoft Edge.

Metamask provides a user-friendly interface for managing and using Ethereum accounts, signing and sending transactions, and interacting with dApps. It also includes a built-in wallet for storing and managing Ether and other Ethereum-based tokens.

To use Metamask, users first need to install the extension in their web browser and create a new account. They can then use the extension to manage their accounts, sign transactions, and interact with dApps.

Metamask also allows users to connect to multiple Ethereum networks, including the mainnet, testnets, and private networks. This allows them to easily switch between networks and test their dApps in different environments.

Metamask is a convenient and user-friendly tool for interacting with the Ethereum blockchain and dApps, and it is widely used by Ethereum users and developers.

Hardhat

To use the Hardhat development environment in creating a scientific journal on the Ethereum blockchain using smart contracts in peer review, the following steps could be followed:

Install the Hardhat development environment on your computer by following the instructions on the Hardhat website (https://hardhat.org/). This will include installing the Hardhat CLI and other dependencies, such as Node.js and the Solidity compiler.

Create a new project directory for the journal and initialize the Hardhat environment by running the "hardhat init" command in the terminal. This will create a default configuration file and directory structure for the project.

Create a new Solidity contract for the journal by using a text editor or Solidity IDE to write the source code. The contract should define the rules and requirements for submitting and reviewing papers, and implement the peer review process using smart contract functions and data structures.

Compile the contract using the Solidity compiler, which is included in the Hardhat environment. This will produce the bytecode and ABI (Application Binary Interface) for the contract, which are needed for deploying and interacting with the contract on the Ethereum blockchain.

Deploy the contract to an Ethereum network, such as the Rinkeby testnet, by using the Hardhat CLI or other tools, such as the Truffle framework. This will deploy the contract to the network and make it available for interaction by other contracts and users.

Test and debug the contract by using the Hardhat environment and other tools, such as the Remix IDE and Truffle console. This will allow you to verify that the contract functions as expected and fix any issues or bugs that may arise.

Writing the contract

In general, it's always a good idea to carefully review and test code before deploying a smart contract to the blockchain. This can help ensure that your contract is working as intended and does not contain any errors or vulnerabilities. There are a number of tools and resources available that can help with this process, such as online compilers, linters, and debugging tools.

It's also a good idea to make use of best practices when writing and deploying smart contracts. This can help reduce the risk of errors and vulnerabilities, and improve the overall security and reliability of the contract. Some best practices to consider include:

- Properly handling errors and exceptions in your code to prevent unexpected behavior

- Implementing access controls to limit who can interact with your contract and what they can do

- Using safe math functions to prevent overflow or underflow errors

- Thoroughly testing your contract with a variety of different inputs and scenarios

Solidity

Defining the intial structs for the Scientific Journal Contract and defining which compiler to use.

```solidity
pragma solidity >=0.7.0 <0.9.0;

import "https://github.com/ipfs/js-ipfs/blob/master/src/index.js";

contract ScientificJournal {
    // Define a struct for papers submitted to the journal
    struct Paper {
        string title;
        string abstract;
        string[] keywords;
        bytes file;
        address owner;
    }

    // Define a mapping to store papers submitted to the journal
    mapping(string => Paper) public papers;

    // Define a struct for reviewers of the journal
    struct Reviewer {
        string name;
        bool isActive;
    }

    // Define a mapping to store reviewers of the journal
    mapping(address => Reviewer) public reviewers;

    // Define a struct for reviews of papers in the journal
    struct Review {
        uint rating;
        string feedback;
    }

    // Define an array to store reviews of papers in the journal
    Review[] public reviews;
```

```solidity
// Define a storage struct for a journal entry on the Ethereum blockchain
struct JournalEntry {
    string ipfsHash;
    address owner;
}

// Define a mapping from entryId to journal entry
mapping (uint256 => JournalEntry) public journalEntries;
```

The JournalEntry storage struct contains two fields:

- ipfsHash: the IPFS hash for the journal entry

- owner: the Ethereum address of the journal entry's owner

The journalEntries mapping uses the entryId (generated using the IPFS hash) as the key, and maps to a JournalEntry storage struct. This allows you to store and retrieve journal entries on the Ethereum blockchain using the entryId as a unique identifier.

Here is an example of how you could define the encodeJournalEntry and addToIPFS functions:

```solidity
function encodeJournalEntry(string memory title) private pure returns (bytes memory) {
    // Encode the journal entry's title as a byte array
    return abi.encode(title);
}

function addToIPFS(bytes memory journalBytes) private view returns (string memory) {
    // Add the journalBytes to IPFS and return the IPFS hash for the entry
    return ipfs.add(journalBytes);
}
```

The encodeJournalEntry function takes the journal entry's title and encodes it as a byte array using the Solidity abi library's encode function.

The addToIPFS function takes the encoded journal entry as input, and uses the ipfs.add function to store the journal entry on IPFS. This function returns the IPFS hash for the entry.

```solidity
function publishJournal(string memory title, address owner) public {
    // Emit an event to indicate that a paper has been published in the journal
    emit PaperPublished(title, owner);

    // Store the journal entry on IPFS
    // First, encode the journal entry as a byte array
    bytes memory journalBytes = encodeJournalEntry(title);

    // Then, add the journal entry to IPFS and get the IPFS hash for the entry
    string ipfsHash = addToIPFS(journalBytes);

    // Use the IPFS hash as the unique identifier for the journal entry on the Ethereum
    blockchain
    uint256 entryId = keccak256(abi.encodePacked(ipfsHash));

    // Store the journal entry on the Ethereum blockchain using the entryId as the key
    JournalEntry storage journalEntry = journalEntries[entryId];
    journalEntry.ipfsHash = ipfsHash;
    journalEntry.owner = owner;
}
```

In this example, the encodeJournalEntry function takes the journal entry's title and encodes it as a byte array. The addToIPFS function takes the encoded journal entry and stores it on IPFS, returning the IPFS hash for the entry.

The IPFS hash is then used as the unique identifier for the journal entry on the Ethereum blockchain. The keccak256 function is used to generate a unique entryId for the journal entry using the IPFS hash.

Finally, the journal entry is stored on the Ethereum blockchain using the entryId as the key, and the ipfsHash and owner are stored as part of the JournalEntry storage struct.

To implement an ipfs.add function in the smart contract, a library or contract will need to be used that provides access to the IPFS API. There are several options available for doing this, and the specific implementation will depend on the specific requirements and preferences.

Here are a few examples of how one could implement an ipfs.add function in the contract:

- If using the Ethereum Virtual Machine (EVM) to run the contract, one can use the ethereum-ipfs package to interact with IPFS from the contract. This package provides an IPFS contract that can be deployed to the Ethereum blockchain, and which provides an add function that can be called from the contract to add entries to IPFS.

- If using the web3.js library to interact with the Ethereum blockchain, you can use the ipfs-api package to interact with IPFS from your contract. This package provides an IPFS object that you can use to call the IPFS API, including the add function to add entries to IPFS.

- If the Ethereum Name Service (ENS) is used to resolve IPFS hashes to human-readable names, you can use the ens-js library to resolve IPFS hashes to ENS names and then use the ipfs.add function provided by the ens-js library to add entries to IPFS.

Results

Discussion

Overall, the code looks well-structured and properly organized. There are a few potential areas for improvement, however. For example:

- The contract includes a function called addToIPFS that adds journal entries to IPFS, but this function is not implemented. It just returns a placeholder string and does not actually add anything to IPFS.

- The contract includes a struct called Reviewer that has a name property, but this property is never used. It might be more appropriate to use the address of the reviewer instead of their name.

- The contract includes a function called publishJournal that is used to publish a paper in the journal, but it does not actually check whether the paper has been reviewed or accepted before

publishing it. It might be a good idea to include some additional logic to ensure that only approved papers are published in the journal.

These are just some potential areas for improvement, and there may be other issues or improvements that could be made to the contract.

In terms of some additional requirements to the project from this point in it's progress a few key requirements that are commonly important for dApps in this domain that have not yet been considered. These might include:

- Ensuring that the dApp is user-friendly and intuitive to use, so that researchers and reviewers can easily submit and review papers.

- Ensuring that the dApp is secure and reliable, so that sensitive information about submitted papers and reviewers is protected and the submission, review, and publication process is trustworthy.

- Ensuring that the dApp is scalable and able to handle a large number of submissions and reviews, as well as a growing user base.

- Ensuring that the dApp integrates seamlessly with the Ethereum blockchain and decentralised storage solutions like IPFS, so that information is properly stored and accessed on the blockchain.

- Ensuring that the dApp follows best practices for smart contract development, such as properly handling errors and exceptions, implementing access controls, and using safe math functions.

These are just some potential requirements that may be important to consider when developing a dApp for a scientific journal project. Depending on the specific goals and needs of your project, there may be other requirements that are important to consider..

Conclusion

The idea of using decentralised applications (dApps) and the Ethereum blockchain to improve the process of publishing academic journals is an interesting and potentially valuable idea. By using the internet for distribution and decentralised applications to support the review process, it may be possible to create a more efficient, transparent, and trustworthy system for publishing academic journals. Additionally, the use of smart contracts on the Ethereum blockchain can help to increase trust in the system and ensure that the process is secure and reliable.

There are many potential directions that this project could take to move forward and build on these ideas. For example, the project could focus on developing and implementing a user-friendly dApp that makes it easy for researchers and reviewers to submit and review papers, as well as for journal

editors to manage the publication process. The project could also focus on integrating the dApp with the Ethereum blockchain and decentralised storage solutions like IPFS, to ensure that information is properly stored and accessed on the blockchain. Additionally, the project could explore other potential applications of the Ethereum blockchain and smart contracts in the context of academic publishing, such as using tokens to incentivize good behavior or using decentralised governance mechanisms to manage the journal.

These are just some potential directions that the project could take to move forward and build on the initial premise. Ultimately, the specific direction and focus of the project will depend on the goals and needs of the project team and stakeholders.

Bibliography

- Nakamoto, S. (2006). 'Bitcoin: A Peer-to-Peer Electronic Cash System'. Available at: http://satoshinakamoto.me/bitcoin.pdf (Accessed 18: September 2022)

- Benet, J.. (2014), 'IPFS - Content Addressed, Versioned, P2P File System (DRAFT 3)'. Available at: https://raw.githubusercontent.com/ipfs/papers/master/ipfs-cap2pfs/ipfs-p2p-file-system.pdf (Accessed 19 September 2022)

- Ethereum Foundation. (2013). Ethereum white paper. Retrieved from https://ethereum.org/ethereum.html

- Buterin, V. (2014). A next-generation smart contract and decentralized application platform. Retrieved from https://github.com/ethereum/wiki/wiki/White-Paper

- Paul Eve, M. (2021) WAREZ, The Infrastructure and Aesthetics of Piracy. Earth, Milky Way, Punctum Books.

- infourminutes.co (2018) IPFS Whitepaper in Four Minutes. Available at: https://medium.com/coinmonks/ipfs-whitepaper-in-four-minutes-b3d5eb0e75c6 (Accessed: 19 September 2022)

- LBRY (2019) Available at: https://lbry.com/faq/different-ipfs (Accessed: 21 September 2022)

- Ernesto Van der sar (2019) Decentralized 'Pirate Bay' with IPFS. Available at: https://torrentfreak.com/torrent-paradise-creates-decentralized-pirate-bay-with-ipfs-190120/ (Acessed: 26 September 2022)

- Ignacia Larrain (2022) Will Google Analytics be Banned in Europe? Not as easy as it seems. Available at: https://visionarymarketing.com/en/2022/04/google-analytics-ban/ (Accessed: 26 September 2022)

- Nisha Jain (2022) EU declares Google Analytics illegal: Here's why. Available at: https://techstory.in/eu-declares-google-analytics-illegal-heres-why/ (Accessed: 26 September 2022)

- Bluetooth SIG (2019) Bluetooth for Linux Developers, Available at: https://www.bluetooth.com/bluetooth-resources/bluetooth-for-linux/ (Accessed: 21 October 2022)

- Akin Gump Strauss Hauer & Feld (2022) New Privacy Shield Agreement Announced, Available at: https://www.jdsupra.com/legalnews/new-privacy-shield-agreement-announced-9279044/ (Accessed:07 Novemeber 2022)

- IPFS. (2022) 'what is ipfs?' IPFS Docs. Available at: https://docs.ipfs.tech/concepts/what-is-ipfs/#what-is-ipfs (Accessed: November 22, 2022).

- Andrew et al. (2022) Statistical Modeling, causal inference, and social science, Statistical Modeling Causal Inference and Social Science. Available at: https://statmodeling.stat.columbia.edu/2022/10/30/distrust-in-science/ (Accessed: November 26, 2022).

- Taylor, Mike (21 February 2012). "It's Not Academic: How Publishers Are Squelching Science Communication". Discover. Available at: https://web.archive.org/web/20220521122023/https://www.discovermagazin earth/its-not-academic-how-publishers-are-squelching-science-communication/ (Accessed:28 November 2022)

- Mehta, Angela (2019). "75% of European spending on scientific journals goes to 'big five' publishers". Available at: https://web.archive.org/web/20221030103117/https://www.chemistryworld.com/news/75-of-european-spending-on-scientific-journals-goes-to-big-five-publishers/4010616.article/ (Accessed 28 November 2022)

- Chow, M. and Birdwell, J. (no date) Confidence in research: Researchers in the spotlight. Available at: https://impact.economist.com/projects/confidence-in-research/pdfs/Confidence_in_Research-full_report.pdf (Accessed: November 28, 2022).

- Nnakwue, A. (2021) A guide to working with orbitdb in Node.js, LogRocket Blog. Available at: https://blog.logrocket.com/guide-to-orbitdb-node-js/ (Accessed: November 29, 2022).

- Oakland, W. H. (1987). Theory of public goods. In Handbook of public economics (Vol. 2, pp. 485-535). Elsevier

- Fabbri, A., Holland, T.J. and Bero, L.A. (2018) Food Industry sponsorship of academic research: Investigating commercial bias in the Research Agenda: Public Health Nutrition, Cambridge Core. Cambridge University Press. Available at: https://www.cambridge.org/core/journals/public-health-nutrition/article/food-industry-sponsorship-of-academic-research-investigating-commercial-bias-in-the-research-agenda/A4D9C0DC429218D5EFDFBE80FAE5E087 (Accessed: November 29, 2022).

- Woolf, M. (n.d.). A brief history of peer-to-peer networks. Retrieved from https://www.makeuseof.com/tag/brief-history-peer-peer-networks/ (Accessed: December 9, 2022)

- Knittel, B. (n.d.). The history of peer-to-peer networks. Retrieved from https://www.techopedia.com/the-history-of-peer-to-peer-networks/ (Accessed: December 9, 2022)

- White, E. (2018, March 15). The evolution of peer-to-peer networking. Retrieved from https://www.forbes.com/sites/elizabethwhite/2018/03/15/the-evolution-of-peer-to-peer-networking/?sh=1c7d0e5b2f40 (Accessed: December 9, 2022)

- Protocol Labs. (2015). Interplanetary File System (IPFS). Retrieved from https://ipfs.io/ (Accessed: December 11, 2022)

- Protocol Labs. (2015). Interplanetary File System (IPFS). Retrieved from https://ipfs.io/ (section on content identifiers)(Accessed: December 11, 2022)

- Protocol Labs. (n.d.). go-ipfs. Retrieved from https://github.com/ipfs/go-ipfs (Accessed: December 11, 2022)

- 3box. (n.d.). OrbitDB. Retrieved from https://github.com/orbitdb/orbit-db (Accessed: December 11, 2022)

- Ethers.io. (n.d.). Ethers.js. Retrieved from https://github.com/ethers-io/ethers.js (Accessed: December 12, 2022)