Islamic University of Technology (IUT)

Department of Computer Science and Engineering (CSE)

Blockchain-based Decentralized Source Code Repository Hosting Service with Middleware Approach

Authors

Md. Tahmid Islam, 170042038

&

Sakibul Islam Munna, 170042060

&

MD. Rafid Haque, 170042072

Supervisor

A.B.M. Ashikur Rahman

Asst. Professor, Department of CSE

A thesis submitted to the Department of CSE

in partial fulfillment of the requirements for the degree of B.Sc.

Academic Year: 2020-2021

April, 2022

Declaration of Authorship

This is to certify that the work presented in this thesis is the outcome of the analysis and experiments carried out by MD. Rafid Haque, Sakibul Islam Munna and Md. Tahmid Islam under the supervision of A.B.M. Ashikur Rahman, Asst. Professor, Department of CSE, Islamic University of Technology (IUT). It is also declared that neither of this thesis nor any part of this thesis has been submitted anywhere else for any degree or diploma. Information derived from the published and unpublished work of others has been acknowledged in the text and a list of references is given.

Authors:

[illegible]

Md. Tahmid Islam

Student ID - 170042038

[illegible]

Sakibul Islam Munna

Student ID - 170042060

[illegible]

MD. Rafid Haque

Student ID - 170042072

Supervisor:

[illegible] 16.05.2022

A.B.M. Ashikur Rahman

Asst. Professor

Department of Computer Science and Engineering

Islamic University of Technology (IUT)

Acknowledgement

We would like to express our grateful appreciation for **A.B.M. Ashikur Rahman**, Asst. Professor, Department of Computer Science & Engineering, IUT for being our adviser and mentor. His motivation, suggestions and insights for this research have been invaluable. Without his support and proper guidance this research would never have been possible. His valuable opinion, time and input provided throughout the thesis work, from first phase of thesis topics introduction, subject selection, proposing algorithm, modification till the project implementation and finalization which helped us to do our thesis work in proper way. We are really grateful to him.

We are also grateful to **Mr. Md Hasan Onik**, Lecturer, Department of Computer Science, American International University-Bangladesh, for his valuable inspection and suggestions.

Abstract

Software developers must work together in order to provide a better product. As a result, many software developers use version control systems throughout project development since it helps them manage source code and enables them to keep track of the many versions they've worked on. Items are organized, regulated, and directed by this strategy. In spite of the fact that the version control system is generally decentralized, there is no properly defined practical method for remotely maintaining the code that is not centralized. For a distributed network of computers, we describe and explain our solution, which leverages a blockchain and smart contracts to authorize, monitor, and perform version control actions on a code repository. Using our strategy, there is no need to have a centralized authority that is trustworthy. The immutability of the code and the ownership information of the code writer are protected via the blockchain. A network of servers (IPFS) also maintains the security of the code repository and its content. In this system, the code is stored on a distributed network of servers, ensuring its availability and integrity, while a decentralized blockchain ensures ownership information and the immutability of the repository by encrypting information from the codebase's remote location with a hash that combines the owner's key and the entire code repository.

Contents

List of Figures

1 Introduction

1.1 Overview

For projects that contain more than a few hundred lines of code or need more than one developer to work on the same project, a Version Control System, also known as Revision Control System or a Source Control System, must be used. As a result, the vast majority of software development initiatives make use of it. Source code and project files may be tracked and restored using a Version Control System (VCS). It employs a repository, a form of database that is used by many other applications, for each file (and the whole project structure). Many projects use version control systems (VCS) and other technologies like issue tracking systems to keep track of their progress. In my role as a software manager, the most difficult task is scaling up the change process to accommodate huge numbers of software developers located in many time zones without losing quality or increasing complexity. Technology has an important role in how new contributors may join a project, how often various development lines are integrated, how code reviews and support for previously published codes are handled, and how new feature development is managed in the majority of cases.

Despite their importance, there is a paucity of knowledge on how these technologies effect development and the trade-offs associated with their use. A number of open and closed-source projects, as well as the Apache Software Foundation, have either suggested or have implemented the shift from centralized version control systems (CVCS) to decentralized version control systems (DVCS) (DVCS). [1] The data connected with a software repository employing Blockchains, on the other hand, will not be able to be stored entirely in these systems.

One of the most exciting new technologies to emerge after Bitcoin's first triumph is blockchain technology. Bitcoin is founded on the currency's foundation and utilizes blockchain technology. [2] Blockchain technology uses the network's consensus mechanism to create a distributed ledger or database that is accessible to all members of the network. As a consequence, there is no longer any need for a third-party validator, creating a completely decentralized system.[3] [4] Mining nodes keep a copy of the Blockchain ledger and are responsible for verifying and validating each transaction that results in a change to the ledger. As a consequence, tamper-proof ledgers are created that are distributed, secure, time-stamped, and shared among all participants. [5] Blockchain technology has been used by a number of industries, including banking, healthcare, supply chain and logistics, and document management and accounting.[6][7][8] As a consequence of its powerful and decentralized design, blockchain technology is being utilized to solve challenges like trust, effectiveness, privacy, and data sharing , amongst other things. As a result of the use of cryptography, this approach removes the need for a third-party transaction authority altogether. "Smart Contracts" are computer programs that may be run on the Blockchain network's nodes. Smart contracts are self-executing codes, which may be used to ensure that set terms and conditions are adhered to.[9][10] Data is stored in blocks rather than digital currencies by blockchain mining nodes as opposed to validators for digital currencies like Bitcoin. Smart contracts are decentralized applications that may be programmed to do various tasks based on inputs sent to their Ethereum addresses.[10] Because it is based on Ethereum's decentralized ledger, Ethereum is a distributed, open source platform with smart contract functionality. Due to Ethereum's open source nature, users are free to create their own own apps. Ether is used to pay for transactions on the Ethereum network by both Bitcoin and Ethereum. Using an Ethereum Address, one may uniquely identify another person on the blockchain (EA). The buzz these days is that blockchain is the next big thing.[3] In contrast, because of the 1MB file size constraint per block on the Bitcoin blockchain, storing large documents is still prohibitively costly. With so much data being stored on blockchains, there is certain to be a huge overhead.[11] [12] Examples of decentralized storage systems designed to satisfy the urgent demand for storing vast quantities of data include IPFS (InterPlanetary File System), Storj, SWARM, and Sia. IPFS is the most widely used and well-established platform because of the nature of this study. The IPFS distributed file system, which is a content-addressable, peer-to-peer, and open source file system, can store and transmit massive amounts of data fast.[13] When it comes to storing vast amounts of data, the blockchain is inefficient. Hashes of documents in the chain have been shown to be more effective than the originals themselves in preventing unauthorized access.[6] A hash is generated and maintained in the smart contract that retrieves this information from IPFS for each document. Every time a document is accessed, its hash value changes.

In a peer-to-peer distributed file system, such as IPFS, data may be stored and exchanged across users. IPFS leverages content addressing to guarantee that each file in a global namespace, which links all computing devices, is uniquely recognized. Similar to a file sharing technique like BitTorrent, IPFS enables users store and transmit files. IPFS is based on a decentralized system of user-operators who individually store a small percentage of the overall data, despite the fact that a centrally maintained server is periodically employed. [14] A more secure way to store and send files has been made possible owing to this approach. The Interplanetary File System is only one of several open-source projects Juan Benet is involved with (IPFS). It was created because of the drawbacks of the client-server architecture and HTTP web service applications. IPFS, a decentralized or secure data distribution network, enables quick throughput, minimum latency, and the ability to transmit data safely. We use a self-certifying namespace, a distributed hash table (DHT), and an incentive-based block exchange to create a more ubiquitous DAG that is more secure (Merkle directed acyclic graph). This hash table is constructed taking into consideration the distributed system's proximity to nodes.[15]

IPFS, in contrast to BitTorrent, aspires to build a single worldwide network of files and data. In practice, this implies that if two peers publish a block of data with the same hash, the peers getting the content from Alice and the peers downloading the content from Bob will trade data with one another. [16] IPFS proposes to replace the protocols that are now used for static site delivery by using gateways that are available over HTTP as a replacement. A public gateway may be used instead of installing an IPFS client on a user's device, if that is their preference. The IPFS GitHub website has a list of these gateways that is updated on a regular basis.

The code repository for a distributed network of computers may be authorized, monitored, and controlled using smart contracts and a public blockchain. Our solution eliminates the requirement for a centralized authority that is trustworthy. The code's immutability and the code author's ownership information are protected by a public blockchain. A distributed network of servers protects the code repository and its content. Protecting the transmitter's identity is a top priority for Broadcast Encryption (BE). We provide a public blockchain-based blockchain-based solution for digital document version control systems. This technique does away with the need for a reliable third-party authenticator altogether. We focus on the most important parts of our methodology, such as the overall system architecture and the most critical interactions between members.

1.2 Problem Statement

The bulk of Source Code Repository Hosting Services are now centralized, which implies that the system is managed by a single company, which is inconvenient. On top of that, as the repositories are in a centralized database in most cases, it implies that a single point of failure is created. This means that the system may fail and the data contained therein may be lost. Additionally, ownership information is not always well-maintained or readily available.

This is addressed by our Blockchain based Source Code Repository Hosting Service. Our solution uses a public blockchain and smart contracts to authorize, monitor, and govern a distributed code repository. Our approach does away with a trusted central authority. A public blockchain protects the code's immutability and the author's ownership information. A network of servers protects the code repository and its content. The sender's identity is safeguarded via Broadcast Encryption. Blockchain-based digital document version management is proposed by us. A third-party authenticator may be avoided by using our method. We focus on the most critical components of our strategy, such as the overall system architecture and the way players interact with the system. New developers will be registered, code contributions will be enabled, other people's work will be retrieved, and the system will be designed and tested as a whole.

1.3 Motivation and Scope of Research

A Blockchain-based Decentralized Source Code Repository Hosting Service has the ability to fundamentally alter the way we store code remotely, work remotely, and communicate. Additionally, it alters how code ownership is controlled.

As a result of a pressing need for a decentralized, reliable, and secure way to share and govern code repositories, this paper presents a blockchain-based solution. An agreement on the state and availability of shared data in a trustless environment may be maintained by participating organizations using a distributed system known as the blockchain. [5]

Data transferred through the chain of custody is safeguarded by cryptographic techniques, which prohibit tampering with records. Indeed, each block in the blockchain is confirmed by all active participants before being added to the chain (i.e. consensus by all participants) (i.e. consensus by all active participants). It may be possible to employ smart contracts for new user registrations and user-provided code repository changes.

In order to automate digital code repositories' version control logic and workflow while allowing for controlled or limited data exchange, we recommend using smart contracts. As an alternative, smart contracts function as decentralized organisers of interactions between parties, as opposed to centralized ones. As a result, the most important contribution of this study is as follows: Using smart contracts, we present a mechanism for regulating the version of a digital code repository. Our solution eliminates the requirement for a trusted third-party authenticator to serve as a go-between. Our blockchain solution's overall system design and architecture emphasize the importance of participant interactions. Document version control may help a wide number of businesses, including healthcare, banking and property registration.

1.4 Thesis Outline

As for the rest of the document, it is structured as follows. Section 2 provides an overview of the relevant literature. The decentralized hosting service for source code repositories is presented in Section 3 of this document. The findings and conclusions of the proposed system are discussed in detail in Section 4. Section 5 is the last section of the report.

2 Background Study/Literature Review

Prior work on blockchain-based version control and controlled data sharing for digital documents as well as topics pertinent to our study will be examined in this portion of the paper.

2.1 Ethereum

In addition to the ether coin, Ethereum is a blockchain platform for decentralized applications (dApps). Smart contracts developed on the Ethereum platform power the network. Smart contracts and the blockchain are critical components of DeFi and other applications. As of January 2022, Ethereum is only second in market valuation to Bitcoin. [17]

Since all cryptocurrencies are built on the blockchain, Ethereum is no exception. Access to all blocks in a chain of linked blocks is available to all blockchain participants. If all network users share the same understanding of the blockchain, which operates like an electronic ledger, distributed agreement on the blockchain's state may be obtained and maintained. The present state of the Ethereum network is determined through a distributed consensus mechanism based on the blockchain.. Whenever a transaction is completed and fresh ether money is coined, or smart contracts for Ethereum decentralized apps (dApps) are performed, new blocks on the Ethereum blockchain are created.

The Ethereum network is secure because of the decentralized structure of the blockchain technology. Any updates to the Ethereum blockchain's global network of computers need distributed consensus (majority agreement). This would need taking over the bulk of the Ethereum platform's processing power, which is very difficult or impossible. [18]

A greater number of apps can run on Ethereum than on other cryptocurrencies like Litecoin or Monero. [**?**] Many different types of apps are available on the Ethereum network for users to build, publish, monetize, and utilize, and all of them may be paid for using ETH or another cryptocurrency.

Ethereum's native coin is Ether (ETH). Ethereum, a blockchain platform that supports a wide range of decentralized applications, may be used to host dApps (decentralized apps) (cryptocurrencies included). More people use the term "ethereum" to describe the blockchain-based technology Ethereum, rather than "ETH." [19]

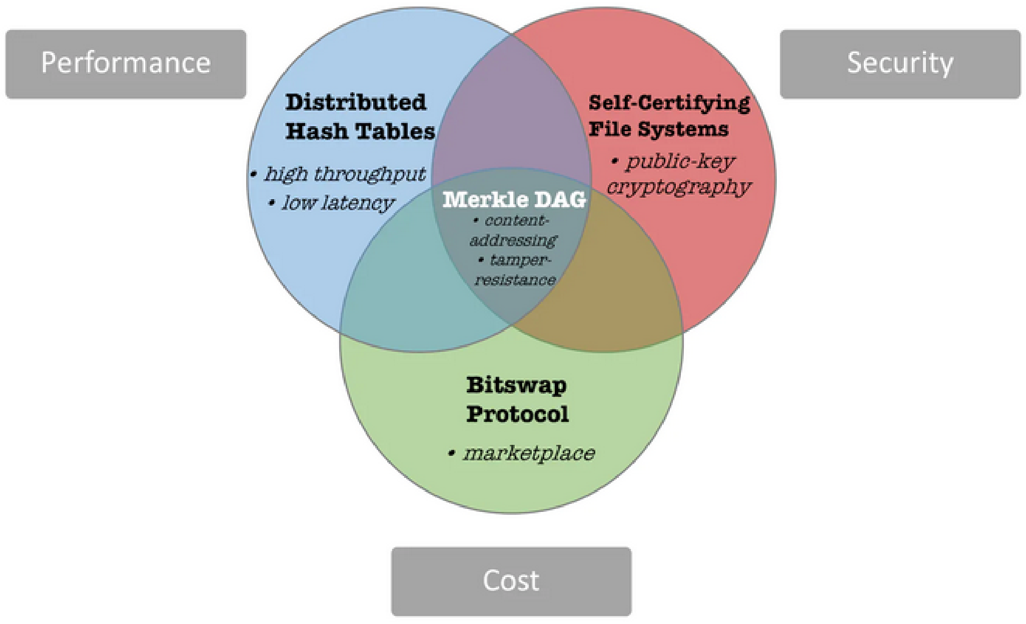
2.2 IPFS

IPFS is a peer-to-peer (p2p) file sharing system with the goal of radically altering how information is delivered throughout and beyond the world. It combines various advancements in communication protocols and distributed systems to create a file system unlike any other. It is a file sharing system that aims to overcome the shortcomings of the client-server approach and the HTTP web. This system is the result of a combination of numerous novel and current technologies. IPFS is an open-source project developed by Protocol Labs, a network protocol research and development lab and former Y Combinator company. [20] [21] Protocol Labs also creates ancillary technologies such as IPLD and Filecoin, which will be discussed in further detail below. Hundreds of developers from all around the globe participated in the creation of IPFS, making its orchestration a mammoth task.

The URL of the server on which the data is kept is used to make a standard HTTP request for the data. On the other hand, data saved on IPFS is accessed by requesting the data's cryptographic hash. In a standard HTTP architecture, data cannot be accessible if the server is unavailable or fails, or if any of the server's connections are lost. Data is replicated over a large number of nodes, which enables it to be retrieved anytime it is required. Due to the fact that several clients are simultaneously requesting information from a single server, the bandwidth available under the typical HTTP paradigm is limited. IPFS, on the other hand, has a high bandwidth need due to the fact that data is requested from the peer with the nearest copy of the required information. To make data publicly accessible using the standard HTTP approach, one must either set up or pay for a hosting server. By contrast, data posted to IPFS does not need the usage of a host server; rather, the data is saved on each node's local storage device. In comparison, HTTP is a well-established industry standard; this is an area where HTTP has an advantage. IPFS is a more recent protocol that is not as widely used as HTTP. It is designed to facilitate file sharing. Apart from that, practically all computers provide HTTP functionality. On the other side, in order for IPFS to operate, one must either use the HTTP to IPFS portal or manually configure an IPFS node on their own computer.

The following are the major components:

Figure 1: IPFS



2.2.1 Hash Tables on a Distributed Scale

In a hash table, information is kept in the form of key/value pairs. Because the data is spread over a number of machines, distributed hash tables can be accessed and searched quickly throughout the network (DHT).

Decentralization, fault tolerance, and scalability are important advantages of DHTs. Without central coordination, DHTs may grow to millions of nodes, and the system can continue to run even if nodes fail or leave the network. Client-server systems are often less resilient when these traits are combined.

2.2.2 Exchanges of Blocks

To successfully distribute data over millions of nodes, Bittorrent uses a revolutionary data exchange protocol, but it is only available in the torrent environment. As a marketplace for all types of data, IPFS offers BitSwap, a generalized version of this protocol. Filecoin, a peer-to-peer storage technology based on IPFS, is built on this marketplace.

2.2.3 DAG Merkle

Merkle Trees and Directed Acyclic Graphs are combined to form Merkle DAGs (DAG). P2P networks are protected by Merkle trees, which guarantee the integrity of data blocks. To arrange data blocks for this verification, cryptographic hash techniques are used. An alphanumeric string (hash) is returned by this basic function, which receives an input and returns an alphanumeric string (hash). It is simple to prove that a given input will create a specific hash, but far more difficult to determine the input from the hash itself.

2.2.4 Self-Certification File System

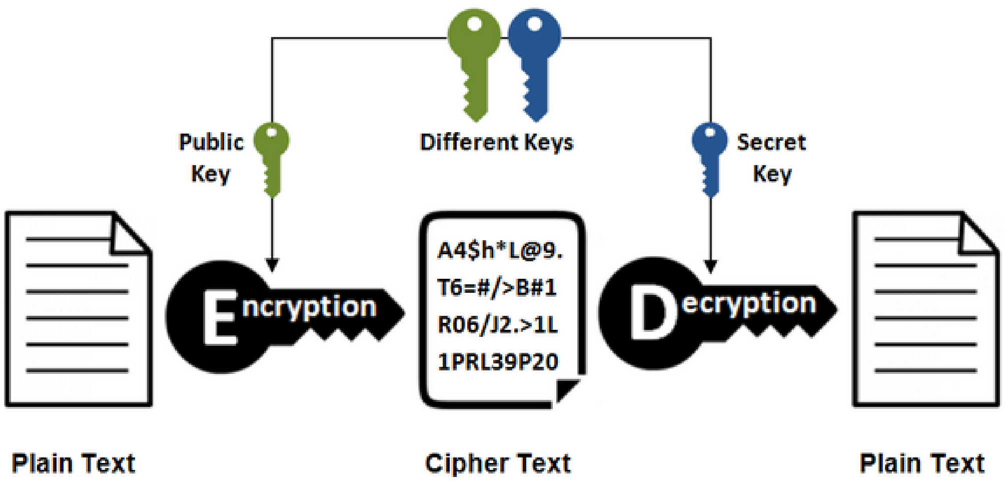
The Self-Certifying File System is the last key IPFS component we'll go through (SFS). It's a distributed file system that doesn't need any specific rights to move data across computers. Self-certification may be applied to filenames since they are utilized for verification purposes when sending data to clients (which is signed by the server). The openness of local storage means that we may view stuff from other locations without fear of security breaches.

It is this idea that serves as the foundation for IPNS (IPNS). An SFS that uses public key cryptography to allow users to verify the authenticity of their own postings is what we call a self-certifying SFS. IPFS nodes may be uniquely recognized as well, as was previously established. Besides the public and private keys, each node in the network has a node ID, which is a hash of the public key. Since private keys are assigned to any data objects that are published, the sender's public key may be used to verify the data's validity. [3][22]

2.3 Asymmetric Encryption

In asymmetric cryptography, two mathematically similar but not identical keys are used — a public key and a private key. This kind of encryption is known as public key cryptography. This method differs from others in that it does not utilize the same key for both encryption and decryption. Encryption relies on the public key, whereas decryption relies on the private key.

Figure 2: Asymmetric Encryption



Calculating the private key from the public key is mathematically impossible. Public keys may be widely circulated, allowing users to encrypt material and verify digital signatures, but private keys can be kept secret, guaranteeing that only the private key owner can decode information and produce digital signatures. This results in a more secure system.

For security reasons, public keys are put on digital certificates for transmission and distribution, since they are too large to memorize alone. For security reasons, private keys are only saved in the software or operating system we are using, or in hardware (such as a USB token or a security module) that offers drivers for use with the software.

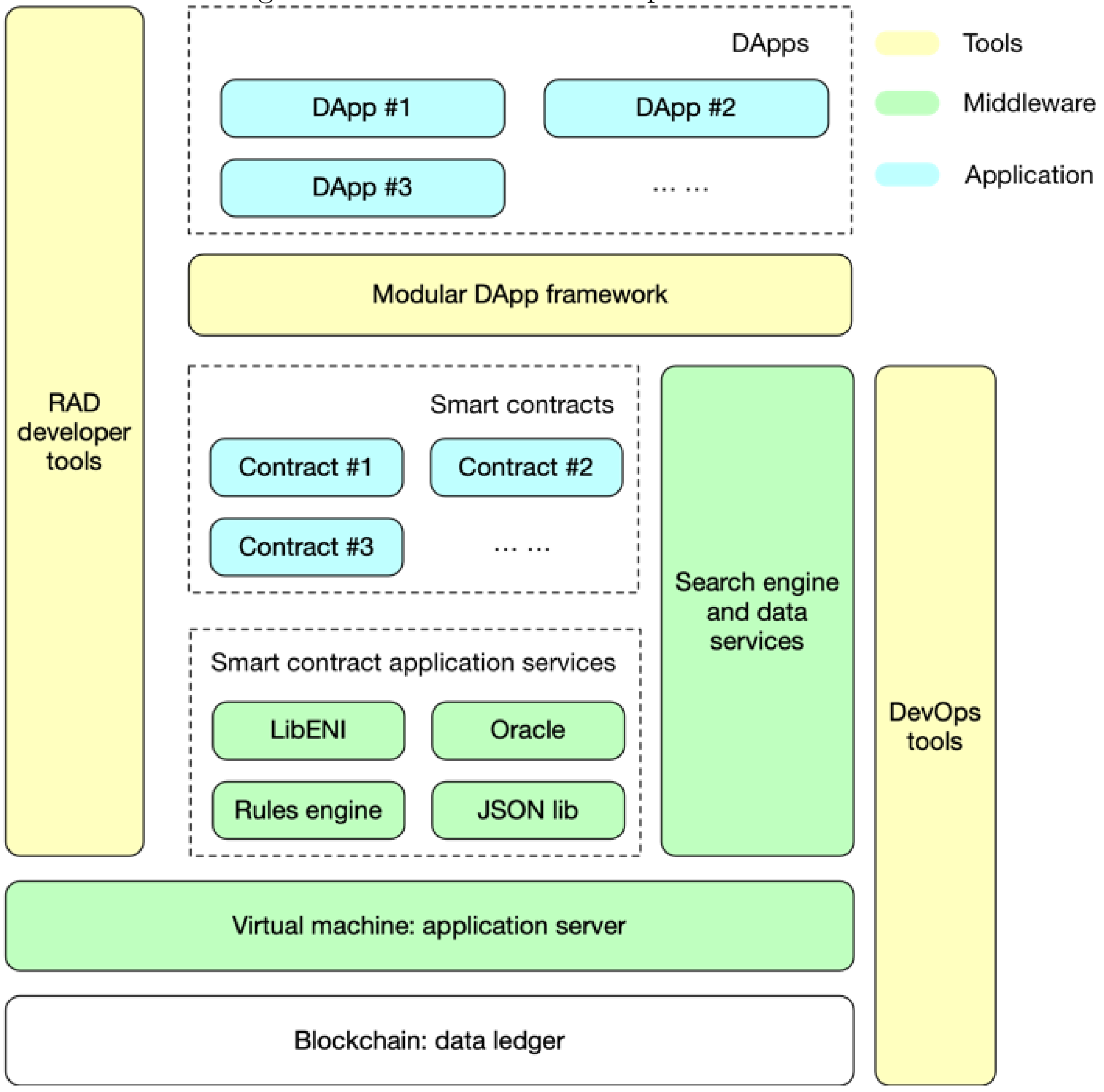
2.4 Blockchain Middleware Approach

It is common to talk about "blockchain middleware" when discussing the many components of a blockchain. In addition, it may include software that combines several blockchain implementations into one unified interface for convenience of use and (typically) as a means of increasing the scalability of the system. A blockchain middleware may also be used to adapt blockchain to use cases that are not well covered by current blockchain solutions. As Maurizio Canton, the CTO of Tibco Software's EMEA area, points out, even while the business case for using blockchain is solid, the implementation might be the difference between reaping all of the advantages of the technology and not. [23] Inherent in the process of implementing this technology is a degree of difficulty. Middleware supporting integration, correlation, and analytics as well as driving security and governance must be a primary concern for any project that is based on the notion of partnership and cooperation.. Canton stresses the need of integrating a wide range of data sources into a business architecture in real time. In order to quickly and easily collect data and improve prediction skills, tools that can visualize blockchain-generated technical and business information are essential. These insights range from fraud detection to user behavior throughout the blockchain.

Business decision-making may be transparent and collaborative with the use of smart contracts on the blockchain. For the implementation of smart contracts in businesses, Second State develops a set of open source infrastructure tools that includes blockchains and virtual machines as well as rules engines, search engines, data analytics, and other development and DevOps tools. [24] When it comes to smart contracts, the blockchain virtual machine is essential. Any participant may write and deploy virtual machine-based smart contracts at any moment without interrupting the blockchain. Smart contracts can automate and ensure the enforcement of the hundreds of business choices that an organization may make every day between various parties.

Second State approaches blockchain infrastructure as an enterprise middleware solution. The virtual machine resembles an application server, and the decentral-

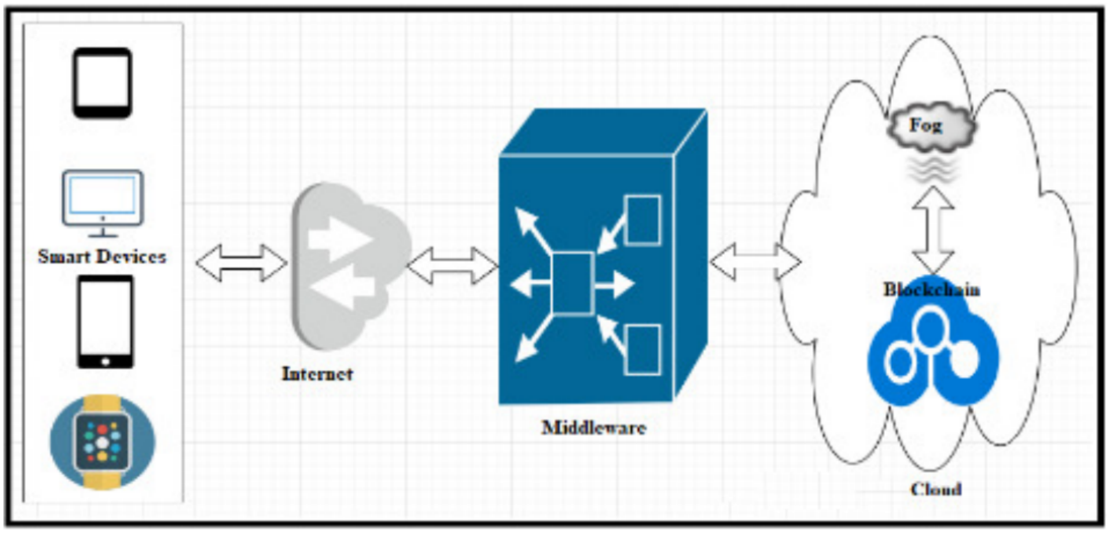
Figure 3: Second State's Enterprise Middleware



ized ledger is like a database. Blockchain data services and smart contracts are comparable to the application services running on the application servers.

This study [25] uses blockchain technology to provide a middleware architecture for the Internet of Things' network of smart devices. Blockchains and the Internet of Things are linked in a new research that gives communication security to the Internet of smart devices. Our key contribution is to connect this new study. It is the goal of this study to develop a new communication paradigm between the Internet of Things and fog computing. Middleware, fog, and the Internet of Things (IoT) are all used in conjunction with blockchain technology in this study. The framework reduces traffic rate vacillation as well as the number of smart devices, which provides Quality of Service (QoS). Using idle state as an example, this study examines the framework's overall performance in terms of the overall efficiency of the system. IoTFog will monitor and analyze real-time data acquired from fog nodes, and then take action.

Figure 4: IoT-Fog Communication Interface using Middleware

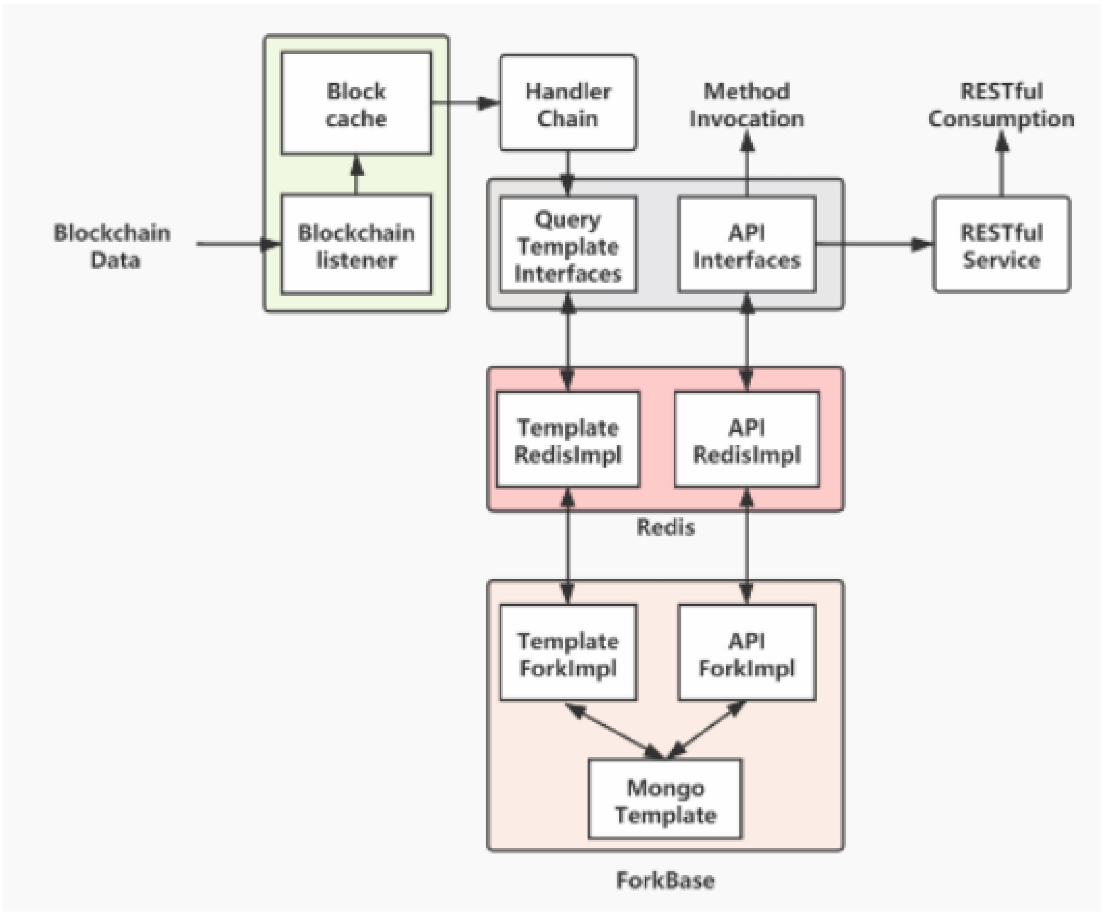


Fog and blockchain are used in this study to provide a communication framework that allows for rapid and reliable communication across the Internet of Things' smart devices. Study in the past has concentrated on developing and optimizing communication frameworks, but this research does not provide a complete framework for IoT-Fog communication among the internet of smart devices.

2.5 Middleware Database

The phrase "blockchain middleware" refers to software functions that are designed to link a number of interconnected blockchain instances and pieces of information. Along with blockchain implementations, it may comprise software that binds multiple blockchain implementations together into a uniform interface for ease of use and (in some situations) scalability. Additionally, blockchain middleware may be used to introduce blockchain to use cases in industries where existing blockchain technology is presently underutilized, such as the pharmaceutical business. According to Maurizio Canton, CTO of Tibco Software for the EMEA region, although the business case for blockchain technology may be appealing, deployment may be a deal breaker if the full benefits and value are to be achieved. [26]

Figure 5: Middlware Database

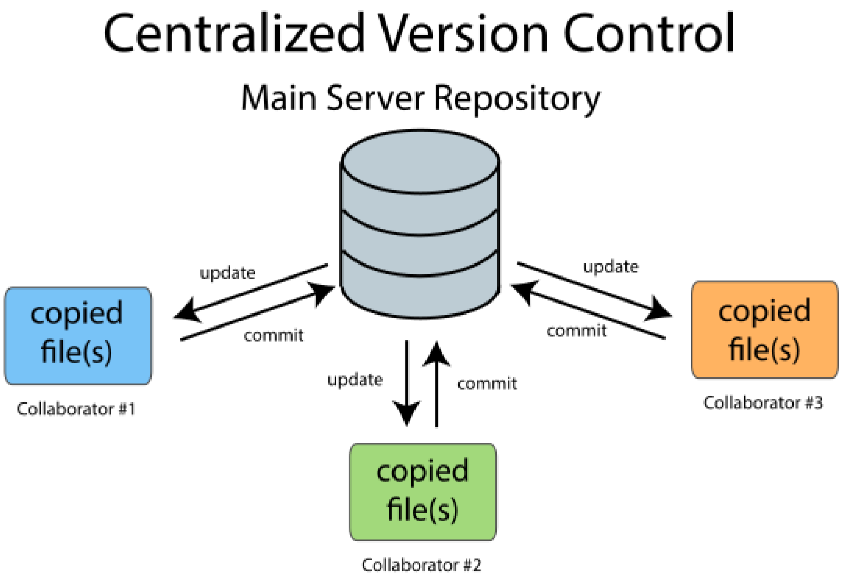


It is generally recognized as a technology with the potential to alter the Internet in general and the financial Internet in particular because of the blockchain's anti-tampering, decentralization, and anonymity transaction capabilities. The blockchain, as a vast distributed ledger, seems to have flaws in terms of the architecture of query processing modules and the efficiency with which clients query the blockchain. The current blockchain storage engine is also incapable of allowing version control and multi-party collaboration.

A research led by Zhaoyi Zhang, Yanru Zhong, and Xiaofan Yu revealed that storage middleware not only dramatically boosts query performance, but also accelerates the development of blockchain applications by several persons working together. [27] In their experiments, they demonstrate how to effectively solve the problem of poor query performance and semantics by linking to other databases, how to implement the version control function by designing the version control semantics separately, and how to effectively use the Pos-tree to delete duplicate data in order to solve the problem of storage space waste while performing version control operations.

2.6 Literature Review

Figure 6: Traditional Centralized Version Control System



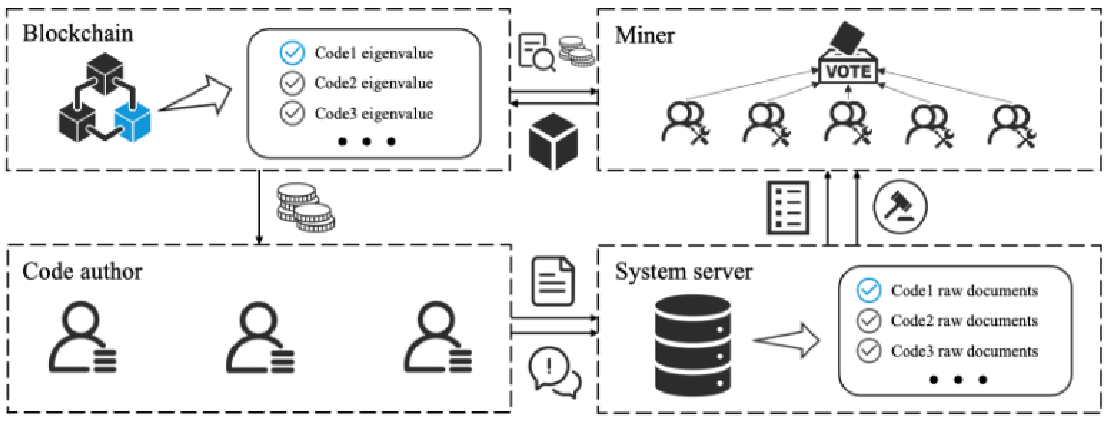
Costa, Felipe Zimmerle da N., and colleagues created Capivara, a decentralized package version control system based on Blockchain technology. This research demonstrated that it is feasible to have a repository for software programs on a Blockchain with distributed consensus, as shown by the proof-of-download approach. However, they provided no evidence of implementation to back up their notion. [28]

A blockchain-based solution and framework for document sharing and version control were proposed by Nizamuddin, Nishara, et al. in order to facilitate multi-user collaboration and track changes in a trusted, secure, and decentralized manner, without the involvement of a centralized trusted entity or third party.[29]

Based on the use of Ethereum smart contracts to manage and regulate the document version control functions among the document's producers and developers, as well as its validators, this solution is implemented. Furthermore, our approach takes use of the IPFS (Interplanetary File System) to store documents on a decentralized file system, allowing for greater efficiency. They've also shown how they've put their concept into action by implementing it. They neglected, however, to account for the system's speed. Due to the fact that they did not address the time-consuming nature of blockchain, such a system would need a large amount of time to accomplish each action. Each transaction on public blockchains such as Ethereum takes at least ten minutes and up to four hours to complete. This degree of time consumption is too high in a real-world scenario where developers may need to submit and retrieve code in less than one or two minutes. This strategy would also be inefficient in terms of cost, since they said that the system would do Ethereum transactions for each push, and as we all know, Ethereum transactions are expensive. Additionally, this article delves deeply into the blockchain system's operation, including how miners will approve submitted code and everything else. This is redundant, since all of them are fundamental Ethereum blockchain processes. The system needed simply to include a smart contract, and the ethereum blockchain would have taken care of the rest. Additionally, the system does not need off-chain code verification by the blockchain's approvers, since code verification can be conducted incredibly rapidly and effectively using our approach, which incorporates encryption and hashing.

RecordsKeeper [30] has suggested a publicly accessible, open source, mineable blockchain ecosystem driven by high-level encryption and blockchain technology for record management and document security that is publically available, open source, and mineable. Data transmission and authorization may be made easier with the use of this organization's strong platform. Peer groups may more easily distribute documents across the network thanks to the decentralized storage network's improved security. Immutable records that can't be changed can be created using blockchain technology, unlike typical database systems like MySQL and Oracle. An end-user may get a rigid structure for maintaining documents on the Blockchain that is verifiable at any moment, enabling them to focus on the particular use case or problem at hand, as a consequence of employing RecordsKeeper. There is no genuine solution for handling document version control, despite the fact that this model has a high degree of nature when it comes to record management. The Swedish government is testing this strategy in an effort to streamline real estate transactions. The Swedish National Land Survey (Lantmäteriet) said today that it is collaborating with blockchain startup ChromaWay, [31] consulting business Kairos Future, and telephone service provider Telia on a proof-of-concept to reduce human errors and increase document security. To confirm the identities of users on a new smart contract system that ChromaWay developed, CEO Henrik Hjelte employed Telia's technology in the proof-of-concept. Using the demo, anybody can observe how smart contracts operate on any blockchain, whether it's Bitcoin or Ethereum. Input and sponsorship came from Lantmäteriet, while Kairos Future created a report. In Magnus Kempe's opinion, the most valuable aspect of the project is the potential for more openness that blockchain-based systems may provide, as well as the potential to position Sweden as a leader in the field of emerging technology. A smart contract-based system uses this method to verify the identity of a user who has registered. Using blockchain technology, any parties engaged in a property title may track, repair, and validate its entire history.

Figure 7: Code Copyright Management System



Recently, blockchain technology has been used for the protection of intellectual property and the administration of corporate transactions. Savelyev et al. [32] investigated the legal implications of blockchain applications in the context of copyright management, and he concluded that blockchain has the potential to disrupt the copyright management sector. Liang et al [33] developed a blockchain-based Intellectual Property (IP) copyright protection system that uses homomorphic encryption to secure intellectual property rights. The researchers' experimental findings indicate that blockchain technology increases the security and reliability of real-time circuit copyright authentication while also decreasing its cost. Xiao et al [34] developed a blockchain-based intellectual property copyright protection method that is focused on IP circuit trade. A distributed random embedding technique, as well as a position mapping function, help to increase the efficiency of the transaction process. Photographic picture copyright protection may be achieved by the use of digital watermarking and perceptual hash technology, which can create immutable hash values that can be put into the blockchain and controlled through particular technology and protocol to offer acceptable protection Meng et al., 2018 [35]. Recent research has also looked at the use of blockchain technology in the protection of digital music copyright, which is now under investigation. Cai et al. [36] utilizes deep learning and blockchain technology to build a digital music copyright protection system and test its effectiveness in the real world. Nan Jing, Qi Liu et al. propose a code copyright management system that is based on blockchain technology.[37] First and foremost, an Abstract Syntax Tree-based code originality verification model is developed and tested. By comparing the uploaded code to other original codes, it is possible to verify whether or not it is really unique. A second use of the Peer-to-Peer blockchain network is to preserve the copyright information pertaining to the original code that was created. Based on the code originality verification paradigm, the nodes in the blockchain network are able to check the authenticity of the code sent to them. Building the blockchain-based code copyright management system follows, which involves the production of blocks, the certification of their legality, and the connecting of these blocks together. All of the steps in the process ensure that the copyright creation is traceable and will not be tampered with in any way. In order to check for plagiarism, the system server turns the code into code eigenvalues using preprocessing methods such as ANTLR and other similar tools. The flaw in this study is that the authors spent a significant amount of time and effort attempting to find a solution to the plagiarism issue. Additionally, in order to address this issue, they have significantly slowed down the transaction processing time. Furthermore, they did not provide any implementation of their proposal. As a copyright management system, these materials have not much to do with a code repository, but are instead utilized for copyrighting the code. It's possible to expand most of them into systems that can function as code repositories. Eleks Labs established an innovative approach to document security by using Ethereum to allow safe storage and transmission for a variety of financial, legal, and other sorts of sensitive data [38]. The organization established a secure environment in which legal transactions may be handled without the need of a third-party middleman. The created system is a permissionless blockchain, which means that anybody may join the network as a participant and observe or commit any transaction. The objective of this system is to offer a safe method for storing and transferring any kind of document, including legal agreements, financial papers, and personal information. The primary objective of designing such a system was to assure efficient and secure transactions without the need of an intermediary. In practice, in order to validate a legal agreement, the parties need a certifier who checks, signs, and registers the agreement's substance. Due to the fact that blockchain technology removes the need for a notary, it is cost efficient. In this case, Ethereum-based smart contracts are used to verify and maintain documents stored on IPFS. Cryptographic methods may be used to verify the signatures of several parties. The smart contract interface enables controlled access and monitoring of document modifications. If they hold an encryption key, every valid member in the network may modify existing documents and monitor the alterations that are traced through the chain. The authors of this work underline the need of sharing, updating, altering, and recreating research and scientific publication data. [39]

Version control on the blockchain and the controlled exchange of digital documents' data were the focus of our investigation in this part. Most of the present ideas described in this section are abstract and high-level in nature, with no details on how they are to be implemented, as we have learned.

3 Proposed Approach

This section summarizes and proposes our strategy, which use the Ethereum blockchain and smart contracts to authorize, monitor, and maintain version control for IPFS-stored code repository. Through transactions and records, our system eliminates the need for a trusted centralized authority and allows the sharing and monitoring of different versions of online documents. To build a decentralized and distributed version control system with high integrity, tenacity, security and reliability throughout the system, we followed the following steps:

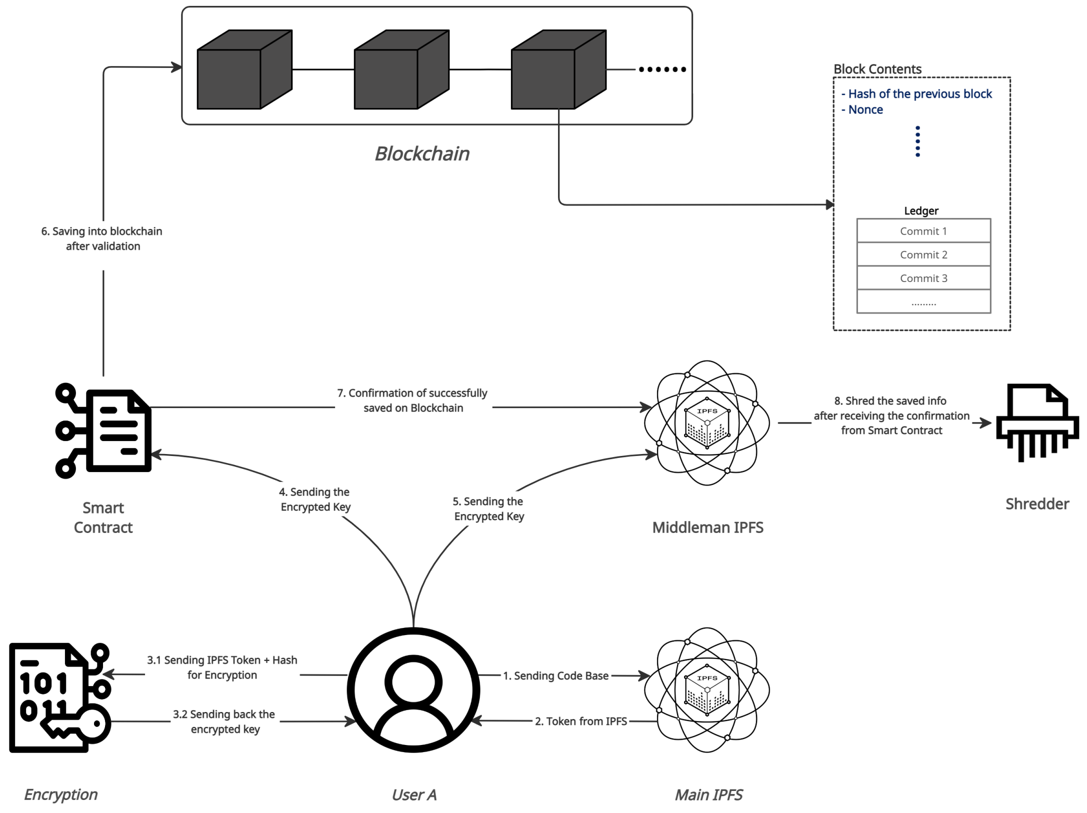
1. Code Submission
2. Code Storage in IPFS
3. Hashing and Encryption
4. Encrypted Token Storage in Blockchain
5. Encrypted Token Storage in Middleware IPFS
6. Code Retrieval via Decryption and Rehashing-Checking

All of these stages will be discussed in further depth in the next sections of this study.

3.1 Code Submission

The technique is deemed complete if all of the code has been prepared by the developers or users and has been submitted to the system for approval before it is implemented. At its core, this component behaves just like any other version control system would, which is to say that it functions in a manner that is completely equivalent to the other systems. They will create a directory at the beginning of the project that will include all of the files and folders that will be required for it at the conclusion of it. After that, they will go to work on the job at hand. Their work will begin as soon as they are placed in the directory to which they have

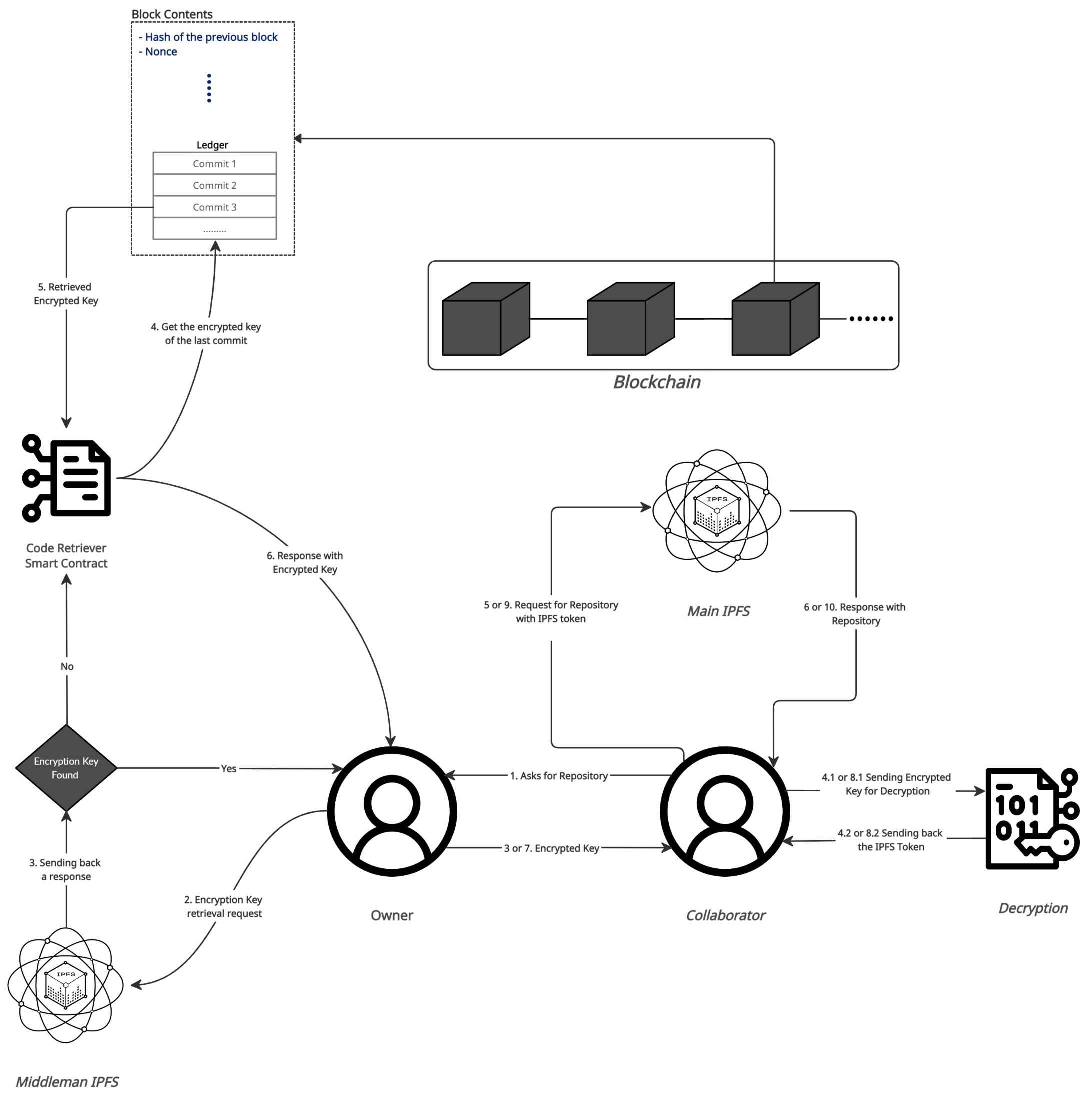
Figure 8: Repository Submission Process



Blockchain
Block Contents
- Hash of the previous block
- Nonce
Ledger
Commit 1
Commit 2
Commit 3
....
6. Saving into blockchain after validation
7. Confirmation of successfully saved on Blockchain
8. Shred the saved info after receiving the confirmation from Smart Contract
Smart Contract
4. Sending the Encrypted Key
5. Sending the Encrypted Key
Middleman IPFS
Shredder
3.1 Sending IPFS Token + Hash for Encryption
3.2 Sending back the encrypted key
1. Sending Code Base
2. Token from IPFS
Encryption
User A
Main IPFS

been allocated. By browsing to the newly generated directory with their mouse and double-clicking on its icon, they may get access to the newly created directory that they made. In the command line, type initializing command to start up the command line, which will enable them to begin working on their project right away. They may incorporate the files in their project by following the steps outlined below on their computer: Include the files using the git add command (git adds). The following two commands are used in git to commit the changes: commit and rollback as well as git commit It is necessary to perform the following instructions in order to commit the modifications: design and develop a piece of computer programming from the ground up. Every project includes a Readme file, which may be in plain text or Markdown format and offers an overview of the project's capabilities. The Readme file can be in plain text or Markdown format. This file is essential in order to get things going. The Readme file is provided only for demonstration purposes alone. As soon as a piece of software is run for the

Figure 9: Repository Retrieval Process



Block Contents
- Hash of the previous block
- Nonce
Ledger
Commit 1
Commit 2
Commit 3
....
Blockchain
5. Retrieved Encrypted Key
4. Get the encrypted key of the last commit
Code Retriever Smart Contract
6. Response with Encrypted Key
Main IPFS
5 or 9. Request for Repository with IPFS token
6 or 10. Response with Repository
No
Encryption Key Found
Yes
1. Asks for Repository
3 or 7. Encrypted Key
4.1 or 8.1 Sending Encrypted Key for Decryption
4.2 or 8.2 Sending back the IPFS Token
3. Sending back a response
2. Encryption Key retrieval request
Owner
Collaborator
Decryption
Middleman IPFS

first time, it's almost always the case that the application generates a Readme file as the first file it creates (and then later added to and committed to).

It has now been established for them a personal repository on the network, which can be accessed using their personal computer. It is not necessary to repeat the process if the file has been successfully uploaded to the system repository the first time. Customers must first enter into the system and then link their crypto wallets to the blockchain, which takes a few minutes. After that, they will be able to access their accounts. Following the completion of the file or code, the new repository button will appear on the right-hand side of the screen. By just clicking on the button to the right of the code, we will be able to submit the code in a matter of seconds.

What occurs in this section is that the user uploads the code files so that they may be uploaded into the distributed storage, hashed, and encrypted before being put into the blockchain at a later time.

3.2 IPFS Codebase Storage

IPFS is a file-sharing system that attempts to overcome the shortcomings of both the HTTP based client-server web paradigms. So, instead of a traditional HTTP client-server architecture, we chose to employ IPFS as our file sharing mechanism in our design. The following are the grounds behind this decision:

- Data is requested using the URL on which the data is housed in a standard HTTP style. On the other hand, data on IPFS is requested using the data's cryptographic hash.
- Data cannot be accessible in a standard HTTP architecture if the server is offline or fails, or if any connection is broken. But on IPFS data is replicated among numerous nodes, allowing it to be retrieved anytime it is required.
- The bandwidth offered in a standard HTTP paradigm is limited since several clients request from a single server at the same time. On the other side, bandwidth is high on IPFS since data is requested from the nearest peer who has a copy of that material.
- To make material publicly accessible in a typical HTTP approach, one must either build up or pay for a hosting server. Uploading material on IPFS, on the other hand, does not need a host server; instead, the data is hosted on the network by each node.

The user will make a submission to the system of the code or repository. It will be saved in the IPFS system after it is submitted. The system will then get a token from the IPFS system that has an address that points to the remote repository.

Anyone may use this URL to access the IPFS and remotely locate the codebase. The request and storage of data will be safe since they will be encrypted using the data's cryptographic hash. This approach also ensures data availability, since data is duplicated over several nodes, enabling it to be accessed at any moment. As a result, system users will not be inconvenienced if one of the servers loses data or becomes destroyed. Additionally, our suggested system would enable high-speed data retrieval and transmission through IPFS, since users will always get data from the system's closest node.

3.3 Hashing and Encryption

It is not immediately transmitted to distributed storage or the blockchain when the code is uploaded in step 1. An internal mechanism takes place throughout this transitional period. Our goal is not to upload the whole program to the blockchain, but rather to upload a single token. Furthermore, since the nodes or validators of the blockchain system should have no contact with the information or code that we are working with in this system, they should be unable to look them up in the blockchain's ledger. As a consequence, before transferring information to the blockchain, we must first encrypt it to prevent it from being read.

With a distributed storage system, we can simply encrypt and share a token or even a link that points another person to the desired repository that is stored remotely. However, when the ledger transactions are processed, how would the code owner or writer be identified by those transactions? It is not sufficient to just provide the URL of the repository. In addition, we must devise a way for associating the encrypted token's owner's identity with it. We're going to use the public key of the owner to represent the identification in this case.

But, we cannot be guaranteed that no one will ever hack into the distributed system of storage (IPFS), or that an ill-intentioned person inside the system would not modify its code or documentation. In order to prevent this, we feel that we should include extra information in the blockchain ledger that would alert the system to the fact that there has been a change in the code, which we believe is necessary to avoid this. In order to do this, the document saved in the IPFS is hashed and then added to the encryption key pool. Additional proof of ownership is provided by the provision of the owner's identifying information (public key).

This is how the ownership, integrity, and immutability of the repository are protected by the solution we developed.

3.4 Encrypted Token Storage in Blockchain

Blockchain technology is well known for its inability to store large amounts of data at one time. When it comes to storing enormous file sizes, blockchains are inefficient. It is not a particularly scalable or efficient path for anything other than core ledger data and associated hashes to be stored "on-chain." It is also quite costly to store data "on-chain." Costs may mount up each terabyte of data sent on the chain per transaction, with costs accruing each time the data is accessed.

Consequently, there is no discussion of keeping the actual code on the blockchain. In this stage, the blockchain will only store the encrypted code of the complete codebase, which is made up of a token that has been obtained from the IPFS storage and a hash that has been generated. In order to create the hash, the complete codebase as well as the writer's key will be used. After being properly validated by the validators or miners, the whole key will be recorded on the blockchain and accessible by everybody. As a result, once the key has been saved, the ownership information of the code is assured since the identity of the person who wrote it is integrated into the code due to the fact that the blockchain system is practically unchangeable. As soon as the encrypted code has been entered into the blockchain, it is available to be seen or read by anybody.

3.5 Encrypted Token Storage in Middleware IPFS

But there is a problem with storing the encrypted tokens just within the blockchain. Blockchains are considered to be sluggish when it comes to real-time transactions. [40] Due to the complexity of the transactions and the blockchain's encrypted, distributed structure, blockchain transactions may take longer than "traditional" payment methods like cash or debit cards.

We need to be able to quickly access each other's data in order to work together on coding projects. This method will make it happen. When the blockchain is not verifying the encrypted code, a middleware system consisting of a centralized database is expected to be employed. Because IPFSs, which are comparable to single source remote servers, run efficiently and, in certain situations, faster than traditional remote servers, it will guarantee that persons working obtain their code in real time. [41] [27]

Since the data has already been recorded into the blockchain, the middleware will delete it following confirmation. Until then, users must rely on middleware to exchange encrypted codes. This will theoretically impair our proposed system's decentralized nature, but it is inevitable because any version control system must be operational, accessible, and consistent in real time. A single source of information poses certain security problems; however, this is a minor worry since the Ethereum network routinely confirms transactions in less than ten minutes, leaving little time for an assault on IPFS. Because IPFS is a dispersed network, hackers cannot target it.

Our system will look for a code by first checking the middleware IPFS, and if the code is not found there, it is clear that the information has been validated and inserted into the blockchain. As a consequence, it will scan the blockchain for information and extract it. It will assure system consistency.

3.6 Code Retrieval via Decryption and Rehash-Checking

Now for the section in which someone attempts to obtain a code. To get the Code, the individual must first determine the location of the code. If the code has been submitted to the system, it has been stored in the IPFS Repository. However, it continues to refuse to disclose the whereabouts of its encrypted code. The code's true owner or author receives the encrypted token from the system, which was created using the code, the owner's key, and the token from the IPFS repository. The code is either stored in the middleware or verified and stored in the blockchain ledger. When a collaborator requests it, the owner just delivers the encrypted code to them.

The retriever receives only the encrypted code from the submitter. The individual decrypts the code using the sender's public key and then obtains the IPFS token and the codebase's hash. This decryption step verifies that the code was transmitted by the sender and was not changed or sent by a third party. The receiver now use the IPFS token to get the code that was stored inside. Now, he hashes the codebase and compares it to the hash from the previously decrypted code. If the code matches, then the IPFS code that was obtained is the correct one. Additionally, it verifies that the IPFS code has not been modified.The portion in which someone tries to gain a code is now up for discussion. In order to get the Code, the person must first discover where the code is located. It is kept in the IPFS Repository if the code has been submitted to the system and has been accepted by the system. It, on the other hand, has maintained its refusal to reveal the location of its encrypted code. The authenticated owner or creator of the code gets an encrypted token from the system, which was generated with the help of the code, the owner's key, and the token from the IPFS repository, and which is then sent to him or her. The code is either saved in the middleware or checked and recorded in the blockchain ledger, depending on the implementation. In response to a collaborator's request, the owner simply sends the encrypted code to that collaborator. The submitter is the only one who may provide the code to the retriever. The person decrypts the code with the help of the sender's public key, and then retrieves the IPFS token as well as the hash of the codebase. This decryption phase ensures that the code was communicated by the sender and that it was not altered or delivered by a third party after it has been encrypted. The IPFS token is now used by the receiver to get the code that was previously stored inside it. He hashes the codebase and compares it to the hash from the previously decrypted code to see if there is any difference. The IPFS code that was acquired is correct if the code obtained matches the code that was obtained. Additionally, it checks to make sure that the IPFS code has not been altered in any way.

3.7 Implementation

We developed our system utilizing Ethereum, IPFS and React as front-end. For testing purposes, we first used our local network as our blockchain environment. Later, the Ropsten Test Network was established to test the technology in a real-world setting. Remix is the IDE we used to write our Solidity code. For testing purposes, two Ethereum addresses were used in Remix. Each participant, developer and collaborator, gets 100 Ether for testing purposes. We set their addresses to be 0ace3b7d915458ef540aade6068fe2f4ae8fa733c and 067383cfd2b6eeeeeeee8ffdc160c. The contract state is used as a requirement for the execution of any function call when functions are built to be called in a certain sequence. We requested 1 ether on the Ropsten Test Network and got it. After that, we used it for all of our transactions.

```
contract SmartContract {
    string[] public storedData;
    event myEventTest(string eventOutput);
    function set(string memory myText) public {
        storedData.push(myText);
        emit myEventTest(myText);
    }
    function get() public view returns (string memory) {
        return storedData[storedData.length - 1];
    }
}
```

Listing 1: SmartContract

```

    function setHandler(event) {
        event.preventDefault();
        setStatus("Sent to IPFS");
        var hashed_pubkey_code = sha256(
            defaultAccount + event.target.setText.value
        );
        var data = hashed_pubkey_code + " " + ipfsLink;
        var encr_codelink_hash = CryptoJS.AES.encrypt(
            JSON.stringify(data),
            defaultAccount
        ).toString();
        contract.set(encr_codelink_hash); // sends the encrypted
 data to the blockchain
    }
    function getCurrentVal() {
        console.log(contract);
        let val = contract.get().then((val) => {
            setErrorMessage(val);
        });
    }
```

Listing 2: Encrypted Key Push and Pull Functions

```

    function decryptMessage(event) {
        event.preventDefault();
        var ownerPublicKey = event.target.ownerPublicKey.value;
        var encryptedMessage = event.target.decrypt.value;
        var bytes = CryptoJS.AES.decrypt(encryptedMessage,
 ownerPublicKey);
        var decryptedData = JSON.parse(bytes.toString(CryptoJS.enc
 .Utf8));
        setDecryptedMessage(decryptedData);
    }

    function findHash(event) {
        event.preventDefault();
        var ownerPublicKey = event.target.
 ownerPublicKey_for_hashchecking.value;
        var code = event.target.code.value;
        var hashed_pubkey_code = sha256(ownerPublicKey + code);
        setHashedValue(hashed_pubkey_code);
    }
```

Listing 3: Decrypting and Hashing Functions

4 Result

4.1 Security and Vulnerabilities

In order to prevent cyberattacks, it is critical that the smart contract's implementation be devoid of defects and weaknesses. Ethereum smart contracts include a number of security flaws, which we'll examine in this section. Code and data on the network need to be protected since they might be attacked. Clearly, blockchain engineering is needed to overcome this difficulty brought by smart contract programming and other blockchain-based applications [42]. In the solidity language, bad programming techniques are to blame for the security vulnerabilities. It's important to note that, in contrast to web apps, the Ethereum platform is directly tied to money, thus any misuse results in quick and serious financial loss. Many industries have struggled to implement an effective security architecture because of these large losses. Smart contract vulnerability testing is now a key aspect of ensuring the soundness and stability of a contract. If you look at an Ethereum-based smart contract for the DAO project, you can see that it had a severe technical weakness and was hacked in 2016. This led to the theft of 3.6 million Ether [43]. As a venture capital fund for crypto and decentralized platforms, the DAO was designed The DAO hack and other incidents have altered people's perceptions of Ethereum's security and brought attention to the issue of cryptocurrencies' and Blockchain technology's security. The smart contract's security is constantly threatened by bugs in the code.

Here are some known vulnerabilities:

- Solidity code
  - Call to unknown address
  - Gasless send
  - Reentrancy attack
  - Exception disorders

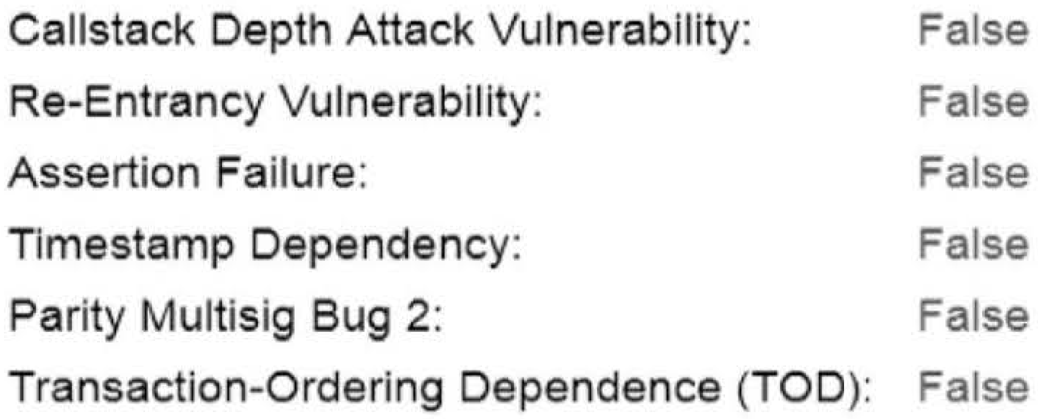
- Type Casts

- Blockchain
  - Generating randomness
  - Unpredictable state
  - Time Constraints

- Ethereum virtual machine
  - Immutable bugs
  - Ether lost in the transfer
  - Stack size limit

Oyente [44] is an essential smart contract analysis tool. Ethereum-based programming languages like as Solidity and Serpent may be used with Oyente. It is also compatible with a lower level Lisp-like language (LLL). This tool have been used to check our smart contract for known security flaws and threats. The report provided by the Oyente tool shows that all identified vulnerabilities were ”False,” and our smart contract was found to be safe enough and devoid of any known flaws.

Figure 10: Oynote Report



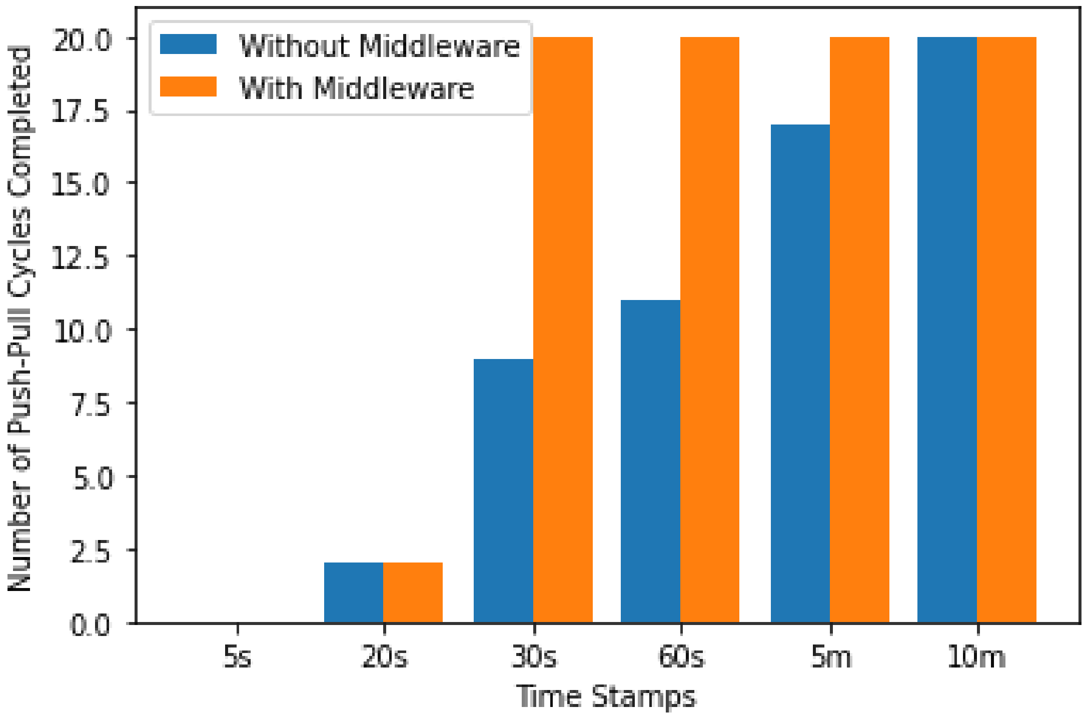
```
Callstack Depth Attack Vulnerability:        False
Re-Entrancy Vulnerability:                   False
Assertion Failure:                           False
Timestamp Dependency:                        False
Parity Multisig Bug 2:                       False
Transaction-Ordering Dependence (TOD):       False
```

4.2 Performance

Blockchains are considered to be sluggish when it comes to real-time transactions. Due to the complexity of the transactions and the blockchain's encrypted, distributed structure, blockchain transactions may take longer than "traditional" payment methods like cash or debit cards. Using Bitcoin to pay for a coffee around lunch is difficult due to the lengthy transaction time. This is not an option unless the seller is ready to take some risk. This approach may theoretically be used to record transactions or interactions in an Internet of Things context. The network's file chains are growing in size, but the number of machines reading and writing to them is increasing. They're merely files on a computer. We expect that advances in engineering and processing speed will soon eliminate this issue. For now, there is concern. So we need to be able to access one other's data quickly in real time to work on coding projects. This plan will make it happen. When the blockchain is not verifying the encrypted code, a middleware system consisting of a centralized database is expected to be employed. Because IPFSs, which are comparable to single source remote servers, run efficiently and, in certain situations, faster than traditional remote servers, it will guarantee that persons working obtain their code in real time. Since the data has already been recorded into the blockchain, the middleware will delete it following confirmation. Until then, users must rely on middleware to exchange encrypted codes. This will theoretically impair our proposed system's decentralized nature, but it is inevitable because any version control system must be operational, accessible, and consistent in real time. A single source of information poses certain security problems; however, this is a minor worry since the Ethereum network routinely confirms transactions in less than ten minutes, leaving little time for an assault on IPFS. Because IPFS is a dispersed network, hackers cannot target it. Our system will look for a code by first checking the middleware IPFS, and if the code is not found there, it is clear that the information has been validated and inserted into the blockchain. As a consequence, it will scan the blockchain for information and extract it. It will assure system consistency.

We experimented with and without the usage of Middleware in our own system. We manually did 20 push and attempted to pull a period of many hours on the Ropsten Test Network. Here is what we discovered as a consequence of our research:

Figure 11: Performance Analysis



Due to the fact that it has not yet reached the middleware or the blockchain, we discovered that the no system can pull within 5 seconds of a push being initiated. After 20 seconds, we can observe that both systems are capable of pulling two pushes. Due to the fact that the middleware IPFS system has not yet been completed, but the blockchain has confirmed the 2 push information, this is the case. Following that, we can notice that all of the push and pull operations are performing smoothly in the system that has middleware integrated, as opposed to the system that does not have middleware service integrated. This finding demonstrates that employing a middleware service in conjunction with blockchain is preferable than a system that just relies on blockchain.

4.3 Discussion

4.3.1 Ensuring Writers Ownership

Using a Private Key to sign a given message with Public Key is the most reliable method of proving ownership of crypto currency. This allows the third-party to verify that the counterparty really understands the corresponding Private Key without having to divulge the key itself or send a transaction to the counterparty.

Our suggested technique is comparable to the one we already use. We know that blockchains are immutable because the code's true author's information is encoded into the encrypted message that is stored in the blockchain's ledger. An immutable ledger in blockchain is a record that can never be modified. It's impossible to modify, thus the data is safe. This ensures a high level of security. Without collusion, it is almost impossible to make any modifications to an immutable system. The immutability of blockchain is due to cryptographic hashes, one of the essential components of the blockchain. It is impossible to decode a hash, which is a major benefit of using it. That's why it's so in demand. To put it another way, SHA-256 is the most often used hash algorithm. A checksum is generated as a result of the hash function's input.

It's possible to utilize the blockchain as a credible evidence of authorship in the event that someone tries to claim someone else's code as their own.

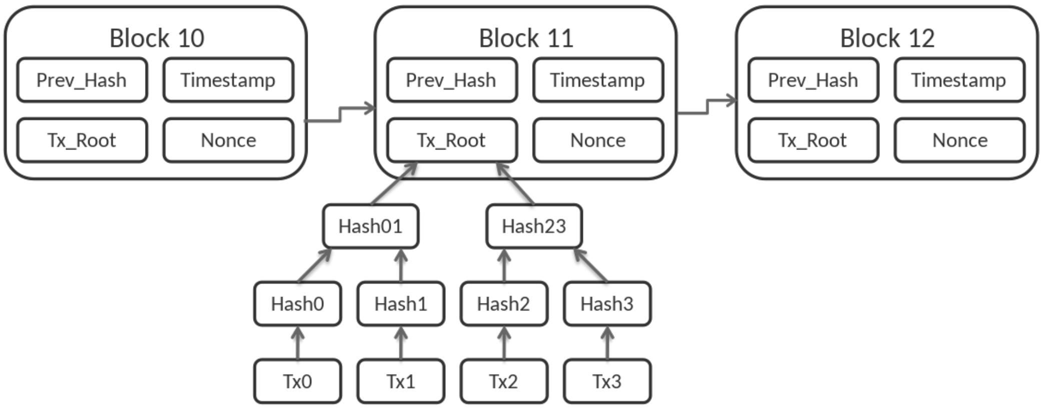
4.3.2 Ensuring Code Integrity

Our suggested solution guarantees the code's integrity. The code in typical centralised system may be changed in numerous ways. Such as hacking, breakdown in the system or even harm done by some ill-intentioned individual in the central authority as well.

Cryptographic hashes are one of the key principles that make blockchain immutable, which is why blockchain is immutable. The most significant advantage of hash is that it cannot be reverse-engineered. That is why it is so well-liked. The most used hash function is SHA-256, which stands for Secure Hash Algorithm 256.

There is an input that is fed into the hash function, and the output is a checksum. In the following diagram, we can see how 'Blockchain is Disruptive' is used as an input, followed by hashing, which results in an encrypted output as a checksum. In the diagram below, after hashing a block, its checksum is an input to another block, which generates a checksum as an output. Every iteration here will result in a new checksum. In a block, transactional data is processed using the preceding hash, as well as the Meta-Data and TX Data Hash, which are all hashed together. As a result, the checksum computed at each block is always unique. This explains why blockchain is, to some extent, immutable. Trades confirmed by a blockchain network in this system include squares of data embedded with timestamps, which is ensured by a hashing cycle. It connects the preceding square's hash and consolidates it. This device establishes the ordered chain that connects each square. The hashing continually incorporates the previous square's meta-information while establishing another hash for it, forming a connection between the square and the chain, which at that point becomes "rugged." The evidence imply that this is a formidable system. In any case, there are a few issues that this component must deal with.

Figure 12: Blockchain Structure ensuring Integrity



The fundamental advantage of blockchain is that data cannot be changed; nevertheless, as shown by previous databases, data may be easily modified and wiped. In the event that the data is tampered with, the blockchain will fail.

Making changes to both disconnected and live blockchain technologies is difficult. When people speak to the blockchain as "permanent," they mean that it is hard to make changes without a conspiracy, not that the information cannot be changed. As a result, it solves the question of why blockchain is immutable. Furthermore, this innovation has both positive and negative implications for information security. Ledgers that convey blockchain innovation may ensure an application's whole history and information trail. When a transaction is added to the blockchain, it remains there to represent the record up to that point in time. The chain's legitimacy may be confirmed at any moment by simply re-calculating the square hashes — if there is a disagreement between block information and its corresponding hash, it indicates that the transactions are not real. This enables associations and their industry controllers to quickly identify information breaches. This ensures total data integrity.

One goal of IPFS is to preserve user's data by enabling users to keep data while lowering the risk of such data being lost or accidentally wiped. Permanency is a term used to describe this situation. Nodes on the IPFS network may automatically cache resources that they download in order to keep those resources available to other nodes. This system is dependent on nodes being willing and able to cache and share network resources. Because storage is limited, nodes must clear some of their previously cached resources to make room for new resources. This method is known as garbage collecting. Data may be pinned to one or more IPFS nodes to ensure that it remains and is not destroyed during garbage collection. Pinning allows us to regulate storage space and data protection. As a result, we should use that option to pin anything we want to keep on IPFS indefinitely.

Out suggested solution overcomes all these difficulties. As our system employs Blockchain, it is difficult to manipulate any data that is residing in there. The blockchain includes encrypted code that has the hash of the full codebase in it together with the identity of the individual who has developed it. So, any discrepancy between the information in blockchain and the information in the real code may be quickly recognized. Also since we implemented IPFS, it offers even more integrity for our repository, as it gives end-to-end integrity. Even if the recorded data in one server of the ipfs is distroyed or updated, the copies of that data in other servers in the distributed system must not be damaged. So, the code in the system will eternally be secure.

4.3.3 Scalability Management

Scalability of blockchain networks refers to the platform's capacity to handle rising transaction loads and node counts.

The consensus protocol or method is used to disseminate, verify, and complete a transaction on a blockchain network. Also, in a blockchain network, this consensus technique balances decentralization, scalability, and security. Thus, the consensus method influences the blockchain network's performance. Network latency is the most important factor affecting dispersed network performance. After validation, the transaction is broadcast to all nodes for majority-based consensus. It minimizes latency and increases overall performance. Infrastructur Blockchain nodes operate on-premises or in the cloud. Node performance will be impacted by the absence of dedicated infrastructure resources (CPU, memory, hard disk). So infrastructure size and IOPS allotment are crucial. Having more nodes slows down transaction propagation and consensus, affecting overall performance. A leader node or other peer node's validation history may help solve this problem. Easy transactions - Most benchmarking research or assertions are based on lab testing. Processing latency increases as smart contracts get more complex in terms of validation logic and read/write operations to the ledger. Larger payloads need more network traffic to duplicate between nodes. Large payloads and documents should be kept offchain and referenced on blockchain. A blockchain network's transactions and state are commonly stored as key-value pairs. The database's efficiency impacts the network's overall performance. With several nodes and hence high availability, each node's transaction handling capacity regulates how many transactions are allowed for further processing by the client applications. This directly impacts network performance. [45]

Simplifying the consensus mechanism may help solve the blockchain's scalability issue. Proof of Work consensus is currently used with Bitcoin and other famous blockchains. Despite its slowness, proof-of-work consensus ensures security. Many blockchain networks regard Proof-of-Stake as a solution to scalability issues. The PoS consensus mechanism does not need miners to employ massive computing power to solve cryptographic algorithms. Choosing validators based on network stakes ensures unanimity. Using PoS consensus for Ethereum networks has several advantages. Improved network capacity, security, and decentralization. Sharding is a popular on-chain scaling technique. One notable example is sharding, a layer-1 scalability strategy for blockchain networks based on distributed databases. Sharding divides huge transactions into smaller, more manageable bits of data. The network processing all shards at once allows for parallel processing of many transactions. Sharding allows data distribution over several nodes while ensuring data consistency. Shards interact to communicate addresses, general status, and balances via cross-shard communication protocols. Layered blockchain may be a solution to scalability concerns. It's a decentralized network design that leverages the main blockchain to establish network parameters. The secondary chain network ensures rapid transaction completion. Nesting seems to be the most viable layer-2 solution to blockchain scalability. [46]

We made our system scalable by integrating off-chain external database to operate as a second layer to our real blockchain network as, it will be absurd to consider if the blockchain can be a storage for the real coding which may be terabytes in size, since it implies all the nodes of the network would have to carry around this much data. So, our approach is to make use the IPFS as the second layer to our system where the real data will be kept and the blockchain will carry the instance that will make sure that the data in the off chain IPFS is valid.

4.3.4 Speed Issues

In the world of cryptocurrency, it is well known that blockchains are very slow when it comes to real-time transactions. [47] [24] It is possible that blockchain transactions may take longer to complete than "conventional" payment methods such as cash or debit cards because of the intricacy of the transactions and the encrypted, distributed nature of the blockchain. To use Bitcoin to pay for a cup of coffee during lunch hour is tough since the transaction may take several hours to complete. Unless the seller is willing to incur some amount of risk, this is not an option. As a theoretical application, this method might be used to blockchain networks that are not just utilized for the storage of cash, such as recording transactions or interactions in an Internet of Things setting. This network's file chains are becoming larger, but as the number of computers reading and writing to it grows, they risk becoming unwieldy and unmanageable. After all, they're just computer files. We believe that this problem will soon be a thing of the past because to advancements in engineering and processing speed. There is still a worry for the time being.

So, it's important to be able to access each other's data rapidly in real life, so we can collaborate on coding projects in real time. This is how this strategy will ensure that it happens. It is anticipated that a middleware system, which would consist of a centralized database, will be used to store the encrypted code when the blockchain is not certifying it. It will ensure that individuals who are working get their code from one another in real time since IPFSs, which are similar to single source remote servers, operate effectively and, in certain cases, even more quickly than conventional remote servers. [41]

Due to the fact that the data has already been entered into the blockchain, the middleware will erase it after it has been confirmed. Users will have to depend on the middleware till then in order to acquire the encrypted code from one another. The decentralized character of our proposed system will be technically compromised as a result of this, but it is an unavoidable evil since it is essential for any version control system to be operational, accessible, and consistent in real time. Because the information is contained in a single source, there are some security concerns; however, this can be considered a very minor concern because the Ethereum network typically validates transactions in less than ten minutes, providing a very small window for an attack on the IPFS to take place. Because IPFS is a distributed network, the very nature of IPFS will prevent hackers from launching attacks against it as well.

The procedure that our system will take in order to search for a code is to first check the middleware IPFS, and if the code is not located there, it is certain that the information has been erased from there and has been confirmed and entered into the blockchain. As a result, it will search for information on the blockchain and extract it from there. It will ensure consistency in the system.

4.3.5 Cost

There are several Source Code Repository Hosting Services available. There are several well-known free ones. Bitbucket, Github, Sourceforge, and so forth. There are also several paid Source Code Repository Hosting Services available for premium services. For instance, ProjectLocker, Gitlab Premium, Fog Creek Kiln, and so on. People pay for the subscription edition of the Source Code Repository Hosting Service because it has additional premium features. They increase accountability. Furthermore, it features improved technology that gives greater security, integrity, consistency, and immutability.

These are the benefits that our suggested system will bring as well, thanks to the power of Blockchain, IPFS, and Middleware. Because our system uses Blockchain, it is impossible to change any data that is stored there, making it more reliable, safe, and consistent than other paid services on the market. The blockchain contains encrypted code that contains the hash of the whole codebase as well as the identify of the person who created it. As a result, any inconsistency between the information in blockchain and the information in the actual code may be identified fast. Also, since we added IPFS, our repository has gained even more integrity since it provides end-to-end integrity. Even if the recorded data on one of the ipfs servers is corrupted or modified, the copies of that data on other servers in the distributed system must not be harmed. As a result, the code in the system will remain secure in perpetuity.

People who now pay for Source Code Repository Hosting Service will have no trouble paying for the solution we have presented, which has greater technology and accountability.

4.3.6 Centralised vs Decentralised Repository

Databases can be classified as either centralized or distributed based on how they store and access data. Distributed database works with multiple database files, while centralized database only has access to the same single database file. If all of your data is housed in a single database, you're dealing with a centralized database. There are two or more databases in a distributed database that are located in different parts of the network. Because there is only one database file, managing, updating, and taking backups of data is much easier in a centralized database. Distributed databases require more time to synchronize data because they contain multiple databases. It takes longer to retrieve data from a centralized database because multiple users are accessing the same file. It is frequently used. Because data is retrieved from the nearest database file when using a distributed database, access times are faster. Users lose access to a database in a centralized database if the database fails. Users can still access other database files in a distributed database even if one database fails. Data consistency is better in a centralized database. It gives the user a complete picture. Data replication is possible in a distributed database. Because of this, there may be some discrepancies in the data. Considering all these reasons, we chose to use a Decentralised Database as our repository.

5 Conclusion

5.1 Summary

Many software developers use version control systems to manage source code and keep track of the many versions they've worked on. In this proposed system, the code is stored on a distributed network of servers, while a decentralized blockchain ensures ownership information and the immutability of the repository. Using our strategy, there is no need to have a centralized authority that is trustworthy. Cryptographic hashes are one of the key principles that make blockchain immutable. The most significant advantage of hash is that it cannot be reverse-engineered.

In a block, transactional data is processed using the preceding hash, as well as the Meta-Data and TX Data Hash, which are all hashed together. Every iteration here will result in a new checksum, which guarantees the code's integrity. Data may be pinned to one or more IPFS nodes to ensure that it remains and is not destroyed during garbage collection. This enables associations and their industry controllers to quickly identify information breaches and ensures total data integrity. In the world of cryptocurrency, it is well known that blockchains are very slow when it comes to real-time transactions. Even if the recorded data in one server of the ipfs is distroyed or updated, the copies of that data in other servers in the distributed system must not be damaged. We believe that this problem will soon be a thing of the past because to advancements in engineering and processing speed. There are several Source Code Repository Hosting Services available. There are also several well-known free ones, Bitbucket, Github, Sourceforge, and so forth. For instance, ProjectLocker, Gitlab Premium, Fog Creek Kiln, etc. People pay for the subscription edition because it has additional premium features. It features improved technology that gives greater security, integrity, consistency, and immutability. Because our system uses.Blockchain, it is impossible to change any data that is stored there. This makes it more reliable, safe, and consistent than other paid services. Most of the Source Code Repository Hosting Service these days are centralised, which is a bad thing as there is a single source of failure.

To authorize, monitor, and execute version control on a code repository for a distributed network of computers, we employ a public blockchain and smart contracts. Our approach eliminates the necessity for a trustworthy central authority. The immutability of the code, as well as the ownership information of the code writer, are both protected by a public blockchain. A distributed network of servers protects the code repository and its contents. Broadcast Encryption protects the transmitter's identity. We provide a public blockchain-based blockchain-based solution for digital document version control systems. In the first place, using our suggested solution removes the need for a trustworthy third-party authenticator. The most important features of our approach in terms of overall system design and architecture, as well as the most critical interactions between players, are highlighted in particular. Registering new developers, authorizing new code contributions, retrieving other people's work, the concepts of logical flow and interactions, and developing and testing the system's overall operation will all be covered. Decentralized document version management and sharing was one of our main contribution. Blockchain, smart contracts, and the IPFS file system all play a role in our approach. By removing any reliance on any third party, the suggested solution eliminates all of those problems. Also, our suggested system displays a more quick performance than the rest of the proposed systems out there since we employed intermediary IPFS as a temporary hub to store encrypted codes while the blockchain is verifying, assuring there is no annoyance among the users. For findings indicated that our solution is incredibly safe, both for utilizing blockchain and also overcoming the weaknesses that smart contracts come with.

5.2 Future Work

Our suggested method for hosting a decentralized source code repository may easily be expanded to other types of documents that need a distributed storage facility and a decentralized network to monitor ownership.

For instance, writers would undoubtedly benefit from a system in which their literary works are stored in a distributed network of servers that is safe, secure, and accessible from anywhere; and their ownership information is stored in a decentralized ledger that is not controlled by anyone, ensuring that their literary works' ownership information is also secure.

A distributed network of servers can also benefit artists, whose work is stored in a safe, secure, and easily accessible location, and whose ownership information is stored in a decentralized ledger that is not under the control of any one party.

This technology may also assist the NFT (Non-Fungible Token) business, since the NFTs are stored in a distributed network of servers, which is safe and secure, and the ownership information is stored in a decentralized ledger that is not controlled by anybody, which is also safe.

Also, from the technological point of view, instead of using IPFS, we may utilize Arweave. [48] The Arweave project is a decentralized data storage system that operates on blockweave technology and the Arweave token, which is a native cryptocurrency. Despite its benefits, IPFS cannot provide permanent storage without the use of workarounds — and even then, the workarounds tend to have downsides and catches that make NFTs unfit for long-term preservation. The cloud, if it is not kept on an incentivized blockchain, is more similar to the internet. Having an NFT saved on Airweave is the best solution for the long-term since it allows permanent storage without the need to pay monthly storage costs.

References

[1] B. De Alwis and J. Sillito, “Why are software projects moving from centralized to decentralized version control systems?” in *2009 ICSE Workshop on Cooperative and Human Aspects on Software Engineering.* IEEE, 2009, pp. 36–39.

[2] S. Nakamoto, “Bitcoin: A peer-to-peer electronic cash system,” *Decentralized Business Review*, p. 21260, 2008.

[3] J. Benet, “Ipfs-content addressed, versioned, p2p file system,” *arXiv preprint arXiv:1407.3561*, 2014.

[4] M. Nofer, P. Gomber, O. Hinz, and D. Schiereck, “Blockchain,” *Business & Information Systems Engineering*, vol. 59, no. 3, pp. 183–187, 2017.

[5] J. L. Zhao, S. Fan, and J. Yan, “Overview of business innovations and research opportunities in blockchain and introduction to the special issue,” pp. 1–7, 2016.

[6] K. Salah, M. H. U. Rehman, N. Nizamuddin, and A. Al-Fuqaha, “Blockchain for ai: Review and open research challenges,” *IEEE Access*, vol. 7, pp. 10 127–10 149, 2019.

[7] M. A. H. Khan, N. Aktar, N. Sultana, S. Akhter, and M. F. Hossain, “Baseline survey for farm productivity improvement through agricultural technologies in charland of mymensingh,” *International Journal of Business, Management and Social Research*, vol. 7, no. 01, pp. 395–411, 2019.

[8] M. S. Ali, M. Vecchio, M. Pincheira, K. Dolui, F. Antonelli, and M. H. Rehmani, “Applications of blockchains in the internet of things: A comprehensive survey,” *IEEE Communications Surveys Tutorials*, vol. 21, no. 2, pp. 1676–1717, 2019.

[9] T. Bocek, B. B. Rodrigues, T. Strasser, and B. Stiller, "Blockchains everywhere-a use-case of blockchains in the pharma supply-chain," in *2017 IFIP/IEEE symposium on integrated network and service management (IM)*. IEEE, 2017, pp. 772–777.

[10] G. Wood *et al.*, "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum project yellow paper*, vol. 151, no. 2014, pp. 1–32, 2014.

[11] N. Z. Benisi, M. Aminian, and B. Javadi, "Blockchain-based decentralized storage networks: A survey," *Journal of Network and Computer Applications*, vol. 162, p. 102656, 2020.

[12] S. Vimal and S. Srivatsa, "A new cluster p2p file sharing system based on ipfs and blockchain technology," *Journal of Ambient Intelligence and Humanized Computing*, pp. 1–7, 2019.

[13] M. Ernst, "Version control concepts and best practices," *Version control concepts and best practices*, 2012.

[14] H. Zhang, B. Liu, H. Susanto, G. Xue, and T. Sun, "Incentive mechanism for proximity-based mobile crowd service systems," in *IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on Computer Communications*. IEEE, 2016, pp. 1–9.

[15] B. Y. Zhao, L. Huang, J. Stribling, S. C. Rhea, A. D. Joseph, and J. D. Kubiatowicz, "Tapestry: A resilient global-scale overlay for service deployment," *IEEE Journal on selected areas in communications*, vol. 22, no. 1, pp. 41–53, 2004.

[16] "IPFS Documentation | IPFS Docs." [Online]. Available: https://docs.ipfs.io/

[17] "Cryptocurrency Prices, Charts And Market Capitalizations." [Online]. Available: https://coinmarketcap.com/

[18] "Ethereum Whitepaper." [Online]. Available: https://ethereum.org

[19] "What is Ethereum?" [Online]. Available: https://ethereum.org

[20] "What is ether (ETH)?" [Online]. Available: https://ethereum.org

[21] Y. Chen, H. Li, K. Li, and J. Zhang, "An improved p2p file system scheme based on ipfs and blockchain," in *2017 IEEE International Conference on Big Data (Big Data)*. IEEE, 2017, pp. 2652–2657.

[22] Q. Zheng, Y. Li, P. Chen, and X. Dong, "An innovative ipfs-based storage model for blockchain," in *2018 IEEE/WIC/ACM international conference on web intelligence (WI)*. IEEE, 2018, pp. 704–708.

[23] M. , "Blockchain Middleware. How we are paving the path to the next... | by MetisDAO | Medium," dec 15 2021, [Online; accessed 2022-04-22].

[24] "The 5 Big Problems With Blockchain Everyone Should Be Aware Of | Bernard Marr," jul 2 2021, [Online; accessed 2022-04-22].

[25] T. Alam, "Design a blockchain-based middleware layer in the internet of things architecture," *JOIV: International Journal on Informatics Visualization*, vol. 4, no. 1, pp. 28–31, 2020.

[26] "Why middleware is a the vital link for blockchain success." [Online]. Available: https://www.linkedin.com/pulse/why-middleware-vital-link-blockchain-success-maurizio-canton

[27] Z. Zhang, Y. Zhong, and X. Yu, "Blockchain storage middleware based on external database," in *2021 6th International Conference on Intelligent Computing and Signal Processing (ICSP)*, 2021, pp. 1301–1304.

[28] F. Z. d. N. Costa and R. J. G. B. de Queiroz, "Capivara: A decentralized package version control using blockchain," *arXiv preprint arXiv:1907.12960*, 2019.

[29] N. Nizamuddin, K. Salah, M. A. Azad, J. Arshad, and M. Rehman, "Decentralized document version control using ethereum blockchain and ipfs," *Computers & Electrical Engineering*, vol. 76, pp. 183–197, 2019.

[30] "RecordsKeeper - Decentralized Database for Decentralized Apps (DApps)." [Online]. Available: https://www.recordskeeper.com/

[31] "ChromaWay." [Online]. Available: https://chromaway.com

[32] A. Savelyev, "Copyright in the blockchain era: Promises and challenges," *Computer law & security review*, vol. 34, no. 3, pp. 550–561, 2018.

[33] W. Liang, D. Zhang, X. Lei, M. Tang, K.-C. Li, and A. Y. Zomaya, "Circuit copyright blockchain: blockchain-based homomorphic encryption for ip circuit protection," *IEEE Transactions on Emerging Topics in Computing*, vol. 9, no. 3, pp. 1410–1420, 2020.

[34] L. Xiao, W. Huang, Y. Xie, W. Xiao, and K.-C. Li, "A blockchain-based traceable ip copyright protection algorithm," *IEEE Access*, vol. 8, pp. 49 532–49 542, 2020.

[35] Z. Cai, "Usage of deep learning and blockchain in compilation and copyright protection of digital music," *IEEE Access*, vol. 8, pp. 164 144–164 154, 2020.

[36] Z. Meng, T. Morizumi, S. Miyata, and H. Kinoshita, "Design scheme of copyright management system based on digital watermarking and blockchain," in *2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC)*, vol. 2. IEEE, 2018, pp. 359–364.

[37] N. Jing, Q. Liu, and V. Sugumaran, "A blockchain-based code copyright management system," *Information Processing & Management*, vol. 58, no. 3, p. 102518, 2021.

[38] "ELEKS Labs - Research and Development Blog." [Online]. Available: https://labs.eleks.com/

[39] "What is blockchain and why should records management professionals care?" Aug. 2020. [Online]. Available: https://www.ironmountain.com/resources/general-articles/w/what-is-blockchain-and-why-should-records-management-professionals-care

[40] K. Gomi, "Council Post: Are Blockchains Vulnerable, Slow And Unfair?" jul 12 2021, [Online; accessed 2022-04-22].

[41] N. Wang, B. Wang, T. Liu, W. Li, and S. Yang, "A middleware approach to synchronize transaction data to blockchain," in *2020 29th International Conference on Computer Communications and Networks (ICCCN)*, 2020, pp. 1–8.

[42] G. Destefanis, M. Marchesi, M. Ortu, R. Tonelli, A. Bracciali, and R. Hierons, "Smart contracts vulnerabilities: a call for blockchain software engineering?" in *2018 International Workshop on Blockchain Oriented Software Engineering (IWBOSE)*, 2018, pp. 19–25.

[43] D. Siegel, "Understanding the dao attack, coindesk," *Retrieved July*, vol. 14, p. 2020, 2016.

[44] L. Luu, D.-H. Chu, H. Olickel, P. Saxena, and A. Hobor, "Making smart contracts smarter," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 254–269. [Online]. Available: https://doi.org/10.1145/2976749.2978309

[45] D. Geroni, "Blockchain Scalability Problem - Why is it Difficult to Scale Blockchain," sep 30 2021, [Online; accessed 2022-04-23].

[46] ——, "Blockchain Scalability Solutions - An Overview," oct 5 2021, [Online; accessed 2022-04-23].

[47] "Blockchain is slow and useless?! - Iconis Agency," jul 28 2021, [Online; accessed 2022-04-22].

[48] T. A. Project, "What is Arweave? | by The Arweave Project | Medium," may 2 2018, [Online; accessed 2022-04-23].