

DECENTRALIZED CLOUD STORAGE USING BLOCKCHAIN

Submitted in partial fulfillment of the requirements for the award of
Bachelor of Engineering Degree in Computer Science and Engineering

By

LOKESH KARTHIK. S

(37110407)



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SCHOOL OF COMPUTING

SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited with Grade “A” by NAAC|12B Status by UGC| Approved by AICTE

JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI - 600 119

March - 2021



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited with Grade “A” by NAAC|12B Status by UGC| Approved by AICTE

Jeppiaar Nagar, Rajiv Gandhi Salai, Chennai - 600 119



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **LOKESH KARTHIK S (17SCS8062)** who carried out the project entitled “**DECENTRALIZED CLOUD STORAGE USING BLOCKCHAIN**” under our supervision from NOVEMBER 2020 to MARCH 2021.

Internal Guide

Dr.D.USHA NANDINI , M.E., Ph.D.,

Head of the Department

Dr.S.VIGNESHWARI , ME.,Ph.D.,

Submitted for Viva Voce Examination held on_____.

Internal Examiner

External Examiner

DECLARATION

I, **LOKESH KARTHIK S (17SCS8062)** hereby declare that the Project Report on “**DECENTRALIZED CLOUD STORAGE USING BLOCKCHAIN**” done by us under the guidance of **Dr.D. USHA NANDINI ,M.E.,Ph.D.**, at Sathyabama Institute of Science and Technology is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in Computer Science and Engineering.

DATE:

PLACE:

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to Board of management of **SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. We are grateful to them.

I convey my thanks to **Dr. T. Sasikala, M.E., Ph.D., Dean, School of Computing and Dr. S.Vigneshwari, M.E., Ph.D., and Dr.L.Lakshmanan, M.E.,Ph.D., Heads of the Department, Department of Computer Science and Engineering** for providing us the necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to our Project Guide **Dr.D.Usha Nandini, M.E., Ph.D.,** for her valuable guidance, suggestions and constant paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the Department of Computer Science and Engineering who were helpful in many ways for the completion of project.

ABSTRACT

Decentralized cloud storage is a Peer-to-peer network where each node provides the storage service to the customer's data. The storage system is based on the blockchain domain where it is completely decentralized. Blockchain ensures the user security and reliability of the user data. Peer-to-peer networking enables the user to store the data in different nodes across the network with security where the data are encrypted. This paper proposes a system which enables smart contract in the cloud storage system where it acts as an agreement between the client and the storage provider. Smart contract lets the user know about what data will be stored and the cost of storage. The decentralized cloud storage is reliable, secure and not power failure prone compared to the earlier systems. This system is integrated with a smart contract which enables to keep an agreement and the transaction will happen accordingly to the implemented smart contract.

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
	ABTRACT	i
	LIST OF FIGURES	v
1	INTRODUCTION	1
2	LITERATURE SURVEY	4
	2.1 LITERATURE SURVEY	5
3	AIM AND SCOPE OF THE PROJECT	8
	3.1 AIM	8
	3.2 PROJECT SCOPE	8
	3.3 OBJECTIVE	8
	3.4 PROPOSED SYSTEM	9
4	METHODOLOGY	11
	4.1 MODULE DESCRIPTION	11
	4.1.1 WEB3J	11
	4.1.2 SOLIDITY	11
	4.1.3 REACTJS	11
	4.1.4 TRUFFLE SUITE	12
	4.1.5 GANACHE	12
	4.1.6 NODE.JS	13
	4.1.7 METAMASK	13
	4.1.8 IPFS INFURA	13
	4.1.9 ETHEREUM	14
	4.1.10 JAVASCRIPT	14
	4.1.11 CSS	15

4.1.12 HTML	16
4.2 SOFTWARE DESCRIPTION	18
4.2.1 BUILDING PEER TO PEER NETWORK	19
4.3 NODE SETUP	19
4.3.1 NODE OPERATIONS	19
4.3.2 MOTIVATION FOR NODE	20
4.4 CLIENT SETUP	20
4.4.1 COMMUNICATION WITH NODE	20
4.4.2 AES ENCRYPTION AND DECRYPTION	21
4.4.3 FILE SPLITTING/MERGING	22
4.4.4 HASH ENCRYPTION	22
4.5 BLOCKCHAIN INTEGRATION	23
4.5.1 BLOCKCHAIN	23
4.5.2 SMART CONTRACT	25
4.5.3 FILE DETAILS RECORD	27
4.5.4 TOKEN TRANSFER	28
4.6 SYSTEM DEVELOPMENT METHODOLOGY	29
4.6.1 SOFTWARE DEVELOPMENT APPROACH	29
4.6.2 REQUIREMENT ANALYSIS	30
5 RESULTS AND DISCUSSION	31
5.1 CLIENT APPLICATION	31
5.1.1 RESULT OF NODE APP	31
6 CONCLUSION AND FUTURE SCOPE	36
6.1 CONCLUSION	36
6.2 FUTURE WORK	37

REFERNCES	38
APPENDIX	39
A) SOURCE CODE	39
B) SCREENSHOTS	44
C) PUBLICATION AND PLAGIARISM CHECK REPORT	48

LIST OF FIGURES

FIGURE NO.	FIGURE NAME	PAGE NO.
3.1	ARCHITECTURE DIAGRAM	9
4.1	P2P NETWORK	15
4.2	SECURE FILE STORAGE PROTOCOL	16
4.3	FILE PROCESSING	18
4.4	FILE STORAGE PROTOCOL	19
4.5	MERKLE TREE	22
4.6	SMART CONTRACT FLOW DIAGRAM	24
4.7	TOKEN TRANSFER	26
4.8	SCRUM METHODOLOGY	28
5.1	METAMASK	30
5.2	GANACHE	31
5.3	FILE UPLOADING	32
5.4	FILE UPLOAD AND RETRIEVING SCREENSHOT	33

CHAPTER 1

INTRODUCTION

In the past decade, technological advancements have been made by research consortiums to adapt data sharing approaches. In such a way, research-based activities can improve through collaboration with the growing fields of Information Technology, Internet of Things and Digitization of every business, organizational work and projects, Information has become the biggest valuable asset for anyone. With the abundance of data and its ever-growing nature, it's equally important to store it in an organized way such that it's easily accessible and secure. For this purpose. Databases are used as a warehouse to store data. Database play a crucial role for any individual as well as any organization and business to store its data. Realizing the importance of data and insufficiency of storage, databases are replicated, distributed and backed up in different ways. Individuals store data in the cloud provided by different privately companies. Organizations set up their data centers at different part of the globe to store its data. For the security and bandwidth, data are scattered and replicated to different servers at different places. This seems to provide good solution for the management of rapidly increasing data. And also ensures data safety. In future, the rate of increment of data is sure to reach high. To cope with it, the current database system needs to be more reliable, safe and available all the time. Cloud servers store the excessive amount of data, which is a centralized authority. There is various type of risks associated with a central authority, such as single point failure. To avoid such failure, third parties are involved to provide data backups. To eliminate third party for developing a trust-based model, a blockchain is introduced to provide trust and transparency. Decentralized storage is a solution, which allows storage of data independently on multiple nodes of the network in the form of a distributed ledger. Peer to Peer network is the distributed network where each node in the network communicates with each other directly or through a series of channels via other nodes. There is no client server to access the resource. Each node will act both as a host or a client as needed. There is no any Central server for controlling the system flow and other nodes. Node is one of the member of P2P

system which is willing to provide store for the clients in return to tokens. Node is fully responsible for handling the client's data. It can view the encrypted data send by client which is stored in node's storage. Node can list all the data of the client and organize it. In our system, node is getting paid for storing the client's data by token of our system. According to the agreement between node and client, the node gets payment as file is downloaded by the client or duration of agreement finishes. Introduction of smart contracts is used here as an agreement between the client and the service provider where the user knows what data is stored and what is the price for the stored data. Smart contracts are the set of protocols in which the decentralized follows and the added advantage is it discloses any confusion between the users and the provider. Smart contracts are implemented in the truffle suite in this project. IPFS is a BitTorrent like network where number of nodes are connected on the network so that the data can be uploaded and downloaded easily. IPFS is used here to distribute the data blocks across the network where several nodes can store the data which is encrypted. The address of this network is available in the IPFS so that the transactions of this network can be watched using wallets like ganache which is used in this project. Ganache is a private wallet where the user can watch the transactions and information about the block is also available. The frontend of this project is created using React which uses JavaScript programming language. This project proposes a decentralized cloud storage using IPFS system where the data stored on this network is more secure and reliable than the conventional storage system. It is a simple and secure file storage system where the file is distributed across the network and when the file needs to be retrieved, decryption is performed on the data to restore in its original form. First, the data is selected and it is fed in the system where the data is first sharded into several number of blocks of bits. The number of shards is determined by the total number of nodes available in the network. The encryption is the most crucial point of this process where the data needs to be secure before distributing it across the network. After the data is separated into several shards which is considered as a data block is hash encrypted so that the receiver of this block doesn't understand the information. This process happens until the entire data is hashed encrypted. Introduction of smart contracts is used here as an agreement

between the client and the service provider where the user knows what data is stored and what is the price for the stored data. Smart contracts are the set of protocols in which the decentralized follows and the added advantage is it discloses any confusion between the users and the provider. Smart contracts are implemented in the truffle suite in this project. IPFS is a BitTorrent like network where number of nodes are connected on the network so that the data can be uploaded and downloaded easily. IPFS is used here to distribute the data blocks across the network where several nodes can store the data which is encrypted. The address of this network is available in the IPFS so that the transactions of this network can be watched using wallets like ganache which is used in this project. Ganache is a private wallet where the user can watch the transactions and information about the block is also available. The frontend of this project is created using React which uses JavaScript programming language. The problem is the storage and processing limitation of network nodes. For this purpose, interplanetary file system (IPFS) is adapted, which is a p2p architecture. There is no risk of single point failure. It's similar to web3, but with different features. It performs content addressing and works in a similar way as bit torrent. Availability of data is ensured by storing it on a decentralized platform; IPFS.

CHAPTER 2

LITERATURE SURVEY

Existing System: Earlier, due to the non-usage of smart contracts in the previous proposed systems have issues with transparency of transactions. The cloud storage system doesn't ensure the user about the cost of the storage of data and what type of data will be stored. Some of the earlier systems didn't feature peer-to-peer networking which seems as a possible disadvantage. Currently, user uses his/her offline storage devices and other secondary storages for data backup and protection. Most of us often use the cloud service of Amazon, Google, Dropbox, Microsoft and others. A huge amount of users data are stored in cloud which is in fact someone's computer or storage devices. Such organization has complete authority over users data. In recent years, trend has increased rapidly in using those data without users acknowledge and permission by those company for their uses and pursue higher benefits from it. Most of the papers doesn't purpose a privacy policy where some information the transaction destination which can be improved with further developing the system. Some papers propose encryption algorithms aren't secure as encryption holds an important process in the cloud storage system where the data can't be interrupted by hackers. Advanced encryption algorithms are being developed which might provide better security to the users.

Proposed System: This paper proposes a decentralized cloud storage using IPFS system where the data stored on this network is more secure and reliable than the conventional storage system. It is a simple and secure file storage system where the file is distributed across the network and when the file needs to be retrieved, decryption is performed on the data to restore in its original form. First, the data is selected and it is fed in the system where the data is first sharded into several number of blocks of bits. The number of shards is determined by the total number of nodes available in the network. The encryption is the most crucial point of this process where the data needs to be secure before distributing it across the

network. After the data is separated into several shards which is considered as a data block is hash encrypted so that the receiver of this block doesn't understand the information. This process happens until the entire data is hashed encrypted. Introduction of smart contracts is used here as an agreement between the client and the service provider where the user knows what data is stored and what is the price for the stored data. Smart contracts are the set of protocols in which the decentralized follows and the added advantage is it discloses any confusion between the users and the provider. Smart contracts are implemented in the truffle suite in this project. IPFS is a BitTorrent like network where number of nodes are connected on the network so that the data can be uploaded and downloaded easily. IPFS is used here to distribute the data blocks across the network where several nodes can store the data which is encrypted. The address of this network is available in the IPFS so that the transactions of this network can be watched using wallets like ganache which is used in this project. Ganache is a private wallet where the user can watch the transactions and information about the block is also available. The frontend of this project is created using React which uses JavaScript programming language.

2.1 LITERATURE SURVEY

Title : Blockchain-Based Secure Storage and Access Scheme for Electronic Medical Records in IPFS

Author : Jin Sun

Year : 2014

Jin Sun proposed an access scheme for electronic medical records in IPFS with blockchain-Based Secure Storage. The author constructed attribute-based encryption scheme for secure storage and efficient sharing of electronic medical records in IPFS environment. The ideology that the author suggests is that we encrypt the data based on attributes and determine the attributes of the users. Each user's private key is related to their respected attributes whereas the ciphertext is related to the policy.

Title : Blockchain-based Decentralized Storage Scheme

Author : Yan Zhu

Year : 2019

Yan Zhu suggests a decentralized storage scheme instead of a conventional centralized system. The provider performs a data integrity certificate to the user and only after it is verified, the user pays the fee for storage. The payment information is stored in the blockchain which is secure.

Title : A Peer-to-Peer Cloud Storage Network

Author : Shawn Wilkinson

Year : 2016

Shawn Wilkinson used a peer-to-peer cloud storage which implements end-to-end encryption which would allow to share and transfer data without a need for third party data providers. The author implements a challenge algorithm which would cryptographically check the integrity and availability of the user data.

Title : Simple Decentralized Storage

Author : David Vorick

Year : 2014

David Vorick introduced a simple decentralized storage system which enables the formation of contracts between peers. Contracts are like agreements between users and the provider which defines what data will be stored and at what price.

Title : A peer-to-peer electronic cash system

Author : Satoshi Nakamoto

Year : 2008

Satoshi Nakamoto introduced peer-to-peer networking which changed the entire domain. The author suggests a user to send payments directly from one party to another without a need for financial institution. This paper provides a solution to double-spending problem with the introduction of peer-to-peer networks.

Title : IPFS - Content Addressed, Versioned, P2P File System

Author : Juan Bernet

Year : 2014

Juan Bernet introduced Inter Planetary File System which is a distributed peer-to-peer file system that connects the computing devices. The author designed the system in ways that it provides a high through-put content-addressed block storage model.

Title : Zerocash: Decentralized Anonymous Payments from Bitcoin.

Author : Eli Ben-Sasson

Year : 2014

Eli Ben-Sasson explains in detail about decentralized anonymous payments where the author points out some of the flaws in the bitcoin privacy guarantees. This paper fulfills that flaw by unlinking transaction from the payment's origin and it also reveals the payment destinations and the amount.

CHAPTER 3

AIM AND SCOPE OF THE PROJECT

3.1 AIM

The main aim of this project is to develop a system over blockchain which can store the users data in a decentralized database distributed across the peer topeer network.

3.2 PROJECT SCOPE

In this project, we propose a working decentralised database system using decentralized network and blockchain. The scope of the project is to show viability of decentralized database system using blockchain and smart contracts.

3.3 OBJECTIVE

To develop a system over Ethereum Blockchain which can store the users data in a decentralized database distributed across the peer to peer network. The specific objectives are as follows:

- To research about cryptography, P2P network, web technology and blockchain.
- To contribute in the active research on decentralized applications and cryptography.
- To develop a distributed cloud storage platform.

3.4 PROPOSED SYSTEM

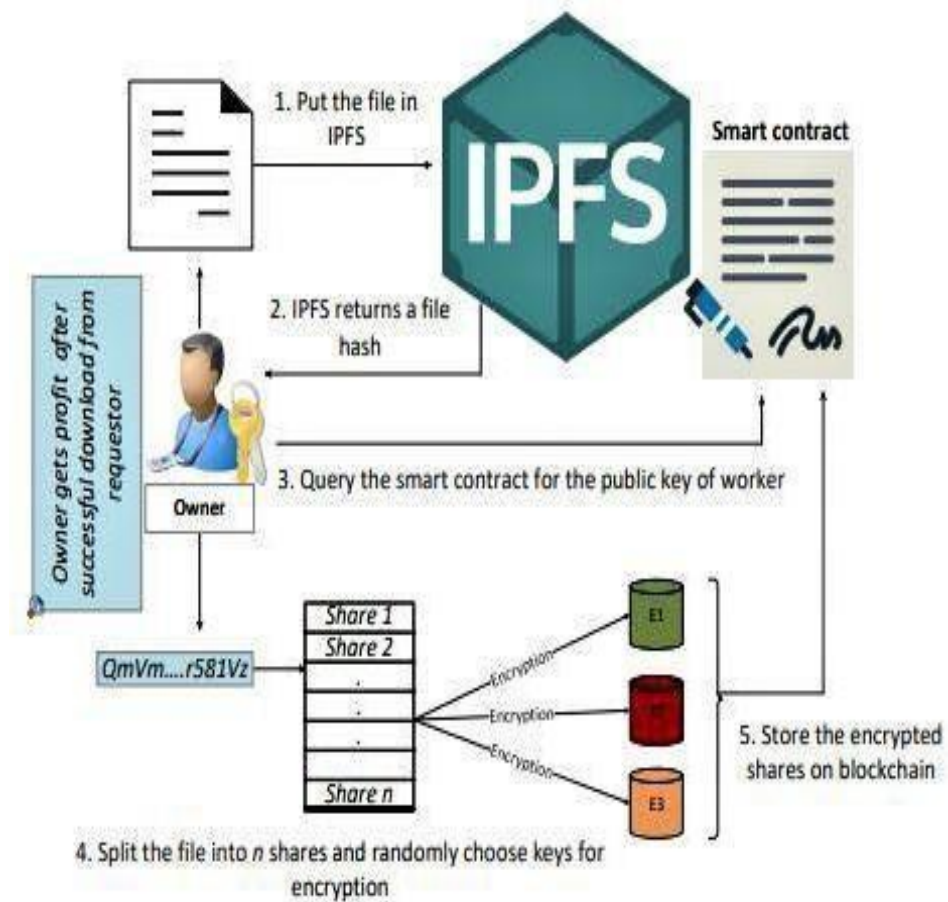


Fig. 3.1: Architecture Diagram

This project proposes a decentralized cloud storage using IPFS system where the data stored on this network is more secure and reliable than the conventional storage system. It is a simple and secure file storage system where the file is distributed across the network and when the file needs to be retrieved, decryption is performed on the data to restore in its original form. First, the data is selected and it is fed in the system where the data is first sharded into several number of blocks of bits. The number of shards is determined by the total number of nodes available in the network. The encryption is the most crucial point of this process where the data needs to be secure before distributing it across the network. After the data is separated into several shards which is considered as a data block is

hash encrypted so that the receiver of this block doesn't understand the information. This process happens until the entire data is hashed encrypted. Introduction of smart contracts is used here as an agreement between the client and the service provider where the user knows what data is stored and what is the price for the stored data. Smart contracts are the set of protocols in which the decentralized follows and the added advantage is it discloses any confusion between the users and the provider. Smart contracts are implemented in the truffle suite in this project. IPFS is a BitTorrent like network where number of nodes are connected on the network so that the data can be uploaded and downloaded easily. IPFS is used here to distribute the data blocks across the network where several nodes can store the data which is encrypted. The address of this network is available in the IPFS so that the transactions of this network can be watched using wallets like ganache which is used in this project. Ganache is a private wallet where the user can watch the transactions and information about the block is also available. The frontend of this project is created using React which uses JavaScript programming language.

CHAPTER 4

METHODOLOGY

4.1 MODULE DESCRIPTION

4.1.1 Web3j

Web3j is a lightweight, highly modular, reactive, type safe Java and Android library for working with Smart Contracts and integrating with clients (nodes) on the Ethereum network. This allows you to work with the Ethereum blockchain, without the additional overhead of having to write your own integration code for the platform. It has complete implementation of JSON-RPC client (Application Programming Interface) API over Hyper Text Transfer Protocol (HTTP) and Inter Process Communication (IPC). It has android compatible version and supports Infura. It has complete support to ethereum wallet.

4.1.2 Solidity

Solidity is a contract-oriented programming language for writing smart contracts. Solidity is a statically-typed programming language designed for developing smart contracts that run on the EVM. It was influenced by C++, Python and JavaScript and is designed to target the Ethereum Virtual Machine (EVM). With Solidity, developers are able to write applications that implement self-enforcing business logic embodied in smart contracts, leaving a nonrepudiable and authoritative record of transactions. Solidity support inheritance, including multiple inheritance with C3 linearization.

4.1.3 ReactJS

It is a JavaScript library for building user interfaces. It is maintained by Facebook and a community of individual developers and companies. React can be used as a base in the development of single-page or mobile applications. Complex React applications usually require the use of additional libraries for state management, routing, and interaction with an API. It makes developer painless to create

interactive UI. Properties commonly called props are passed from parent component to child. It also has feature of stateful components which can be passed to child components. React creates an in-memory data structure for virtual Document Object Model(DOM) and updates the browser DOM efficiently. Lifecycle in ReactJS are hooks which allows execution of code at set of points during component's lifetime.

4.1.4 Truffle Suite

Truffle is the most popular development framework for Ethereum with a mission to make your life a whole lot easier. Truffle takes care of managing your contract artifacts so you don't have to. Includes support for custom deployments, library linking and complex Ethereum applications. A world class development environment, testing framework and asset pipeline for blockchains using the Ethereum Virtual Machine (EVM), aiming to make life as a developer easier. Tighten the feedback loop between deployment, operation, and debugging; all within powerful development sandboxes. Share insights with the whole team.

4.1.5 Ganache

Quickly fire up a personal Ethereum blockchain which you can use to run tests, execute commands, and inspect state while controlling how the chain operates.

The main features of ganache are

- Blockchain log output - See the log output of Ganache's internal blockchain, including responses and other vital debugging information.
- Advanced mining control - Configure advanced mining with a single click, setting block times to best suit your development needs
- Ethereum blockchain - Byzantium comes standard, giving you the latest Ethereum features needed for modern dapp development.
- Built-in block explorer - Examine all blocks and transactions to gain insight about what's happening under the hood.

4.1.6 Node.js

Node.js is an open-source, cross-platform, back-end, JavaScript runtime environment that executes JavaScript code outside a web browser. Node.js is an open-source, cross-platform, back-end JavaScript runtime environment that runs on the V8 engine and executes JavaScript code outside a web browser. Node.js lets developers use JavaScript to write command line tools and for server-side scripting—running scripts server-side to produce dynamic web page content before the page is sent to the user's web browser. Consequently, Node.js represents a "JavaScript everywhere" paradigm, unifying web-application development around a single programming language, rather than different languages for server-side and client-side scripts.

4.1.7 Metamask

MetaMask is an extension for accessing Ethereum enabled distributed applications, or "Dapps" in your browser! The extension injects the Ethereum web3API into every website's javascript context, so that dapps can read from the blockchain. MetaMask is a cryptocurrency wallet used to interact with the Ethereum blockchain. MetaMask allows users to store and manage account keys, broadcast transactions, send and receive Ethereum-based cryptocurrencies and tokens, and securely connect to decentralized applications through a compatible web browser or the mobile app's built-in browser.

4.1.8 IPFS Infura

Infura provides secure, reliable, scalable, and easy to use APIs to access the Ethereum network and the IPFS. In many cases, using this API this is preferable to embedding IPFS directly in your program it allows you to maintain peer connections that are longer lived than your app and you can keep a single IPFS node running instead of several if your app can be launched multiple times.

4.1.9 Ethereum

Ethereum, and its provision of smart contracts provided real functionality even if the results of its open system are dubious. Ethereum is a decentralized, open-source blockchain featuring smart contract functionality. It is the native cryptocurrency of the platform. It is the second-largest cryptocurrency by market capitalization, after Bitcoin. Ethereum is the most actively used blockchain.

4.1.10 JavaScript

JavaScript often abbreviated as JS, is a programming language that conforms to the ECMAScript specification. JavaScript is high-level, often just-in-time compiled, and multi-paradigm. It has curly-bracket syntax, dynamic typing, prototype-based object-orientation, and first-class functions. Alongside HTML and CSS, JavaScript is one of the core technologies of the World Wide Web. JavaScript enables interactive web pages and is an essential part of web applications. The vast majority of websites use it for client-side page behavior and all major web browsers have a dedicated JavaScript engine to execute it. As a multi-paradigm language, JavaScript supports event-driven, functional, and imperative programming styles. It has application programming interfaces (APIs) for working with text, dates, regular expressions, standard data structures, and the Document Object Model (DOM). The ECMAScript standard does not include any input/output (I/O), such as networking, storage, or graphics facilities. In practice, the webbrowser or other runtime system provides JavaScript APIs for I/O. JavaScript engines were originally used only in web browsers, but they are now core components of other runtime systems, such as Node.js and Deno. These systems are used to build servers and are also integrated into frameworks, such as Electron and Cordova, for creating a variety of applications. Although there are similarities between JavaScript and Java, including language name, syntax, and respective standard libraries, the two languages are distinct and differ greatly in design.

The main features of JavaScript are:

- Imperative and structured
- Weakly typed
- Dynamic
- Object-orientation (prototype-based)
- Functional
- Delegative
- Miscellaneous
- Vendor-specific extensions

4.1.11 CSS

Cascading Style Sheets (CSS) is a style sheet language used for describing the presentation of a document written in a markup language such as HTML. CSS is a cornerstone technology of the World Wide Web, alongside HTML and JavaScript. CSS is designed to enable the separation of presentation and content, including layout, colors, and fonts. This separation can improve content accessibility, provide more flexibility and control in the specification of presentation characteristics, enable multiple web pages to share formatting by specifying the relevant CSS in a separate .css file which reduces complexity and repetition in the structural content as well as enabling the .css file to be cached to improve the page load speed between the pages that share the file and its formatting. Separation of formatting and content also makes it feasible to present the same markup page in different styles for different rendering methods, such as on-screen, in print, by voice (via speech-based browser or screen reader), and on Braille-based tactile devices. CSS also has rules for alternate formatting if the content is accessed on a mobile device. The name cascading comes from the specified priority scheme to determine which style rule applies if more than one rule matches a particular element. This cascading priority scheme is predictable.

Internet media type (MIME type) `text/css` is registered for use with CSS by RFC

2318 (March 1998). The W3C operates a free CSS validation service for CSS documents. In addition to HTML, other markup languages support the use of CSS including XHTML, plain XML, SVG, and XUL. There is no single, integrated CSS4 specification, because the specification has been split into many separate modules which level independently. Modules that build on things from CSS Level 2 started at Level 3. Some of them have already reached Level 4 or are already approaching Level 5. Other modules that define entirely new functionality, such as Flexbox, have been designated as Level 1 and some of them are approaching Level 2. The CSS Working Group sometimes publishes "Snapshots", a collection of whole modules and parts of other drafts that are considered stable enough to be implemented by browser developers. So far, five such "best current practices" documents have been published as Notes, in 2007, 2010, 2015, 2017, and 2018. Since these specification snapshots are primarily intended for developers, there has been growing demand for a similar versioned reference document targeted at authors, which would present the state of interoperable implementations as meanwhile documented by sites like Can I Use...and the Mozilla Developer Network. A W3C Community Group has been established in early 2020 in order to discuss and define such a resource. The actual kind of versioning is also up to debate, which means that the document once produced might not be called "CSS4".

4.1.12 HTML

The HyperText Markup Language, or HTML(HyperText Markup Language) is the standard markup language for documents designed to be displayed in a web browser. It can be assisted by technologies such as Cascading Style Sheets (CSS) and scripting languages such as JavaScript. Web browsers receive HTML documents from a web server or from local storage and render the documents into multimedia web pages. HTML describes the structure of a web page semantically and originally included cues for the appearance of the document. HTML elements are the building blocks of HTML pages. With HTML

constructs, images and other objects such as interactive forms may be embedded into the rendered page. HTML provides a means to create structured documents by denoting structural semantics for text such as headings, paragraphs, lists, links, quotes and other items. HTML elements are delineated by tags, written using angle brackets. Tags such as `` and `<input />` directly introduce content into the page. Other tags such as `<p>` surround and provide

information about document text and may include other tags as sub-elements.

Browsers do not display the HTML tags, but use them to interpret the content of the page. HTML can embed programs written in a scripting language such as JavaScript, which affects the behavior and content of web pages. Inclusion of CSS defines the look and layout of content. The World Wide Web Consortium (W3C), former maintainer of the HTML and current maintainer of the CSS standards, has encouraged the use of CSS over explicit presentational HTML. An HTML Application (HTA; file extension ".hta") is a Microsoft Windows application that uses HTML and Dynamic HTML in a browser to provide

the application's graphical interface. A regular HTML file is confined to the security model of the web browser's security, communicating only to web servers and manipulating only web page objects and site cookies. An HTA runs as a fully trusted application and therefore has more privileges, like creation/editing/removal of files and Windows Registry entries. Because they operate outside the browser's security model, HTAs cannot be executed via HTTP, but must be downloaded (justlike an EXE file) and executed from local file system.

4.2 SOFTWARE DESCRIPTION

4.2.1 BUILDING PEER TO PEER NETWORK

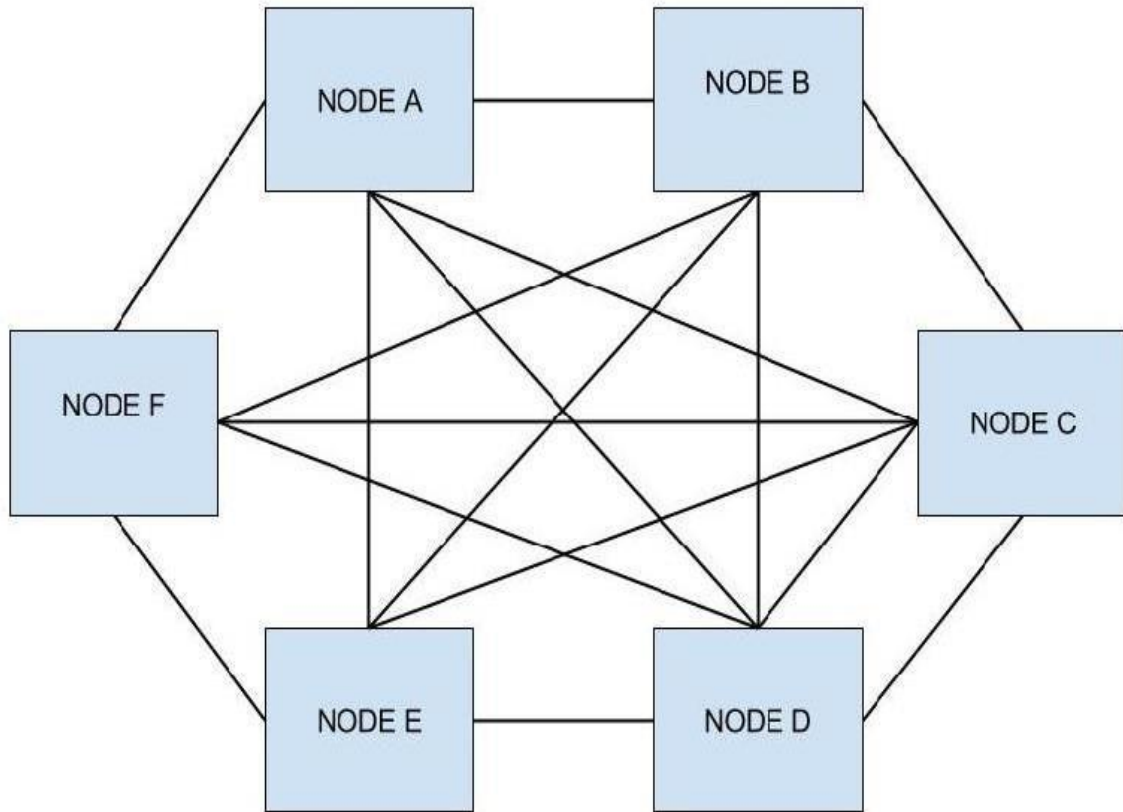


Fig. 4.1: P2P Network

Peer to Peer network is the distributed network where each node in the network communicates with each other directly or through a series of channels via other nodes. There is no client server to access the resource. Each node will act both as a host or a client as needed. There is no any Central server for controlling the system flow and other nodes.

4.3 NODE SETUP

Node is one of the member of P2P system which is willing to provide store for the clients in return to tokens.

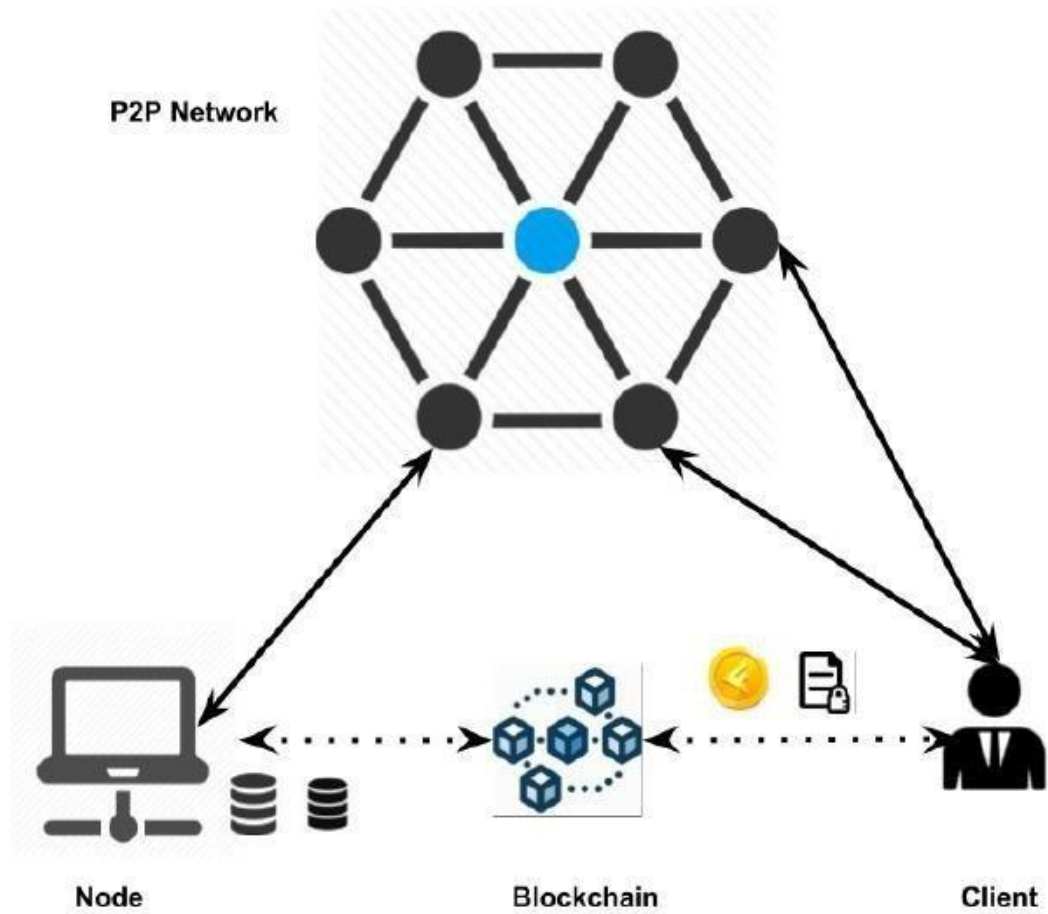


Fig. 4.2: Node Workflow

4.3.1 Node Operations

Node is fully responsible for handling the client's data. It can view the encrypted data send by client which is stored in node's storage. Node can list all the data of the client and organize it.

4.3.2 Motivation for Node

In our system, node is getting paid for storing the client's data by token of our system. According to the agreement between node and client, the node gets payment as file is downloaded by the client or duration of agreement finishes.

4.4 CLIENT SETUP

4.4.1 Communication with Node

File Processing is the main function of our app is to encrypt/decrypt the file, split it into user defined parts and upload it to nodes. Like that, download it from nodes, merge the chunk to single file and then decrypt to original file. Figure given below will give more clear idea about file processing in our system.

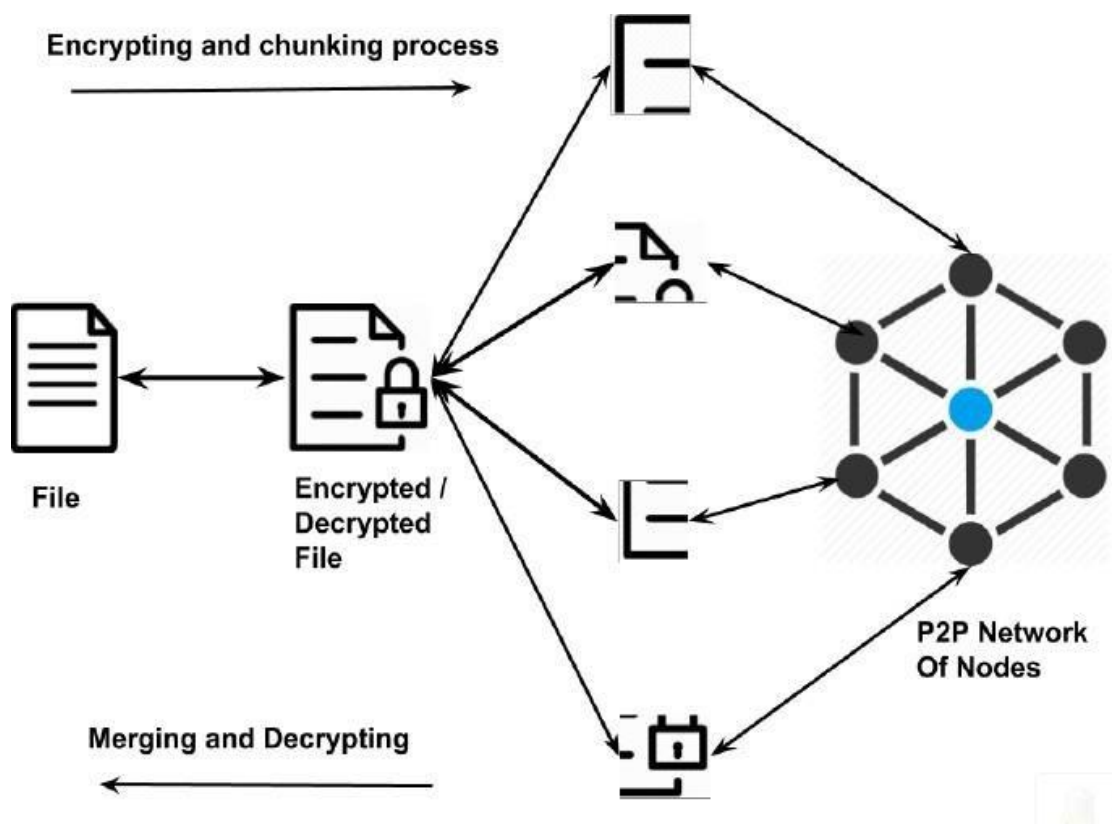


Fig. 4.3: File Processing

4.4.2 AES Encryption and Decryption

Using Random AES Key Generator, random key for AES gets generated. Like that, Random IV spec generator generates Random IV which is required during AES encryption. 24 byte AES key along with 16 byte Random IV encrypts the file. AES key is further encrypted with master key. Therefore, our encrypted file consist first 40 byte of key as header and remaining byte as encrypted data of the file. Finally, the encrypted file gets chunked into user defined number and gets transferred to network. Like that after downloading the chunks from network, all the parts get merged into encrypted file. First 40 bytes is separated as header and remaining as file cipher. Among 40 bytes, 24 bytes is seperated and AES key is derived. After that remaining 16 bytes Random IV and AES key is used to decrypt the file which gives original file as an output.

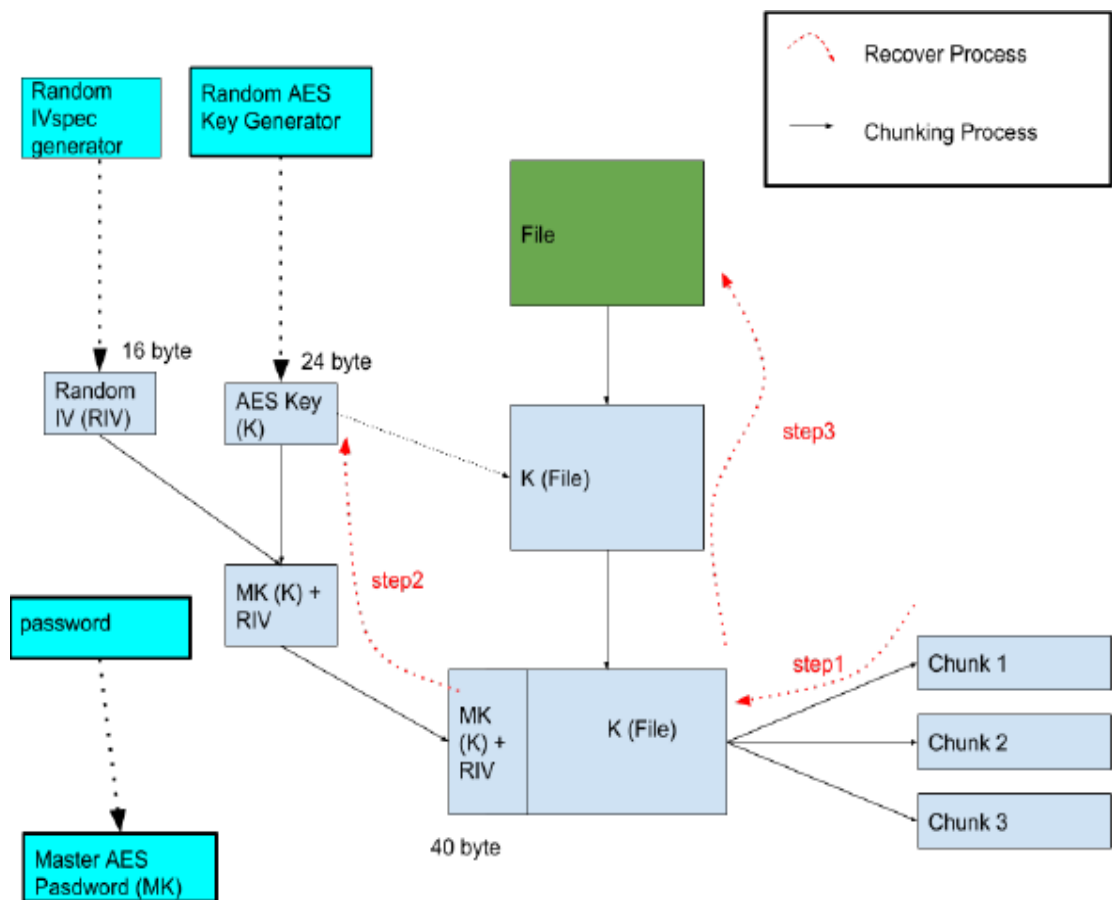


Fig. 4.4: Secured File Storage Protocol

4.4.3 File Splitting/Merging

The encrypted file is splitted into number of nodes selected and transferred to each node via network. Like that, during download those separate chunks from different nodes get merged into single cipher file.

4.4.4 Hash Encryption

Encryption is a two-way function; what is encrypted can be decrypted with the proper key. Hashing, however, is a one-way function that scrambles plain text to produce a unique message digest. With a properly designed algorithm, there is no way to reverse the hashing process to reveal the original password. An attacker who steals a file of hashed passwords must then guess the password. Here's how it works: A user enters a password and an ID in a browser and sends it (preferably over a secure link) to the authentication server. The server uses the ID to look up the associated message digest. The password submitted by the user is then hashed with the same algorithm, and if the resulting message digest matches the one stored on the server, it is authenticated. In this process the server does not store or need to see plain-text passwords. Stealing hashed files does the attacker little good because the attacker cannot reverse the hashing process. But because people rarely use completely random passwords there is a trick that can be used to help guess the passwords in the file. An attacker can run a collection of a million or so commonly used passwords through a hashing algorithm and get a list called a rainbow table of associated message digests for these passwords. It is child's play for a computer to compare a file of stolen password hashes against a rainbow table. For every match, the table will show the password for that hash. The protection against this is to salt the hash: Add a random number to each password before it is hashed. The resulting message digest is the product of both the password and the salt value and will not match anything on the rainbow table. Of course, the attacker can always try adding random values to common passwords to find a matching hash, but now the difficulty of guessing the password makes it impractical. The return on investment of such a process is so low that a stolen file of properly hashed and salted passwords is essentially worthless.

4.5 BLOCKCHAIN INTEGRATION

4.5.1 Blockchain

Blockchain is the growing list of records called blocks, containing structured information linked using the art of cryptography distributed globally . Each block in blockchain contains the cryptographic hash of previous block along with timestamp and data which typically varies with use-cases. Merkle trees are the fundamental part of blockchain. Every block contains the block header which is outcome of recursive cryptographic hashes of all the data nodes or transaction from bottom to up approach. During hash generation, the order of data matters for the final hash of the block. If single detail of transactions or order of transaction changes then changes the merkle hash. Therefore, Merkle Root summarizes all of the data in the related transactions, and is stored in the block header. This feature makes blockchain resistance to modification which is considered secure by design. A database is a collection of information that is stored electronically on a computer system. Information, or data, in databases is typically structured in table format to allow for easier searching and filtering for specific information. Large databases achieve this by housing data on servers that are made of powerful computers. These servers can sometimes be built using hundreds or thousands of computers in order to have the computational power and storage capacity necessary for many users to access the database simultaneously. While a spreadsheet or database may be accessible to any number of people, it is often owned by a business and managed by an appointed individual that has complete control over how it works and the data within it. Blockchain is P2P network which relies on protocol for inter-node communication and validating the blocks. Blockchain was invented by Satoshi Nakamoto in 2008 to serve as the public transaction ledger of the cryptocurrency bitcoin. Blocks in blockchain are the holder of valid transactions that are hashed and encoded in Merkle tree. Every block contains the cryptographic hash of previous blocks along with its own data which therefore forms the chain. This iterative mechanism in each block conforms the integrity of previous block all the way back to genesis block. Block time is the average time it takes in the network to generate 1 extra block in blockchain. By the time block is

generated, the data of that block is verified. This means lesser the block time, faster the transactions. A hard fork is a rule change such that the software validating according to the old rules will see the blocks produced according to the new rules as invalid. Storing data in P2P network allows Blockchain to eliminate the pitfalls of centralization. There will be no central point vulnerability, no center point of failure in blockchain. It is open to public which makes it more user-friendly than traditionally owned records. Being permissionless and open, there is no need to guard against bad actors. Ethereum is an open-source, public, blockchain-based distributed computing platform and operating system featuring smart contract (scripting) functionality.

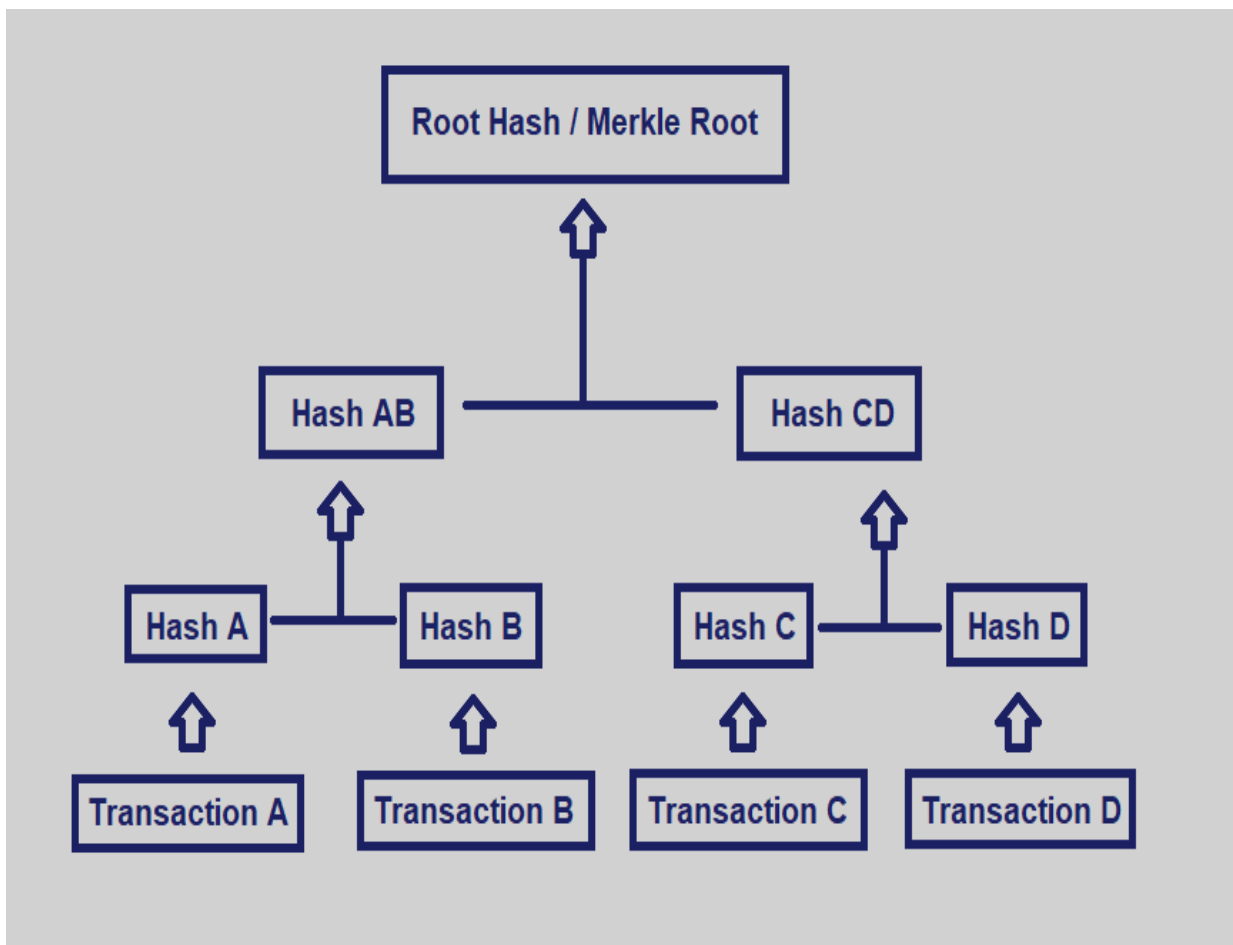


Fig. 4.5: Merkle Tree

Ether is a cryptocurrency whose blockchain is generated by the Ethereum platform. Ethereum provides a decentralized Turing-complete virtual machine, the Ethereum Virtual Machine (EVM), which can execute scripts using an international network of public nodes. "Gas", an internal transaction pricing mechanism, is used to mitigate spam and allocate resources on the network. Ethereum addresses are composed of the prefix "0X" a common identifier for hexadecimal, concatenated with the rightmost 20 bytes of the Keccak-256 (SHA-3) hash (big endian) of the ECDSA (Elliptic Curve Digital Signature Algorithm) public key. Smart Contracts

Ethereum's smart contracts are based on different computer languages, which developers use to program their own functionalities. Smart contracts are high-level programming abstractions that are compiled down to EVM bytecode and deployed to the Ethereum blockchain for execution. They can be written in Solidity (a language library with similarities to C and JavaScript), Serpent (similar to Python, but deprecated), LLL (a low-level Lisp-like language), and Mutan (Go-based, but deprecated). There is also a research-oriented language under development called Viper (a strongly-typed Python-derived decidable language).

ERC20 Token

ERC-20 is a technical standard used for smart contracts on the Ethereum blockchain for implementing tokens. ERC stands for Ethereum Request for Comment, and 20 is the number that was assigned to this request. The clear majority of tokens issued on the Ethereum blockchain are ERC-20 compliant. The ERC20 token standard describes the functions and events that an Ethereum token contract has to implement.

4.5.2 Smart Contract Development

Smart contract acts as an agreement between client and postman during the transaction of data. Therefore, we have tried to ensure (look) from both parties during its development. We have tried to keep as less data as possible in blockchain network without compromising services. The key to these contracts is the decentralised network known as blockchain. Smart contracts use blockchain technology to verify, validate, capture and enforce agreed-upon terms between multiple parties.

Smart contracts on the blockchain allow for transactions and agreements to be carried out among anonymous parties without the need for a central entity, external enforcement, or legal system. The transactions are transparent, irreversible, and traceable. Blockchain is the perfect environment for smart contracts, as all the data stored is immutable and secure. The data of a smart contract is encrypted and exist on a ledger, meaning that the information recorded in the blocks can never be lost, modified, or deleted. Smart Contract act as aglobal database which gets deployed in the blockchain.

There are 2 roles in our contract

Node : stores client data,earns tokens .

Client : stores data on node, pays tokens.

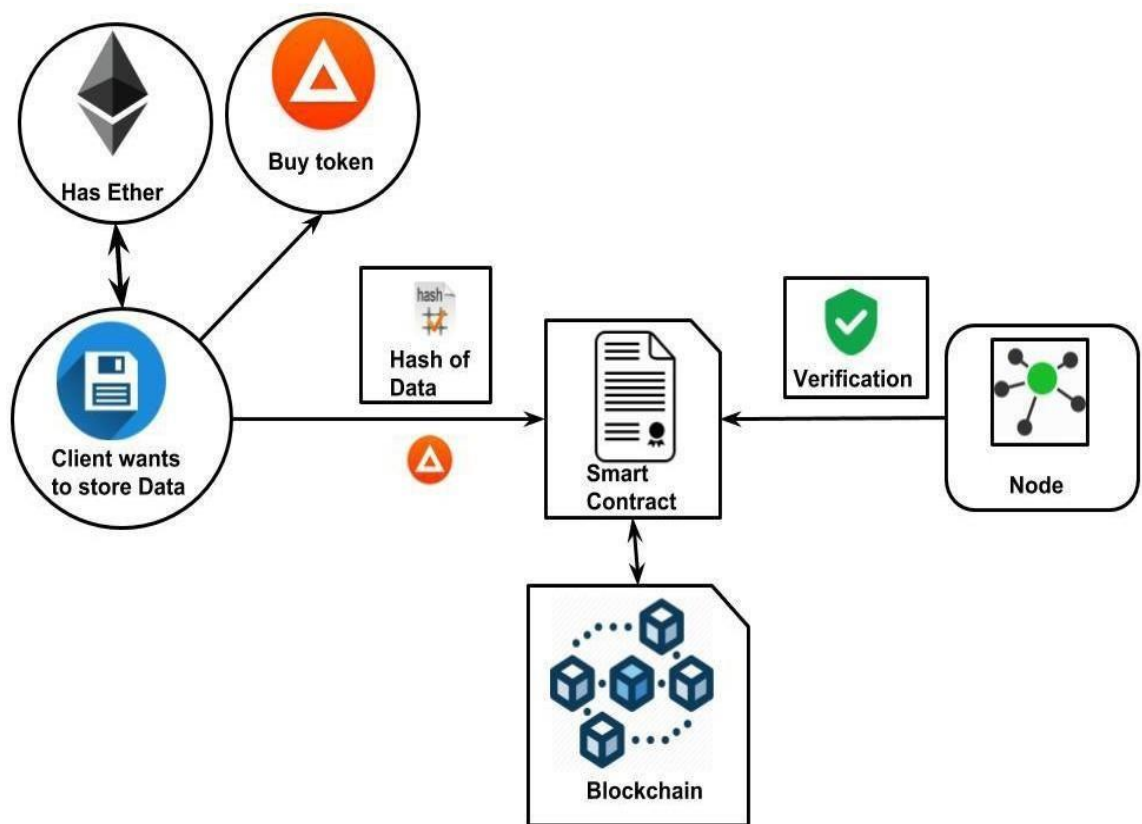


Fig. 4.6: Smart Contract Flow Diagram

User having ether in their ethereum account can interact with the contract. Our system has its own ERC20 token which can be bought by ether. We have defined our own exchange rate. In order to store data in our system, client must pay agreed amount of tokens to the node as specified during node-client agreement in smart contract. Client gets facility to store data on node's storage. During this process, client and postman come on agreement based on storage size, bandwidth, time of storage and token amount. Once, agreement is done, all details of agreement between client, node and associated data chunk is written smart contract which gets deployed in Blockchain.

4.5.3 File Details Record

As client's chunked file gets stored in node's storage, there should be proper tracking mechanism of file along with its chunk, associated node and client in Smart Contract. Our system should ensure client for file retrieval. Smart contract tracks all the record associated with client, file and node. Usually data are stored as key-value data structure.

- Address with FileDetail Array It tracks the owner of file along with array of owned file details by each individual. File Details contains the hash of file and its filename.
- File Hash with Chunk Hash Array It tracks the chunk of file array associated with the main file so that when file hash is known, its chunk can be traced.
- Chunk Hash with File hash Every node doesn't have file hash where as smart contract needs file hash for getting chunk info. This data gap is fulfilled by this data structure where once chunk hash is known, file hash can be known immediately.
- Chunk hash to index/File hash to index Looping in smart contract is very expensive. Therefore for every index of chunk hash in array is mapped with its chunk hash so that once chunk hash is known its index can be immediately known. Like that, client can have number.

- ChunkIndex to Download index All the chunk uploaded to node will not be downloaded. Therefore, this data structure tracks the chunk data which have download request so that payment agreement can be done.

4.5.4 Token Transfer

During our contract deployment, we created fixed amount of token which serves as payment mechanism in our system. Tokens can be exchanged with ether token. For buying tokens, contract gives use option to buy it with ethers. Once tokens are purchased, users can use our system properly. We have made a system of locking the balance of 2 agreed parties so that both parties will perform their duty. Once the agreement is made between client and node, the specified amount of tokens from their balance is transferred to their lock balance. Here, lock balance means that those deposited tokens are restricted to use for other purposes. Those tokens get locked until agreement is not completed. When correct file is provided by node to client which gets verified by blockchain, then payment is provided to node from lock balance. Here, tokens of node is also transferred to lock balance so that if node doesn't behaves what is agreed in agreement then, client is compensated with those tokens. Once the agreement is done, fund is not returned back.

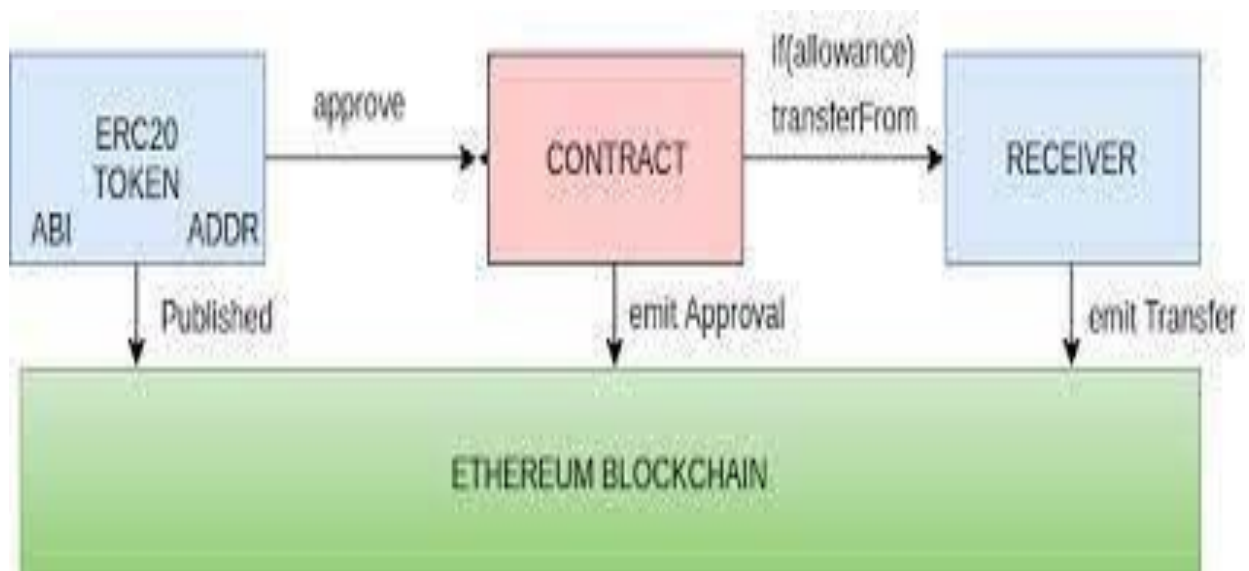


Fig. 4.7: Token transfer

4.6 SYSTEM DEVELOPMENT METHODOLOGY

4.6.1 Software Development Approach

Making huge and dynamic system using traditional approach such as waterfall model of software development cost more time and manpower. Therefore to meet the requirements of the system with more flexibility and timely delivery, we have chosen Scrum methodology under the Agile Development method. Instead of providing complete, detailed descriptions of how everything is to be done on a project, much of it was left up to the project development team. The scrum team is selforganizing in that there is no overall team leader who decides which person will do which task or how a problem will be solved. So, each time every team member tried to solve problems and find solutions. And in Scrum, a team is cross functional, meaning everyone is needed to take a feature from idea to implementation. Project progress were shown via a series of sprints. In keeping with an agile methodology, sprints are timeboxed to no more than a month long, most commonly two weeks. We met at every start of sprint and figure out each individual commits of task and created backlog to keep track of work. We commonly had daily meetings during college team for discussing project's problems and solutions. This helped all team member to be fully synchronized which is one of the main benefit of Scrum methodology. We made sprint burndown chart and release burndown chart which helped us to know remaining work and track the overall project schedule. Hence scrum is adaptive, has small repeating cycles and there is short-term planning with constant feedback, inspection and adaption and is therefore chosen as the software development methodology.



Fig. 4.8: Scrum methodology

4.6.2 Requirement Analysis

The functional and non-functional requirements of this project are as listed below

Function Requirement:

- The system shall encrypt, decrypt the data of user.
- The system shall split and merge the file.
- The system shall built P2P network and allow clients to connect.
- The system shall read and write data from Rinkeby Test Net.

Non Function Requirement:

- The system must ensure data retrieval of clients.
- The system must allow client to store all types of files.
- The system should be dynamic enough to easily adapt with increasing number of data.

CHAPTER 5

RESULTS AND DISCUSSION

5.1 CLIENT APPLICATION

Based on the observations of the system, we came to the following observations and evaluations.

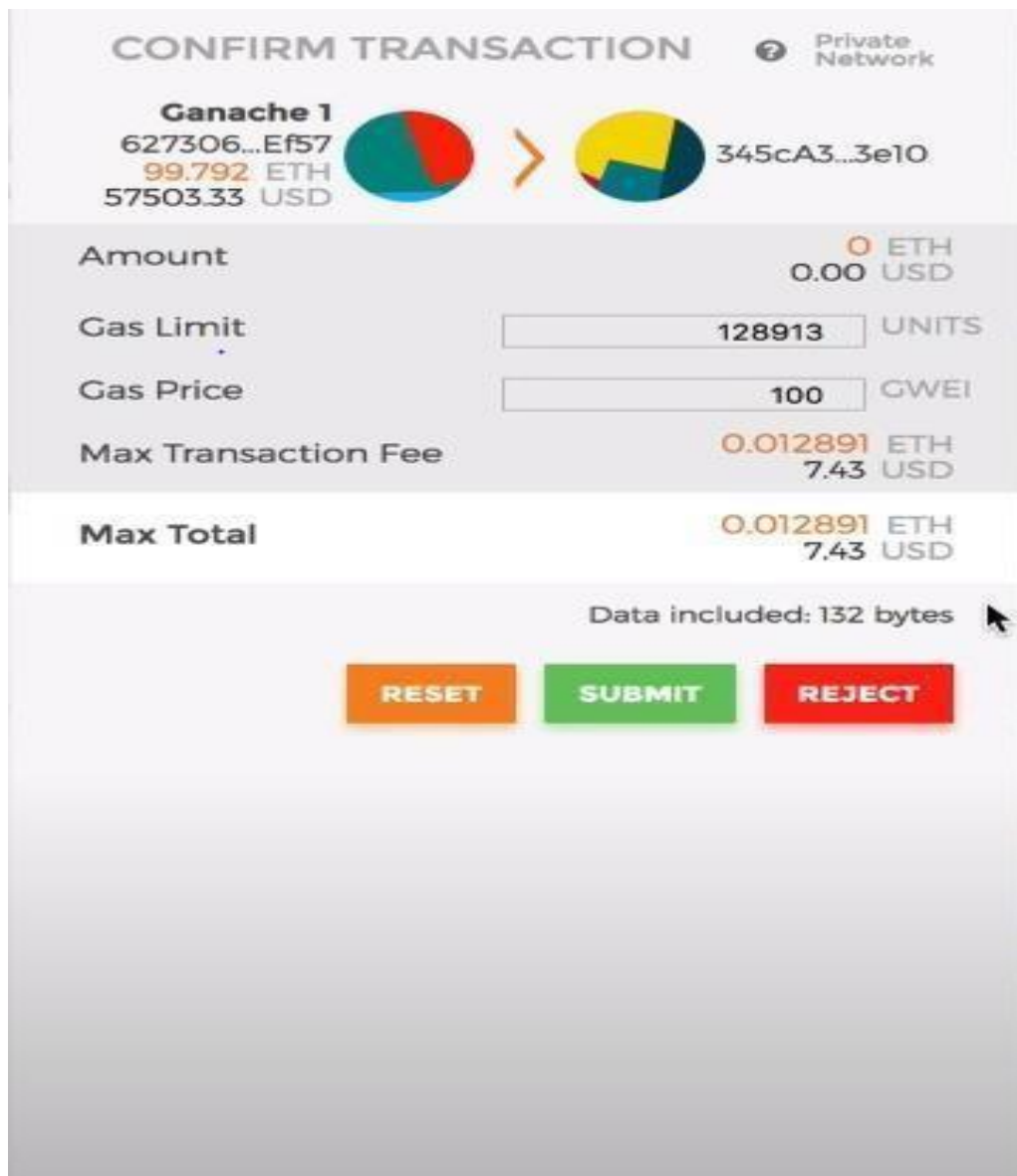
- The file storage service works well through all phases of the feature: Selection, Encryption, Splitting, Upload, Listing, Download, Merging, Decryption. Every record metadata is store in Smart Contract.
- The task is significantly smoother when a copy of all transaction is stored in a local database and reading from contract is done for verification check only during conflicting situation. This makes the user experience smooth.
- The node can now receive a real time notification of every details. The node app is more user friendly and easily understandable about the existing real time system.
- The file chunking-merging and encryption-decryption is highly dependent upon the two factors: File Size and Processor speed. The time of any of the file operation is linearly dependent upon the file size. So, uploading huge size file produces significant load on the Android Thread and in some caseis forcefully killed by the system causing system crash.
- A webapp of the client side can be more impressive and easy to use for loading contract, wallet, handling heavy processing task is more easier and fast.

5.1.1 RESULT OF NODE APP

- The node interface communicates with the :
- The interface shows how much storage space the files have taken, number of files the node has received and how many files in total clients have downloaded.
- List of all the chunks received as file can be watched in this panel. The list

shows hash, creation and expiry date along with the number of download count for individual files. Events like receiving files are shown in real time using websockets.

- This interface shows information about a deployed node with its public key, node id, private key. Other nodes connected or discovered by this node is shown with their node Identity, IP address and port.



The image shows a Metamask transaction confirmation interface. At the top, it says "CONFIRM TRANSACTION" and "Private Network". Below this, it shows the transaction details for "Ganache 1" with a public key "627306...Ef57" and a balance of "99.792 ETH" and "57503.33 USD". A large orange arrow points to the right, indicating the transaction direction. On the right, it shows the destination address "345cA3...3e10". Below this, the transaction details are listed: "Amount" is "0.00 ETH" and "0.00 USD"; "Gas Limit" is "128913" "UNITS"; "Gas Price" is "100" "GWEI"; "Max Transaction Fee" is "0.012891 ETH" and "7.43 USD"; and "Max Total" is "0.012891 ETH" and "7.43 USD". At the bottom, it says "Data included: 132 bytes" and there are three buttons: "RESET", "SUBMIT", and "REJECT".

Field	Value	Unit
Amount	0.00	ETH / USD
Gas Limit	128913	UNITS
Gas Price	100	GWEI
Max Transaction Fee	0.012891	ETH / 7.43 USD
Max Total	0.012891	ETH / 7.43 USD

Fig. 5.1: Metamask

MetaMask is a cryptocurrency wallet used to interact with the Ethereum blockchain. MetaMask allows users to store and manage account keys, broadcast transactions, send and receive Ethereum-based cryptocurrencies and tokens, and securely connect to decentralized applications through a compatible web browser or the mobile app's built-in browser.

The screenshot displays the MetaMask interface. At the top, there are navigation tabs: ACCOUNTS (selected), BLOCKS, TRANSACTIONS, and LOGS. A search bar is located to the right of these tabs. Below the navigation bar, a status bar shows various network metrics: CURRENT BLOCK (4), GAS PRICE (20000000000), GAS LIMIT (6721975), NETWORK ID (5777), RPC SERVER (HTTP://127.0.0.1:7545), and MINING STATUS (AUTOMINING). Below the status bar, the MNEMONIC (candy maple cake sugar pudding cream honey rich smooth crumble sweet treat) and HD PATH (m/44'/60'/0'/0/account_index) are displayed. The main section shows a list of accounts with their addresses, balances, transaction counts, and indices.

ADDRESS	BALANCE	TX COUNT	INDEX
0x627306090abaB3A6e1400e9345bC60c78a8BEf57	99.94 ETH	4	0
0xf17f52151EbEf6C7334FAD080c5704D77216b732	100.00 ETH	0	1
0xC5fdf4076b8F3A5357c5E395ab970B5B54098Fef	100.00 ETH	0	2
0x821aEa9a577a9b44299B9c15c88cf3087F3b5544	100.00 ETH	0	3
0x0d1d4e623D10F9FBA5Db95830F7d3839406C6AF2	100.00 ETH	0	4

Fig. 5.2: Ganache

See the log output of Ganache's internal blockchain, including responses and other vital debugging information. Configure advanced mining with a single click, setting block times to best suit your development needs Byzantium comes standard, giving you the latest Ethereum features needed for modern dapp development. Examine all blocks and transactions to gain insight about what's happening under the hood.

Upload file to IPFS

FileName

Upload File No file chosen

Fig. 5.3: File uploading

With decentralized cloud storage, end-to-end encryption is standard on every file- each file is encrypted on a user's computer before it's uploaded, broken into pieces, and then spread out to uncorrelated Nodes across our network. Plus, centralized cloud storage costs a lot more than our decentralized network. IPFS is used here to distribute the data blocks across the network where several nodes can store the data which is encrypted. is available in the IPFS so that the transactions of this network can be watched using wallets like ganache which is used in this project.

Ganache is a private wallet where the user can watch the transactions and information about the block is also available. The frontend of this project is created using React which uses JavaScript programming language.

File Upload

Name : Sample

LINK : [QmcPXZAmAXyDjUqcveb6K9pfjTZ3XBGGAhK3MBFENtsF1n](#)

Fig. 5.4: File upload and retrieving screenshot

The contents of the retrieved block are decrypted and merged into the original file where the content is checked whether any loss of data is recorded. This type of cloud storage is secure and is cost effective compared to the centralized server. Each node in this network acts as a server and data is shared among them without knowing what the data contains and whose is it.

CHAPTER 6

CONCLUSION AND FUTURE SCOPE

6.1 CONCLUSION

The project consists of a P2P network where a node can join the network and provide the storage services for the client. The system uses several Cryptography and Network algorithms and provides two major services to client : Secure File Storage and Secret Sharing. The agreement between the parties are bound by Smart Contract and uses ERC20 token as a value for service.

The two layers of the system can also be implemented separately as components for different use cases. The project was completed with a exciting exploration and research in Cryptography and Blockchain field. There is always room for the improvements in any projects. The project can be further enhanced including following features:

- Currently, we have only mobile app for clients to use. Therefore, web app can be made for client purposes.
- So that large sized files can be easily encrypted, splitted and merged.
- Compensation mechanism for faulty party can be further added in our system.
- The one approach for this could be the introduction of a central authority.
- It can also be achieved by using a third auditor node selected at random from the network.
- Algorithms and the protocols used in the cryptography processes could further be fine tuned.

6.2 FUTURE WORK

The future model can be improved with better encryption algorithms and this domain is still under development so better data distribution algorithms can be used. Blockchain-based cloud computing doesn't only decentralize data storage. More than ever, this technology is being used as part of data logistics platforms. Many authorized users are allowed access to the data at once and can interact, interpret, and change it as they see fit. This type of cloud computing solution allows employees to work together virtually and securely to analyze and maintain information. The virtualization of contractual agreements and other exchanges is also easier with the use of blockchain, leading to a wide increase in usage over many industries.

REFERENCES

- [1] Crosby.M, P. Pattanayak, S. Verma, and V. Kalyanaraman, "Blockchain technology: Beyond bitcoin," *Applied Innovation*, vol. 2, pp. 6–10, 2016.
- [2] David Vorick et al. *Sia: Simple Decentralized Storage*. 2014.
- [3] Eli Ben-Sasson et al. *Zerocash: Decentralized Anonymous Payments from Bitcoin*.2014.
- [4] Jin Sun et al. *Blockchain-Based Secure Storage and Access Scheme for Electronic Medical Records in IPFS*.IEEE.2014.
- [5] Juan Bernet.*IPFS - Content Addressed, Versioned, P2P File System*.2014.
- [6] King.S - *Ppcoin: Peer-to-peer crypto-currency with proof-of-stake*.2014.
- [7] McConaghy.T, R. Marques, A. Müller, D. De Jonghe, T. McConaghy, G. McMullen,R. Henderson, S. Bellemare, and A. Granzotto, "Bigchaindb: a scalable blockchain database," white paper, BigChainDB, 2016
- [8] Nakamoto S. *Bitcoin: A peer-to-peer electronic cash system*. Consulted, 2008.
- [9] Shawn Wilkinson et al. *Storj: A Peer-to-Peer Cloud Storage Network*. 2016.
- [10] Vitalik Buterin.*Ethereum: A next-generation smart contract and decentralized application platform*.2013
- [11] Wilkinson.S, T. Boshevski, J. Brandoff, and V. Buterin, "Storj a peer-to-peer cloud storage network," 2014.
- [12] Yan Zhu et al. *Blockchain-based Decentralized Storage Scheme*.IOP publishing.1237.4.2019.

APPENDIX

A) SOURCE CODE

Solidity:

1. Storage contract:

```
pragma solidity 0.4.24;

contract SimpleStorage {

    string ipfsHash;

    function set(string x) public {

        ipfsHash = x;

    }

    function get() public view returns (string) {

        return ipfsHash;

    }

}
```

2. Migration:

```
pragma solidity ^0.4.2;

contract Migrations {

    address public owner;

    uint public last_completed_migration;

    modifier restricted() {

        if (msg.sender == owner) _;

    }

    function Migrations() public {

        owner = msg.sender;

    }

}
```



```

function setCompleted(uint completed) public restricted {
last_completed_migration = completed;}

function upgrade(address new_address) public restricted {
    Migrations upgraded = Migrations(new_address);
    upgraded.setCompleted(last_completed_migration);
}
}

```

Reactjs:

App.js:

```

import React, { Component } from 'react'
import SimpleStorageContract from '../build/contracts/SimpleStorage.json'
import getWeb3 from '../utils/getWeb3'
import ipfs from './ipfs'
import './css/oswald.css'
import './css/open-sans.css'
import './css/pure-min.css'
import './App.css'
class App extends Component {
  constructor(props) {
    super(props)
    this.state = {
      ipfsHash: "",
      web3: null,
      buffer: null,
      account: null
    }
  }
}

```

```

}

this.captureFile = this.captureFile.bind(this);

this.onSubmit = this.onSubmit.bind(this);

}

componentWillMount() {
  // Get network provider and web3 instance.
  // See utils/getWeb3 for more info.

  getWeb3
    .then(results => {
      this.setState({
        web3: results.web3
      })
      // Instantiate contract once web3 provided.
      this.instantiateContract()
    })
    .catch(() => {
      console.log('Error finding web3.')
    })
}

instantiateContract() {
  /*
   * SMART CONTRACT EXAMPLE
   *
   * Normally these functions would be called in the context of a
   * state management library, but for convenience I've placed them here.
   */

```

```

const contract = require('truffle-contract')

const simpleStorage = contract(SimpleStorageContract)
simpleStorage.setProvider(this.state.web3.currentProvider)

// Get accounts.

this.state.web3.eth.getAccounts((error, accounts) => {
simpleStorage.deployed().then((instance) => {
    this.simpleStorageInstance = instance
    this.setState({ account: accounts[0] })

    // Get the value from the contract to prove it worked.
    return this.simpleStorageInstance.get.call(accounts[0])
}).then((ipfsHash) => {
    // Update state with the result.
    return this.setState({ ipfsHash })
})
})
}

captureFile(event) {
    event.preventDefault()

    const file = event.target.files[0]

    const reader = new window.FileReader()
    reader.readAsArrayBuffer(file)

    reader.onloadend = () => {
        this.setState({ buffer: Buffer(reader.result) })
        console.log('buffer', this.state.buffer)
    }
}

```

```

}

onSubmit(event) {
  event.preventDefault()
  ipfs.files.add(this.state.buffer, (error, result) => {
    if(error) {
      console.error(error)
      return}

    this.simpleStorageInstance.set(result[0].hash, { from: this.state.account
  }).then((r) => {
    return this.setState({ ipfsHash: result[0].hash })
    console.log('ipfsHash', this.state.ipfsHash)
  })
})
}

render() {
  return (
    <div className="App">
      <nav className="navbar pure-menu pure-menu-horizontal">
        <a href="#" className="pure-menu-heading pure-menu-link">IPFS File
Upload DApp</a>
      </nav>
      <main className="container">
        <div className="pure-g">
          <div className="pure-u-1-1">
            <h1>Your Image</h1>
            <p>This image is stored on IPFS & The Ethereum Blockchain!</p>

```

```

<img src={`https://ipfs.io/ipfs/${this.state.ipfsHash}`} alt=""/>

    <h2>Upload Image</h2>

    <form onSubmit={this.onSubmit} >

        <input type='file' onChange={this.captureFile} />

    <input type='submit' />

    </form>

</div>

</div>

</main>

</div>);
}
}

export default App

```

B) SCREENSHOTS

Upload file to IPFS

FileName

Upload File No file chosen

1.File Upload

CONFIRM TRANSACTION

Private Network

Ganache 1

627306...Ef57

99.792 ETH

57503.33 USD

345cA3...3e10

Amount

0 ETH

0.00 USD

Gas Limit

128913

UNITS

Gas Price

100

GWEI

Max Transaction Fee

0.012891 ETH

7.43 USD

Max Total

0.012891 ETH

7.43 USD

Data included: 132 bytes

RESET

SUBMIT

REJECT

1. Metamask

File Upload

Name : Sample

LINK : [QmcPXZAmAXyDjUqcveb6K9pfjTZ3XBGGAhK3MBFENtsF1n](#)

3. File Retrieving

ACCOUNTS

BLOCKS

TRANSACTIONS

LOGS

SEARCH FOR BLOCK NUMBERS OR TX HASHES

CURRENT BLOCK
4

GAS PRICE
20000000000

GAS LIMIT
6721975

NETWORK ID
5777

RPC SERVER
HTTP://127.0.0.1:7545

MINING STATUS
AUTOMINING

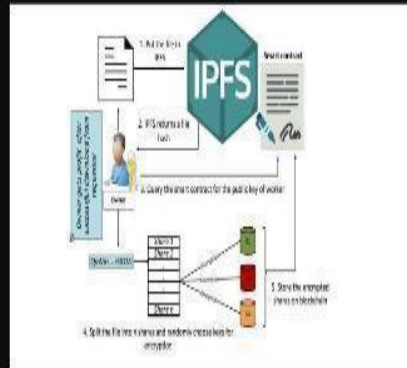
MNEMONIC

HD PATH

candy maple cake sugar pudding cream honey rich smooth crumble sweet treat

m/44'/60'/0'/0/account_index

4. Ganache



5. Final Output

Decentralized Cloud Storage using IPFS.docx

by Decentralized Cloud Storage Using Ipfs.docx Decentralized Cloud
Storage Using Ipfs.docx

Submission date: 03-Apr-2021 04:00AM (UTC-0400)

Submission ID: 1549522807

File name: Decentralized_Cloud_Storage_using_IPFS.docx (268.23K)

Word count: 2137

Character count: 11244

Abstract: Decentralized cloud storage is a Peer-to-peer network where each node provides the storage service to the customer's data. The storage system is based on the blockchain domain where it is completely decentralized. Blockchain ensures the user security and reliability of the user data. Peer-to-peer networking enables the user to store the data in different nodes across the network with security where the data are encrypted. This paper proposes a system which enables smart contract in the cloud storage system where it acts as an agreement between the client and the storage provider. The smart contract lets the user know about what data will be stored and the cost of storage. The decentralized cloud storage is reliable and secure compared to the earlier systems. This system is integrated with a smart contract which enables to keep an agreement and the transaction will happen accordingly to the implemented smart contract.

Keyword: Decentralized, Blockchain, Smart contracts

Introduction: In the last decade, technology development has been made by researchers to adapt data sharing methods. For these purposes, databases are being used as a warehouse to store data. Database plays a vital role for any individual as well as any other organizations to store its data. Recognizing the necessity of data and inadequate of storage, databases are reproduced, distributed, and reserved in different ways. Users store data in the cloud storage facilities provided by different companies. For security and bandwidth, data are scattered and replicated to different servers at different places. This actually seems to give a better solution for the management of swiftly increasing data. And it also guarantees data safety. In the future, the rate of increase of

data will definitely reach a record high. To get by with it, the present database structure requires to be more reliable, secure, and accessible all the time. Centralized cloud servers store an enormous portion of data, which is of centralized control. There are several types of inconvenience associated with the centralized control, such as single point failure. To bypass these failures, third-party applications are engaged to supply data backups. To freeze out the mediator for developing a confidence-based version, a blockchain is organized to produce trust and clarity. Decentralized cloud storage is the key, which enables the storage of data on various peers of the network in the structure of a distributed register. The issue with this network is the storage and processing limitations of the network nodes. For this specific purpose, the interplanetary file system (IPFS) is made use of, where a peer-to-peer network is used. In this network, there is no danger of single-point omission. It is alike web3, but with different attributes. It executes content addressing and functions in a similar way as the torrent.

Literature Survey: Jin Sun et al. proposed an access schema for electronic medical record in IPFS with blockchain based on Secure Storage. The author developed an attribute-based encryption schema for secured storage and systematic sharing of electronic medical records in an IPFS circumstance. The ideology that the author suggests is that the encryption of data is based on attributes and determines the attributes of the clients. Each client's private key is associated to their corresponding attributes whereas the ciphertext is related to the policy. Yan Zhu et al. suggest a decentralized storage scheme instead of a conventional centralized system. The provider performs a data integrity certificate to the user and only after it is verified,

user pays the storage fee. This payment information is stored in the secure blockchain. Shawn Wilkinson et al. used P2P cloud storage that executes end-to-end encryption which allows to share and transfer of data without a need for third-party data providers. The author implements a challenging algorithm that would cryptographically check the integrity and availability of the user data. David Vorick et al. introduced a simple decentralized storage system that enables the formation of contracts between peers. Contracts are like agreements between users and the provider which define what data will be stored and at what price. Satoshi Nakamoto introduced peer-to-peer networking which changed the entire domain. The author suggests a user send payments directly from one party to another without a need to the financial institution. This paper provides a solution to the double-spending problem with the introduction of peer-to-peer networks. Juan Bernet introduced InterPlanetary File System which is the distributed P2P file system that connects computing devices. The author designed the architecture in a way that provide a high delivery content addressed block storage model. Eli Ben-Sasson et al. explain in detail decentralized anonymous payments where the author points out some of the flaws in the bitcoin privacy guarantees. This paper fulfills that flaw by non-linking transactions from the payment's origin and it also reveals the payment desired location and the amount.

Existing System: Earlier, due to the non-usage of smart contracts in the previously proposed systems have issues with transparency of transactions. The cloud storage system doesn't ensure the user about the cost of the storage of data and what type of data will be stored. Some of the earlier systems didn't feature peer-to-peer networking which seems like a possible disadvantage. Most of the papers don't propose a privacy policy where some information the transaction destination which can be improved with further developing the system. Some papers propose encryption algorithms aren't secure as

encryption holds an important process in the cloud storage system where the data can't be interrupted by hackers. Advanced encryption algorithms are being developed which might provide better security to the users.

Centralized Vs Decentralized

	Centralized	Decentralized
Peers' connectivity	NA	Unstructured
Data Encryption	The file is encrypted and is being stored in a central server.	Data is encrypted and stored across peers
Data integrity	There is no definite operation to ensure data integrity	Heartbeat mechanism
Data Availability	Server isn't a reliable place as it is vulnerable	There are backups files created to ensure data availability to user in case of system failure.

Proposed System: This paper proposes decentralized cloud storage using an IPFS system where the data stored on this network is more secure and reliable than the conventional storage system. It is a simple and secure file storage system where the file is distributed across the network and when the file needs to be retrieved, decryption is performed on the data to restore it to its original form. First, the data is selected and it is fed into the system where the data is first sharded into several blocks of bits. The number of shards is determined by the total number of nodes available in the network. Encryption is the most crucial point of this process where the data needs to be secure before distributing it across the network. After the data is separated into several

shards which are considered as a data block is hash encrypted so that the receiver of this block doesn't understand the information. This process happens until the entire data is hashed encrypted. The introduction of smart contracts is used here as an agreement between the client and the service provider where the user knows what data is stored and what is the price for the stored data. Smart contracts are the set of protocols in which the decentralized follows and the added advantage is it discloses any confusion between the users and the provider. Smart contracts are implemented in the truffle suite in this project. IPFS is a BitTorrent-like network where the number of nodes is connected on the network so that the data can be uploaded and downloaded easily. IPFS is used here to distribute the data blocks across the network where several nodes can store the data which is encrypted. The address of this network is available in the IPFS so that the transactions of this network can be watched using wallets like ganache which is used in this project. The data encryption is a vital step in this model as the data is shared with multiple users and security of data should be maintained.



Figure 1. System architecture

The above flowchart explains the working process of the proposed model. The file is being selected for insertion into the IPFS and once the

file is inserted, the IPFS returns a hash value which is the address of the file in the network. The hash address is being shared with the user for their reference. Smart contract is being initialized for the transaction so that the trust between the users and provider is achieved. The data is split into n number of shares across the network for storage in multiple nodes. The user contains the key for the decryption of the hash file and it is not shared with anyone other than the user. Each shared file has a hash value which is to be decrypted by the user. The encrypted files are stored in blockchain where it is stored in several data blocks. There are various types of encryption techniques used in blockchain such as SHA -256 algorithm which gives a consistent hash value of 256 bits. Ganache is a private wallet where the user can watch the transactions and information about the block is also available. When the file is uploaded, the system requests for the transaction fee to be paid for the storage system. The frontend of this project is created using React which uses JavaScript Programming language. React is being used here for its various features and reusability ability.

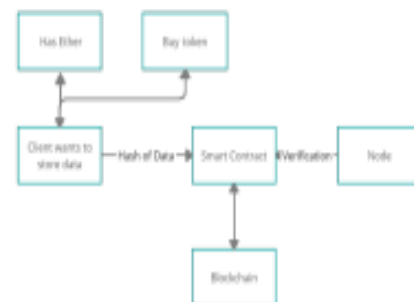


Figure 2. Transaction Architecture

The above flowchart describes the transaction organization of the storage model. The storage is paid in ether which is a cryptocurrency. The fee is described as a token so that the user can buy the tokens using ether. Once the amount is paid, the smart contracts execute the protocol which is to store the data once the fee is paid. The fee is verified for the storage of the split hash data of the file. The file is stored in the blockchain and there is a two-way communication between the

smart contract and Ethereum to process the fee and storage. Cryptocurrency is considered as a safe mode of transaction compared to the conventional centralized bank server. Ether is just one of the available cryptocurrencies in the blockchain domain.

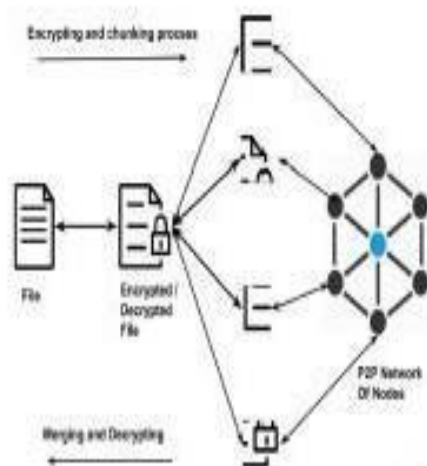


Figure 3. The sharding process

The file is being encrypted before split into several shards of data which is being stored in the nodes in the network. The file can be retrieved with the help of the key provided to the user and once the data is decrypted, the file is retrieved.

Results and Discussion:

Upload file to IPFS

FileName

Upload File No file chosen

Figure 4. Output window of the application

The contents of the retrieved block are decrypted and merged into the original file where the content is checked whether any loss of data is

recorded. This type of cloud storage is secure and is cost-effective compared to the centralized server. Each node in this network acts as a server and data is shared among them without knowing what the data contains and whose is it.

File Upload

Name : Sample

LINK : [QmPXZAmAN7DUccch6K9p57ZXBGGAhKJMBFENaFlQ](#)

Figure 5. Upload window of the application



Figure 6. Payment portal for storage usage

After the file is chosen, the metamask pops up to remind the transaction fees for this service which is paid in cryptocurrency where the transaction remains anonymous. Once the operation is authorized, the link is provided for the user to retrieve the file later.

Conclusion: This paper has devised a new model in blockchain in which storage is easier, secure, and immutable. The peer-to-peer network supported the entire application for node-to-node

communication which is achieved in this model. This blockchain application ensures user data security and reliability. The smart contract used in this model implements protocols for easy transactions between the users and provider.

Future Scope: Decentralized storage has a huge scope for the future and it is still developing. Data storage might be an issue in the future as more References;

- [1] R.C. Merkle. Protocols for public key cryptosystems, (April 1980). In Proc. 1980 Symposium on Security and Privacy, IEEE Computer Society, pages 122-133.
- [2] Nakamoto S. Bitcoin: A peer-to-peer electronic cash system. Consulted, 2008.
- [3] Vitalik Buterin. *Ethereum: A next-generation smart contract and decentralized application platform*. 2013
- [4] David Vorick et al. Sia: Simple Decentralized Storage. 2014.
- [5] S. King. Ppcoin: Peer-to-peer cryptocurrency with proof-of-stake. 2014.

and more data are being collected in our everyday life from IOT devices, Social media and so on. Decentralization technique could be the key to this problem.

- [6] Juan Bernet. IPFS - Content Addressed, Versioned, P2P File System. 2014.
- [7] Eli Ben-Sasson et al. Zerocash: Decentralized Anonymous Payments from Bitcoin. 2014.
- [8] Jin Sun et al. Blockchain-Based Secure Storage and Access Scheme for Electronic Medical Records in IPFS. IEEE. 2014.
- [9] Shawn Wilkinson et al. Storj: A Peer-to-Peer Cloud Storage Network. 2016.
- [10] Yan Zhu et al. Blockchain-based Decentralized Storage Scheme. IOP publishing. 1237.4. 2019.

Decentralized Cloud Storage using IPFS.docx

ORIGINALITY REPORT

13%

SIMILARITY INDEX

9%

INTERNET SOURCES

5%

PUBLICATIONS

0%

STUDENT PAPERS

PRIMARY SOURCES

1	mafiadoc.com Internet Source	6%
2	Yan Zhu, Chunli Lv, Zichuan Zeng, Jingfu Wang, Bei Pei. "Blockchain-based Decentralized Storage Scheme", Journal of Physics: Conference Series, 2019 Publication	1%
3	Jin Sun, Xiaomin Yao, Shangping Wang, Ying Wu. "Blockchain-Based Secure Storage and Access Scheme For Electronic Medical Records in IPFS", IEEE Access, 2020 Publication	1%
4	www.coincenter.org Internet Source	1%
5	Yahya Shahsavari, Kaiwen Zhang, Chamseddine Talhi. "A Theoretical Model for Block Propagation Analysis in Bitcoin Network", IEEE Transactions on Engineering Management, 2020 Publication	1%

6	citeseerx.ist.psu.edu Internet Source	1%
7	doaj.org Internet Source	1%
8	P Sreehari, M Nandakishore, Goutham Krishna, Joshin Jacob, V. S. Shibu. "Smart will converting the legal testament into a smart contract", 2017 International Conference on Networks & Advances in Computational Technologies (NetACT), 2017 Publication	1%
9	www.ijaerd.co.in Internet Source	<1%
10	www.researchgate.net Internet Source	<1%

Exclude quotes On

Exclude matches Off

Exclude bibliography On