RV Educational Institutions ®
RV College of Engineering ®

Autonomous
Institution Affiliated
to Visvesvaraya
Technological
University, Belagavi

Approved by AICTE,
New Delhi, Accredited
By NAAC, Bengaluru
And NBA, New Delhi

Go, change the world

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

DECENTRALISED FILE STORAGE USING BLOCKCHAIN

MINOR PROJECT REPORT

Submitted by,

Prasanna Bhat	**1RV18CS116**
Ratan Narayan Hegde	**1RV18CS131**
Rohit Parashuram Myali	**1RV18CS137**

Under the guidance of

Dr. Sowmyarani C N
Associate Professor
Dept of Computer Science
R.V.College of Engineering,

In partial fulfilment for the award of degree
of
Bachelor of Engineering
in
Computer Science and Engineering
2020-2021

RV COLLEGE OF ENGINEERING®, BENGALURU-59

(Autonomous Institution Affiliated to VTU, Belagavi)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

Certified that the minor project work titled ***'DECENTRALISED FILE STORAGE USING BLOCKCHAIN'*** is carried out by **Prasanna Bhat (1RV18CS116), Ratan Narayan Hegde (1RV18CS131), and Rohit Parashuram Myali (1RV18CS137)** who are bonafide students of RV College of Engineering, Bengaluru, in partial fulfilment for the award of degree of **Bachelor of Engineering in Computer Science and Engineering** of the Visvesvaraya Technological University, Belagavi during the year 2020-2021. It is certified that all corrections/suggestions indicated for the Internal Assessment have been incorporated in the minor project report deposited in the departmental library. The Minor Project report has been approved as it satisfies the academic requirements in respect of minor project work prescribed by the institution for the said degree.

Signature of Guide	**Signature of Head of the Department**	**Signature of Principal**
Dr. Sowmyarani C N	**Dr. Ramakanth Kumar P**	**Dr.K.N.Subramanya**

External Viva

Name of Examiners	**Signature with Date**
1	
2	

RV COLLEGE OF ENGINEERING®, BENGALURU-59

(Autonomous Institution Affiliated to VTU, Belagavi)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

DECLARATION

We, Prasanna Bhat, Ratan Narayan Hegde, Rohit Parashuram Myali, students of sixth semester B.E., department of CSE, RV College of Engineering, Bengaluru, hereby declare that the minor project titled ***'DECENTRALISED FILE STORAGE USING BLOCKCHAIN'*** has been carried out by us and submitted in partial fulfilment for the award of degree of **Bachelor of Engineering** in **Computer Science and Engineering** during the year 2020-21.

Further we declare that the content of the report has not been submitted previously by anybody for the award of any degree or diploma to any other university.

We also declare that any Intellectual Property Rights generated out of this project carried out at RVCE will be the property of RV College of Engineering, Bengaluru and we will be one of the authors of the same.

Place: Bengaluru

Date:

Name	**Signature**
1. Prasanna Bhat (1RV18CS116)	
2. Ratan Narayan Hegde (1RV18CS131)	
3. Rohit Parashuram Myali (1RV18CS137)	

ACKNOWLEDGEMENT

We are indebted to our guide**, Dr. Sowmyarani C N,** Associate Professor**, Dept of CSE** for her wholehearted support, suggestions and invaluable advice throughout our project work and also helped in the preparation of this thesis.

We also express gratitude to our Minor Project lab faculty **Dr. Sowmyarani C N, Associate Professor, and Dr. Krishnappa H K, Associate Professor,** Department of Computer Science and Engineering for their valuable comments and suggestions.

Our sincere thanks to **Dr. Ramakanth Kumar P.**, Professor and Head, Department of Computer Science and Engineering, RVCE for his support and encouragement.

We express sincere gratitude to our beloved Principal, **Dr. K. N. Subramanya** for his appreciation towards this project work.

We thank all the **teaching staff and technical staff** of the Computer Science and Engineering department, RVCE for their help.

Lastly, we take this opportunity to thank our **family** members and **friends** who provided all the backup support throughout the project work.

Abstract

2.5 quintillion bytes of data are produced each day. Out of the total data in the world over 90 percent of data was produced in the last 2 years. Cloud storage is one of the leading options to store massive data, however, the centralized storage approach of cloud computing is not secure. Much of the data currently available through the internet is quite centralized and is stored with a handful of technology companies that have the experience and capital to build massive data centers capable of handling this enormous data. The problem with this approach is the security of data. On the other hand, Blockchain is a decentralized cloud storage system that ensures data security. Any computing node connected to the internet can join and form a peer network thereby maximizing resource utilization and security.

Blockchain is a distributed peer to peer system where each node in the network stores a copy of blockchain thus making it immutable. In the proposed system, the user's file is encrypted and stored across multiple peers in the network using the IPFS (InterPlanetary File System) protocol. IPFS creates a hash value. The hash value indicates the path of the file and is stored in the blockchain. This system focuses on decentralized secure data storage, high availability of data, and efficient utilization of storage resources. A web application built using Reactjs, web3js on top of IPFS and ethereum blockchain connected through metamask is used.

The Decentralized application currently hosted on a local blockchain performs better than centralized storage options. Our system not only solves the privacy and security concerns of centralized cloud storage but also provides a medium for the peer to rent their underutilized storage and earn cryptocurrency in return.

Through analysis and evaluations, our proposed scheme significantly improves data integrity and credibility for distributed file storage. Besides, based on blockchain, decisions on distributed file storage shall be more easier and reasonable. Though some limitations like complex scalability and verifying credibility of users exist, it still can be considered as a Minimum Viable Product for the same. Future improvements like incentives to people who rent their storage and a credit system can be implemented. Also a separate cryptocurrency can be implemented and used for this to be made as a standalone application.

Table of Contents

1. Introduction

1.1 State of the art development

In recent years, with the continuous improvement of information technology, people's demand for computing and resource storage has also shown a rapid growth trend. People are constantly exploring new ways of computing to satisfy the pursuit of higher computing power and larger storage space. After the emergence of computing models such as peer-to-peer computing, grid computing, utility computing, and distributed computing, cloud computing has become a focus of academic and industrial circles. Cloud computing distributed computing tasks across resource pools of a large number of computers, enabling various application systems to acquire computing power, storage space, and software services as needed.

1.2 Motivation

Due to issues in integrity, trust, control, and credibility, we focus on overcoming the issue of integrity and credibility for distributed file storage. There are various systems and platforms for distributed file storage, and they aim to collect all kinds of data. Notably, this incurs a severe privacy problem, since most users have no knowledge of these actions, much less about control of such actions. To solve this problem, we suppose in this work that all provided services should obey the smart contracts, especially some assigned protocols. Based on this, this proposed work devotes to the following issues:

Data Credibility. Our research focuses on the data credibility for distributed file storage; we should guarantee that authorized users must control all personal data. Meanwhile, the systems and platforms regard the services as guests who have corresponding permissions.

Data Integrity. All data should be verified and detected to guarantee the integrity of stored data. All data-trace is totally transparent for each authorized user, and any illegal modification is impractical on the platform.

Access Control. Any users should be granted access permission as they log in the system or platform. These permissions should define which resources the users can utilize. Within the permissions, users can change the access range of their stored data. Meanwhile, all participating users must store data access control strategies or policies on the blockchain. Thus, illegal access is hardly impossible.

1.3 Problem statement

Data privacy and security are concerns when data resides in third party storage. Storage can be created from the underutilized resources of peers. Data security, privacy, availability, and resource utilization are the areas handled by the proposed system.

1.4 Objectives

The objective of this project is to implement a technique that will satisfy the following objectives:

- To develop a decentralized file sharing system that makes effective use of resources and easy availability of files.
- To develop a system that is secure using blockchain technology.

1.5 Methodology

- Uploading of files by user which in turn uploaded IPFS.
- Getting the IPFS hash value for the file. This hash is then mapped with the user's address using a smart contract and gets stored securely in the blockchain.
- For Data Retrieval The hash value for the required file is known from the previous upload.
- This IPFS hash value is used to find the peer node on the network and fetch the file and then download it on the user's device.

1.6 Summary

Blockchain is a decentralized cloud storage system that ensures data security. Any computing node connected to the internet can join and form a peer network thereby maximizing resource utilization.

2. Literature Survey

2.1 Introduction

Existing way of storing files had some adverse effects on the securities so there had been several experiments/projects done to ensure best security and overall performance of the system . so here we have listed some of the works done in order to replace the insecurity system of storing files.

2.2 Related Work

Zhe, Diao, et al [1], discusses the increasing demand for cloud storage with associated security and privacy issues in centralized cloud storage. As per the discussion by encrypting the data and scattering the data across multiple nodes, a high level of data security can be achieved. Lee et al [2], has shown encryption enhances the security of user's data stored in cloud storage. Authors have used the AES encryption algorithm to enhance security with speed without impacting the system's performance. Nakamoto, Satoshi [3], uses the concept of bitcoin in blockchain technology to show transaction records. The transaction details are stored in the blocks and are chained to each other serially, using the concept of hashing. Every peer involved in the network has a copy of the blockchain to verify the credibility of the blockchain. The author claims that the transactions stored in the peer to peer network are tamperproof, cannot be altered by an attacker and the identity of all the parties involved in transactions is secure. The peer-to-peer network uses proof-of-work to record a public history of transactions that quickly becomes computationally impractical for an attacker to change if honest nodes control a majority of CPU power.

Zyskind [4], raises the problems related to centralized cloud storage and suggests blockchain to solve the issues. The proposed system allows two transactions viz, Taccess and Tdata, Taccess for access control permission which will be set by the respective user who owns the data, and Data are used for data storage purposes. The shared encryption key secures the data from third parties. Cachin, Christian et al [5], discusses the architecture of hyper ledger blockchain fabric, limitations of electronic coins, working of hyper ledger fabric, and proof of work consensus algorithm. Hyperledger Fabric is a permissioned blockchain network that allows only limited nodes that have permission to add new blocks in the blockchain. In paper [6], the author explains the architecture of Ethereum and the working of smart contracts. Bitcoin was used only for sending and receiving cryptocurrency but lacked to add business logic. Ethereum applications like decentralized file storage and decentralized autonomous systems are discussed. Ruj, Sushmita [7], proposes BlockStore, a decentralized framework using blockchain technology to enhance security, transparency in transactions between peers (Host and Renters). The system uses proof of storage and proof of work to verify that hosts do not meddle with data in Blockchain. The proposed system does not encrypt or decrypt data before uploading it to peers which creates a threat to confidentiality and privacy of user's data. Juan Benet et al. [8], introduces a new peer to peer file transfer protocol called IPFS (InterPlanetary File System).

IPFS uses a content-based addressing scheme. As per the author, IPFS provides a high throughput content-addressed block storage model along with content-addressed hyperlinks. Li, Dagang [9], discusses how data sharing in blockchain - based applications differs from traditional applications. The author identifies that data-sharing in decentralized architecture is cumbersome. The author proposes Meta-key for secure data sharing in a decentralized storage system based on blockchain also focuses on the collusion-free property of the proposed cryptographic protocol and proved it strictly. Wohrer et al. [10], explains the solidity used for creating smart contracts in blockchain and its difficulty. All the security issues which have been resolved are 1.Checks -Effects Interaction, 2.Emergency stop, 3.Speed bump, 4. Rate limit, 5. Mutex and 6. Balance limit. The knowledge related to these issues can be found in grey literature and many blog articles. In [11], blockchain provides scalability, security, and sustainability, it is also helpful to transform the way of doing business. In this paper, the author is trying to conduct a comprehensive survey on the technical and application of blockchain technology by discussing its structure to different consensus algorithms. The author has also explained, the structure of blockchain consists of data, timestamp, and address of the previous block in hash form.

The timestamp is recording the time when the block was created. A hash function is the one that takes an input of any length and generates the output with a unique fixed length. Each block contains a hash value of the previous block. Therefore, security is increased in Blockchain. It uses proof of stack (POS), proof of work (POW) consensus algorithms as a measure to discourage the attacks of Denial of Service and miner can validate transactions in a block depending on the amount the user holds respectively. Therefore, blockchain technology is exceedingly recognized and appraised due to its decentralized infrastructure and peer-to-peer nature. In paper [12] D. Sivaganesan has suggested the use of blockchain to improve security and provide transparency in IoT applications. The author has proposed a smart logistic system for the pharmaceutical sector to keep track of the shipment for medicines. This system combines blockchain and IOT, it also includes a smart contract to bond manufactures, distributes, and the dispenser legally. The use of smart contracts avoids third person intervention as well as provides improved security and transparency in the transaction.

2.3 Summary

The various methods in the survey do not have real storage features and provide limited functionality to the user, which is overcome by the proposed method.

3. Software Requirements Specification

A software requirements specification (SRS) describes what the software will do and how it will be expected to perform. It describes the functionality the product needs to fulfill all stakeholders (business, users) needs. It also describes the non-functional requirements that the product satisfies.

3.1 Functional Requirements

3.1.1 Uploading of file:

User uploads a file using the file picker. The system checks the file size and ensures storage availability in the network. The file is uploaded when enough storage is available. Then the system performs steps. Users are notified to try again when enough storage is unavailable.

3.1.2 Encryption of file :

The uploaded file is encrypted using AES 256 bit algorithm. The encryption key is generated using the user's wallet address and randomly generated salt value. This encryption key along with an IV is used to encrypt the user's data. This maintains the confidentiality of the user's data.

3.1.3 Storing of file across multiple peers:

The encrypted file is then divided into blocks of 64KB and sent to different peers across the network with the help of the IPFS protocol. The proposed system uses a private IPFS network to allow registered peers to store the file in the network. The file block is replicated on multiple peer's storages for high availability using the IPFS cluster

3.1.4 Storing of file across multiple peers :

IPFS returns a hash value which indicates the path of the file. The hash value along with metadata is mapped with the user's wallet address and is storedts between parties under certain conditions. This is lines of code stored on a blockchain network and are automatically executed when predetermined terms and conditions are met. In our proposed system preconditions for the smart contract to execute are: 1) Enough Space is available in the network to store files. 2) The user has sufficient wallet balance to pay the peers.

3.1.5 Paying the peers for file storage :

Once the file is stored across peers, total cryptocurrency is calculated and is deducted from the user's wallet. This cryptocurrency is first transmitted to the smart contract from the user's wallet. With the smart contract, this amount is distributed to the peers who have stored the user's file.

3.2 Non-Functional Requirements

3.2.1 Accuracy and Performance Requirements

The application should correctly execute the sequence of displaying and hiding sources as mentioned by the user. The application should execute fast and without any overheads. It should scale well to large inputs as well.

3.2.2 Compatibility Requirements

The application should be highly compatible with different system configurations. The application should also easily co-exist with other applications without much conflicts and dependencies.

3.2.3 Maintainability & Manageability Requirements:

The application should be composed of multiple components and should be easily maintainable and serviceable independently. The application must also be easy to correct faults, improve performance and adapt to changed environments.

3.2.4 Usability Requirements:

The application should be efficient, effective and easy to learn. The application can be used easily by specifying the arguments in the command line. Detailed explanation of input file format must be mentioned.

3.3 Hardware Requirements

The Hardware requirements related to the project are:

- Processor: Quad Core 64-bit
- RAM: 2GB or above

3.4 Software Requirements

- Any Blockchain Browser Installed (A normal Browser like chrome, firefox etc extended with a crypto wallet like metamask, oxygen etc)
- IPFS - InterPlanetary FileSystem

3.5 Summary

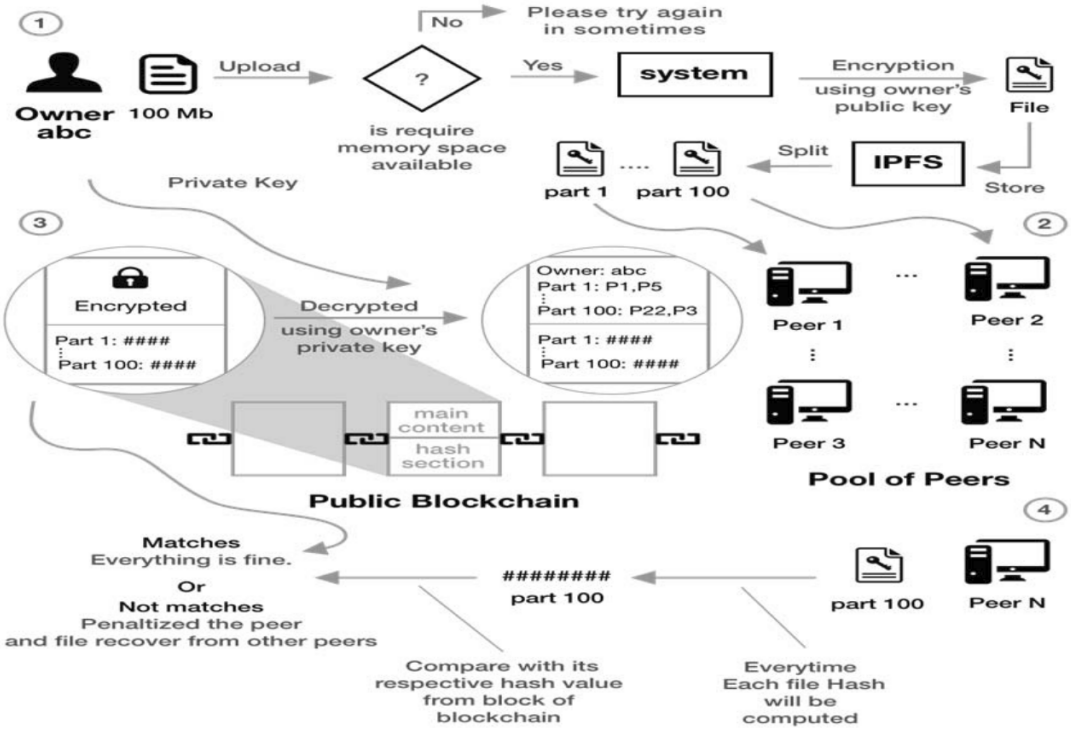
The functional requirements of the program are discussed. The project aims to provide functionalities which are useful and user friendly to the user. The non-functional requirements are discussed which involve performance requirements, testability, fairness, reliability and transparency. The hardware and software requirements are discussed.

4. Design of File Storage System

4.1 High level design

The proposed system works in four modules. The user first creates an account on the metamask. The user's account address and wallet balance are fetched in the app through web3.js from the metamask. Users select the file to upload through file picker. System checks for the number of available peers. Further, the AES algorithm uses the user's wallet address as a key and encrypts the uploaded file. A payment dialogue seeks for the user's confirmation. On confirming the payment, the user's file is stored across available peers using IPFS protocol. IPFS then returns a hash value consisting of the path of the file. This path is then mapped with the user's address using a smart contract and gets stored securely in the blockchain. To achieve high availability and reliability of data, the uploaded data is replicated on three peers. For better performance the system blacklists peers every time they are unavailable for data retrieval.

4.1.1 System Architecture



system design

4.2 Detailed Design

The terminology is briefly discussed below-

- Metamask : Browser extension which acts as a bridge to connect with the ethereum network.
- Ethereum network : It is an open-source, public blockchain- based distributed computing platform. Ethereum uses smart contracts where one can add business logic to make decentralized applications as per the business requirements.
- Peers: These are the users of the system who have pledged to rent their free storage for other user's to store files.
- AES: Advanced Encryption Standard (AES) is a symmetric- key algorithm that supports block length of 128 bit and can have a key size of 128, 192, and 256 bits.
- IPFS protocol: IPFS is an open-source peer to peer file transfer protocol.

The following methodologies are designed according to the need:

A. Uploading of file

User uploads a file using the file picker. The system checks the file size and ensures storage availability in the network. The file is uploaded when enough storage is available. Then the system performs step B . Users are notified to try again when enough storage is unavailable.

B. Encryption of file

The uploaded file is encrypted using AES 256 bit algorithm. The encryption key is generated using the user's wallet address and randomly generated salt value. This encryption key along with an IV is used to encrypt the user's data. This maintains the confidentiality of the user's data.

C. Storing of file across multiple peers

The encrypted file is then divided into blocks of 64KB and sent to different peers across the network with the help of the IPFS protocol. The proposed system uses a private IPFS network to allow registered peers to store the file in the network. The file block is replicated on multiple peer's storages for high availability using the IPFS cluster.

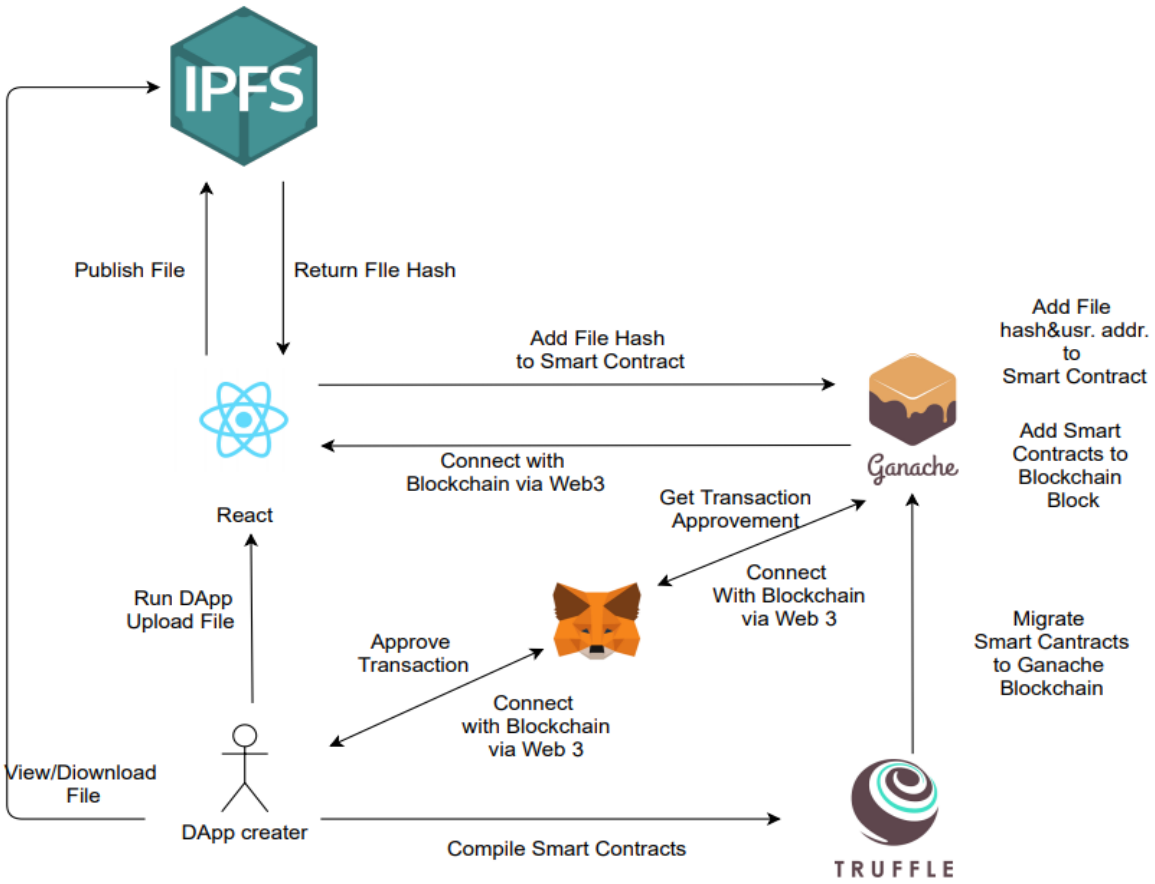
D. Storing of file across multiple peers

IPFS returns a hash value which indicates the path of the file. The hash value along with metadata is mapped with the user's wallet address and is stored in the blockchain using a smart contract. Smart contracts are like agreements and are used to eradicate the need for a third party. They control the transaction between nodes or assets between parties under certain conditions. This is lines of code stored on a blockchain network and are automatically executed when predetermined terms and conditions are met.

E. Paying the peers for file storage

Once the file is stored across peers, total cryptocurrency is calculated and is deducted from the user's wallet. This cryptocurrency is first transmitted to the smart contract from the user's wallet. With the smart contract, this amount is distributed to the peers who have stored the user's file.

4.2.1 workflow



- Uploading of files by user which in turn uploaded IPFS.
- Getting the IPFS hash value for the file. This hash is then mapped with the user's address using a smart contract and gets stored securely in the blockchain.
- For Data Retrieval The hash value for the required file is known from the previous upload.
- This IPFS hash value is used to find the peer node on the network and fetch the file and then download it on the user's device.

5. Implementation of File Storage System

5.1 Programming Language Selection

Solidity is an object-oriented, high-level language for implementing smart contracts. Smart contracts are programs which govern the behaviour of accounts within the Ethereum state.Solidity is a curly-bracket language. It is influenced by C++, Python and JavaScript, and is designed to target the Ethereum Virtual Machine (EVM). You can find more details about which languages Solidity has been inspired by in the language influences section.Solidity is statically typed, supports inheritance, libraries and complex user-defined types among other features. With Solidity you can create contracts for uses such as voting, crowdfunding, blind auctions, and multi-signature wallets.

5.2 Platform selection

A blockchain is a growing list of records, called *blocks*, that are linked together using cryptography. Each block contains a cryptographic hash of the previous block, a timestamp, and transaction data (generally represented as a Merkle tree). The timestamp proves that the transaction data existed when the block was published in order to get into its hash. As blocks each contain information about the block previous to it, they form a chain, with each additional block reinforcing the ones before it. Therefore, blockchains are resistant to modification of their data because once recorded, the data in any given block cannot be altered retroactively without altering all subsequent blocks.

Blockchains are typically managed by a peer-to-peer network for use as a publicly distributed ledger, where nodes collectively adhere to a protocol to communicate and validate new blocks. Although blockchain records are not unalterable as forks are possible, blockchains may be considered secure by design and exemplify a distributed computing system with high Byzantine fault tolerance.

5.3 Code Conventions

Uniform naming conventions are followed throughout the application. Below table 1 depicts the naming conventions used.

Type of Name	**Naming convention**
Class name	Camel-case starting with uppercase
Class function	Lowercase separated by underscore

Enum Class name	Camel-case starting with uppercase
Struct name	Camel-case starting with uppercase

Indentation

- The 4-space indentation rule is followed.
- Tabs are preferred over spaces.
- Large expressions are broken into several lines.

Comments

- Comments are used to describe a piece of code, for future reference.

Maximum line length

- No more than 79 characters in a line
- The statements are broken into multiple lines and turned into shorter lines of code.

Blank lines

- Top-level functions and classes are separated by two blank lines
- Methods inside a class are separated by a single blank line

5.5 Summary

The proposed system enhances the security of data by encrypting and distributing the data across multiple peers in the system. Implemented system uses the AES 256 bit encryption algorithm to encrypt the data ensuring the confidentiality of the user's data. Encrypted data is then distributed and stored across peers in the network using the IPFS protocol. Our system not only solves the privacy and security concerns of centralized cloud storage but also provides a medium for the peer to rent their underutilized storage and earn cryptocurrency in return thereby, maximizing the storage resource utilization.

6. Experimental Results and Testing

6.1 Evaluation Metrics

The evaluation of the Decentralized Storage System is based on the functional requirements. The following discussion shows the experimental results.

6.1.1 Authentication and Authorization

In order to evaluate the security of the blockchain network the user is authenticated based on the single sign on (SSO) feature of the crypto wallet authentication and this key is required for the authentication and this is asserted with the test framework.

6.1.2 Services

There are different services like File Upload service, getting a list of files uploaded by different users, connecting to Crypto Wallet present in the browser and transfer of ethers for each transaction as gas cost.

The evaluation for file upload service is performed by uploading the file from a test framework and checking for validity at the ipfs terminal by the file hash available from the former part.

The evaluation of the list of file upload transactions and its integrity is done by asserting the presence of a transaction receipt on the Etherscan website attached with the file uploaded.

6.1.3 Deployment

Verifying the integrity of deployed smart contracts and also the migration of json data to the blockchain. The evaluation for this matrix is done by deploying the smart contract to the test blockchain and verifying the deployment details

6.2 Unit Testing

The unit testing describes the tests on individual functions that build the Decentralized Storage application. The below is the recordings of the unit tests.

6.2.1 Deployment on Local Blockchain

This test is performed by command $truffle migrate and verifying the deployment stats and the transaction has on Etherscan.io. This should also reflect in the reduced ether balance of the author or the owner of the smart contract

```
2_deploy_contracts.js
=====================

   Replacing 'DStorage'
   --------------------
   > transaction hash:    0x5fecbe2f9fbe9cbff0d3118c9ed4d1d491a86a323dd728b34f4c5f0ca18cb7a9
   > Blocks: 0            Seconds: 0
   > contract address:    0x90Ee4500DeB129738201b1f342C99F494c464648
   > block number:        3
   > block timestamp:     1627045939
   > account:             0x68D0813FeAcCd96edaEdc60286E90D8308412Fb5
   > balance:             99.97759958
   > gas used:            852421 (0xd01c5)
   > gas price:           20 gwei
   > value sent:          0 ETH
   > total cost:          0.01704842 ETH


   > Saving migration to chain.
   > Saving artifacts
   -------------------------------------
   > Total cost:          0.01704842 ETH


Summary
=======
```

Unit testing - output for deployment

6.2.2 Migration

This is to push the smart contracts to the Ethereum blockchain (either local, tesnet or mainnet) and to set up necessary steps for linking contracts with other contracts as well as populating contracts with initial data. Then the transaction hash is verified on the truffle console

```
1_initial_migration.js
======================

   Replacing 'Migrations'
   ----------------------
   > transaction hash:    0x7f8a3301195aceeeaa31c8d1800e6733566f3a85bb3a831b393b4f9954d1c5a7
   > Blocks: 0            Seconds: 0
   > contract address:    0xcE166DA57232ACAe061ec52261F58B9d88FEa30e
   > block number:        1
   > block timestamp:     1627045938
   > account:             0x68D0813FeAcCd96edaEdc60286E90D8308412Fb5
   > balance:             99.99549526
   > gas used:            225237 (0x36fd5)
   > gas price:           20 gwei
   > value sent:          0 ETH
   > total cost:          0.00450474 ETH


   > Saving migration to chain.
   > Saving artifacts
   -------------------------------------
   > Total cost:          0.00450474 ETH
```

Unit testing - output for migration

6.2.3 Services and Functional Requirements

Using some javascript based test frameworks like Mocha and Chai various unit test's pertaining different edge cases and different use cases have been written as the below shown example and the output of running these test's on the test network like Ganache and truffle

```
let result, fileCount
const fileHash = 'QmV8cfu6n4NT5xRr2AHdKxFMTZEJrA44qgrBCr739BN9Wb'
const fileSize = '1'
const fileType = 'TypeOfTheFile'
const fileName = 'NameOfTheFile'
const fileDescription = 'DescriptionOfTheFile'

before(async () => {
  result = await dstorage.uploadFile(fileHash, fileSize, fileType, fileName, fileDescription, { from: uploader })
  fileCount = await dstorage.fileCount()
})

//check event
it('upload file', async () => {
  // SUCESS
  assert.equal(fileCount, 1)
  const event = result.logs[0].args
  assert.equal(event.fileId.toNumber(), fileCount.toNumber(), 'Id is correct')
  assert.equal(event.fileHash, fileHash, 'Hash is correct')
  assert.equal(event.fileSize, fileSize, 'Size is correct')
  assert.equal(event.fileType, fileType, 'Type is correct')
  assert.equal(event.fileName, fileName, 'Name is correct')
  assert.equal(event.fileDescription, fileDescription, 'Description is correct')
  assert.equal(event.uploader, uploader, 'Uploader is correct')
```

```
it('upload file', async () => {
  // SUCESS
  assert.equal(fileCount, 1)
  const event = result.logs[0].args
  assert.equal(event.fileId.toNumber(), fileCount.toNumber(), 'Id is correct')
  assert.equal(event.fileHash, fileHash, 'Hash is correct')
  assert.equal(event.fileSize, fileSize, 'Size is correct')
  assert.equal(event.fileType, fileType, 'Type is correct')
  assert.equal(event.fileName, fileName, 'Name is correct')
  assert.equal(event.fileDescription, fileDescription, 'Description is correct')
  assert.equal(event.uploader, uploader, 'Uploader is correct')

  // FAILURE: File must have hash
  await dstorage.uploadFile('', fileSize, fileType, fileName, fileDescription, { from: uploader }).should.be.reje

  // FAILURE: File must have size
  await dstorage.uploadFile(fileHash, '', fileType, fileName, fileDescription, { from: uploader }).should.be.reje

  // FAILURE: File must have type
  await dstorage.uploadFile(fileHash, fileSize, '', fileName, fileDescription, { from: uploader }).should.be.reje

  // FAILURE: File must have name
  await dstorage.uploadFile(fileHash, fileSize, fileType, '', fileDescription, { from: uploader }).should.be.reje

  // FAILURE: File must have description
  await dstorage.uploadFile(fileHash, fileSize, fileType, fileName, '', { from: uploader }).should.be.rejected;
})
```

Unit Testing - using Mocha and chai frameworks

```
prasa@DESKTOP-2FO2HFS MINGW64 /a/MinorProject/dstorage (master)
$ truffle test
Using network 'development'.


Compiling your contracts...
===========================
> Everything is up to date, there is nothing to compile.



  Contract: DStorage
    deployment
      √ deploys successfully
      √ has a name (126ms)
    file
      √ upload file (1721ms)
      √ lists file (90ms)
    Non Functional Requirements
      √ Authentication with Metamask
      √ Authorization
      √ IPFS integration


  7 passing (3s)
```

Unit Testing - output

6.3 System Testing

The system testing is performed on a complete integrated system to evaluate the compliance of the system with the corresponding requirements.The integration is tested as shown in the above output of the integrated system testing

7. Conclusion and Future Enhancement

7.1. Conclusion, Limitations and Future Enhancements

7.1.1 Conclusion

We explore blockchain in distributed file storage, users no longer require a third-party, and own heavy supervision of their data. Through analysis and evaluations, our proposed scheme significantly improves data integrity and credibility for distributed file storage. Besides, based on blockchain, decisions on distributed file storage shall be more easier and reasonable. Finally, we carried out detailed discussion on the latest relative systems and demonstrated the advantages of this proposed work in distributed file storage.

7.1.2 Limitations

It's difficult to fix any issues in DApps because fixes require every peer in the network to update all the copies in the network. Most of the current centralized apps depend on user verification, which is quite easy given that a single authority controls and verifies it. But DApps don't have a single entity responsible for doing KYC verification. In the current centralized app mechanism, we often have to depend on third-party APIs for fetching certain third-party information. However, with DApps, we don't have this leverage because currently there is no such large third-party DApps ecosystem in place.

7.1.2 Future Enhancement

In the future, an adaptive scheduling algorithm can be incorporated with which files can be accessed multiple times by the user as compared to the one which is accessed rarely. This will help to ensure that frequently accessed files are available easily to the user whenever required. Also, a credit system can be added with which each peer will be assigned a default 100 credit, based on their system uptime, and several successfully served file access that requests their credits will be either deducted or added. Peer's with more credits will be given higher priority for data storage.

7.2. Summary

The proposed system enhances the security of data by encrypting and distributing the data across multiple peers in the system. Implemented system uses the AES 256 bit encryption algorithm to encrypt the data ensuring the confidentiality of the user's data. Encrypted data is then distributed and stored across peers in the network using the IPFS protocol. Our system not only solves the privacy and security concerns of centralized cloud storage but also provides a medium for the peer to rent their underutilized storage and earn cryptocurrency in return thereby, maximizing the storage resource utilization.

8. References

1. Zhe, Diao, ”Study on Data Security Policy Based On Cloud Storage” 2017 ieee 3rd international conference on big data security on cloud (bigdatasecurity), ieee international conference on high performance and smart computing (hpsc), and ieee international conference on intelligent data and security (ids) IEEE, 2017.

2. Lee, Bih-Hwang, Ervin Kusuma Dewi, and Muhammad Farid Wajdi. ”Data security in cloud computing using AES under HEROKU cloud.” 2018 27th Wireless and Optical Communication Conference (WOCC). IEEE, 2018.

3. Nakamoto, Satoshi,”Bitcoin: A peer-to-peer electronic cash system”, (2008).

4. Zyskind, Guy, and Oz Nathan,”Privacy: Using blockchain to protect personal data”,IEEE Security and Privacy Workshops. IEEE, 2015.

5. Cachin, Christian,” Architecture of the hyperledger blockchain fabric”, Workshop on distributed cryptocurrencies and consensus ledgers. Vol. 310. 2016.

6. Buterin, Vitalik, ”A next-generation smart contract and decentralized application platform”, white paper (2014).

7. Ruj, Sushmita, et al, ”BlockStore: A Secure Decentralized Storage Framework on Blockchain” 2018 IEEE 32nd International Conference on Advanced Information Networking and Applications (AINA). IEEE, 2018.

8. Benet, Juan, “ IPFS - Content Addressed, Versioned, P2P File System.” 2014.

9. Li, Dagang, et al. Meta-Key: ”A Secure Data-Sharing Protocol Under Blockchain-Based Decentralized Storage Architecture”, IEEE Network- ing Letters 1.1 (2019): 30-33.

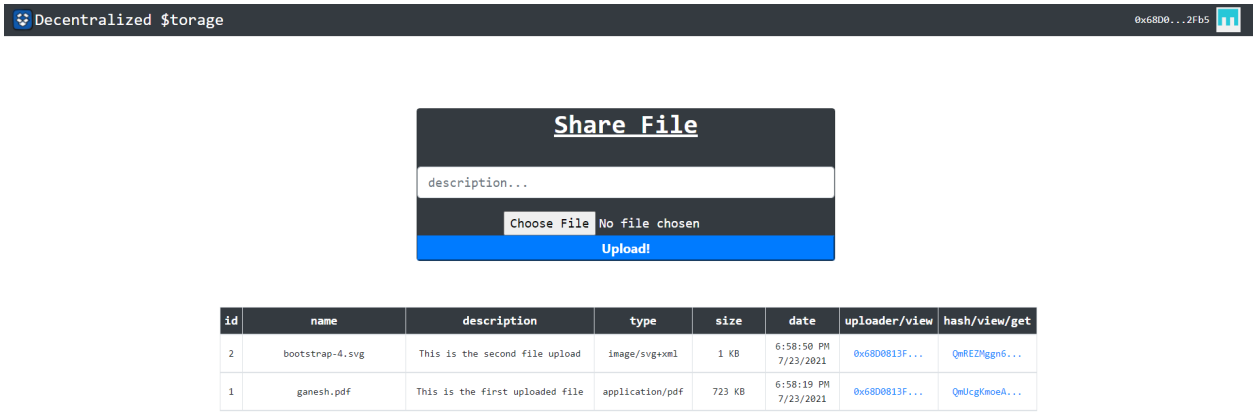
10. Wohrer, Maximilian, and Uwe Zdun,”a Smart contracts: security patterns in the ethereum ecosystem and solidity”, International Workshop on Blockchain Oriented Software Engineering (IWBOSE). IEEE, 2018.

11. Sum, V. “ SECURITY AND PRIVACY MECHANISM USING BLOCKCHAIN.” Journal of Ubiquitous Computing and Communication Technologies (UCCT) 1.01 (2019): 45 -54

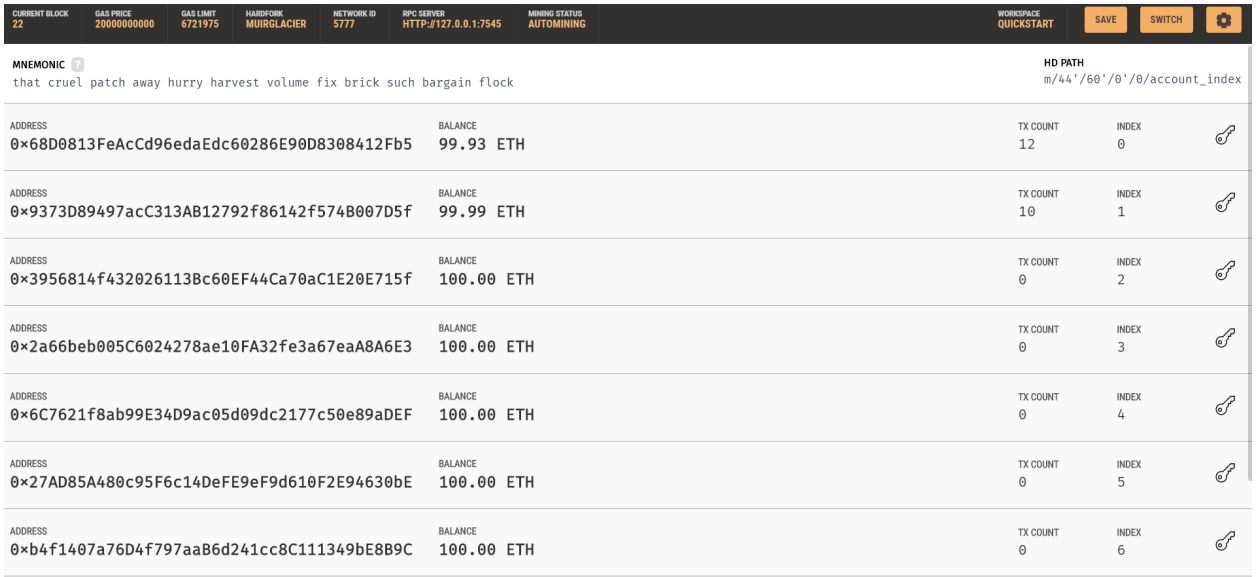
12. Sivaganesan, D. “ BLOCK CHAIN ENABLED INTERNET OF THINGS.” Journal of Information Technology 1.01 (2019): 1-8..

9. Appendices

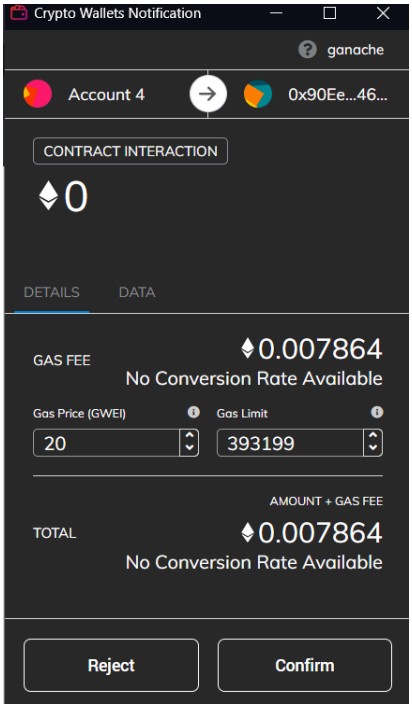
9.1 Screenshots



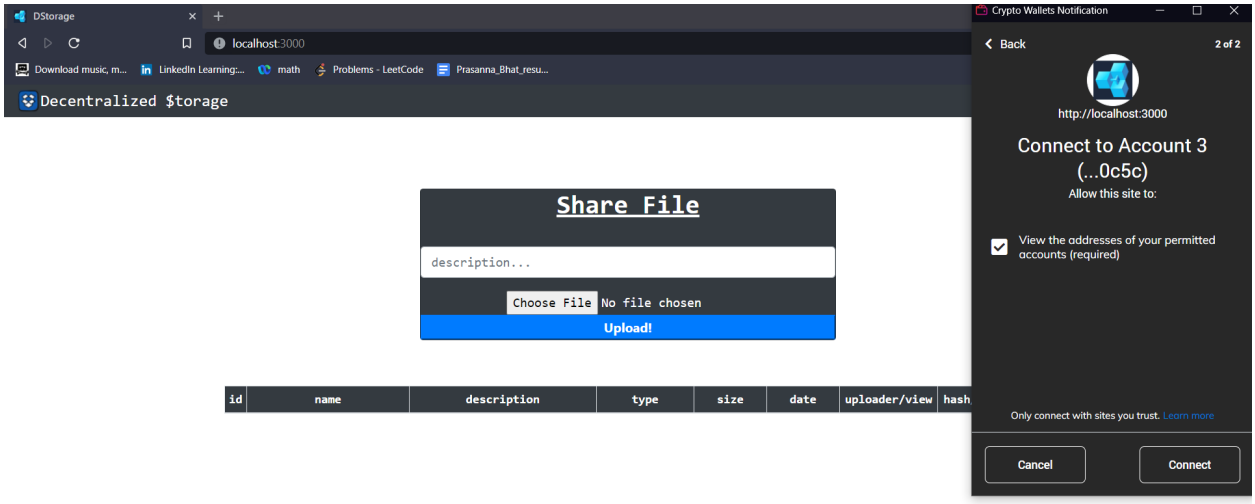
Decentralized Storage web application



Local test Blockchain Network - Ganache



File Upload Notification - Crypto Wallet(Brave / Metamask)



Integration of Crypto Wallet and Chrome Browser