ABSTRACT

This project is about web application for file security. In this project, a blockchain-based secure file security system is presented.

The main focus of this system is to provide data authenticity and quality of data to user. A decentralized storage IPFS provides the solution for bloating problem at owner's end. Data hashes returned by the IPFS are encrypted using AES encryption to encrypt the file hash with private key of the user and then it adds it to to previous hash in the blockchain to provide security from malicious activity

Incremental methodology is used in this project to ensure testing at every iteration when new features are added.It ensures we can reassess the whole system to make sure it meets with our requirement.The final system is bugless after a few times of refining and debugging.

LIST OF FIGURES

LIST OF ABBREVIATIONS

AES – Advanced Encryption Standard

ARPANET – Advanced Research Projects Agency Network

API – Application Programming Interface

CPU – Control Processing Unit

CSS – Cascading Style Sheet

DAG – Directed Acylic Graph

DHT – Distributed Hash Table

GUI – Graphical User Interface

HTML – Hyper-text Markup Language

HTTP – Hypertext Transfer protocol

ID – Identifier

IPFS – Interplanetary File System

IPLD – Interplanetary Linked Data

IPNS – Interplanetary Naming System

Nonce – Number only used once

P2P – Peer to Peer

RAM – Random Access Memory

SFS – Self Certifying File System

SDLC – System Development Life Cycle

SHA – Secure Hashing Algorithm

TCP – Transmission Control Protocol

TTP – Trusted Third Party

UDP – User Datagram Protocol

UI – User Interface

TABLE OF CONTENTS

CHAPTER ONE

1.1 Background Study

In a research community, information sharing is an important step to gain maximum understanding from the previous work. Existing data sharing solutions rely on trusted third party(TTP). Such systems lack trust, transparency, security, and immutability as a result of TTP's involvement. To triumph over these issues, this project proposes a blockchain-based secure information security platform with the aid of using leveraging the benefits of interplanetary file system (IPFS). The proprietor uploads meta data to an IPFS server, which is then partitioned into n mystery shares. The proposed scheme achieves safety and get right of entry to manage by executing the access roles written in clever agreement with the aid of using proprietor. After the a success delivery of information, the user can access or files securely anytime with another node. Use of user registration is not required since all persons files in system are not explicitly confined to one blockchain but accessed on any node. In this scenario, decentralized storage, Blockchain, encryption, and incentive mechanism are combined. To enforce the proposed scenario, proof of work blockchain is written in python language and deployed when the user connects to a internet. The proposed scheme achieves transparency, safety, access control, authenticity of proprietor, and quality of data.

1.2 Statement of Problem

HTTP, for example, is an inefficient and costly technology. Instead of collecting bits from numerous computers at the same time, HTTP downloads a file from a single machine at a time. When streaming videos, we may save 60% on bandwidth expenses by using P2P technologies like IPFS. Another issue with today's internet is its vulnerability. Pages and websites can vanish in an instant. IPFS will allow for historic versioning (similar to github) while also making it easy to create up durable data mirrored networks. Lastly, the IPFS technology makes it feasible to distribute large amounts of data at a high rate of efficiency. We can also conserve storage space because data isn't duplicated.

1.3Aims And Objectives

Aims The main aim is to develop a file security system using blockchain and interplanetary files system(IPFS)

Objectives

i) Build a secure system for file storage and retrieval.

ii) To build a system that provide data authenticity and quality of data to customer

iii) The system aims to address the deficiencies of the client-server model and HTTP web through a novel p2p file sharing system**.**

1.4 Methodology

The system will be developed using the IPFS protocol. The IPFS layer will be added with ipfs client api.

The server is coded to run to get portability and speed.The web interface will be designed using the flask framework coded in Python. The User Interface will be designed using HTML,CSS and BOOTSTrap 3.

The backend of the project which will constitute the main blockchain will be implemented with python,the blockchain will be coded in python

Here are the main components:

1.4.1 Distributed Hash Tables

A hash table is a data structure that uses key/value pairs to store data. Data is disseminated over a network of computers and efficiently coordinated to provide efficient access and lookup between nodes in distributed hash tables (DHT).

The key advantages of DHTs are in decentralization, fault tolerance and scalability. DHTs can grow to handle millions of nodes because nodes do not require central coordination, the system can function consistently even when nodes fail or leave the network, and nodes do not require central coordination. When these properties are combined, the outcome is a system that is more resilient than client-server architectures.

1.4.2 Block Exchanges

By relying on a novel data exchange protocol, the popular file sharing system Bittorrent is able to successfully coordinate the transmission of data between millions of nodes, but it is limited to the torrent environment. IPFS implements BitSwap, which is a broader version of this protocol that acts as a marketplace for any type of data. This marketplace serves as the foundation for Filecoin, a peer-to-peer storage platform based on IPFS.

1.4.3 Merkle DAG

Merkle DAGs are a cross of Merkle Trees and Directed Acyclic Graphs (DAG). Merkle trees assure that data blocks sent through peer-to-peer networks are accurate, undamaged, and unchanged. This is accomplished by employing cryptographic hash functions to organize data chunks. This is a function that takes an input and generates a unique alphanumeric string (hash) that corresponds to it. It's simple to verify that a given input will produce a given hash, but guessing the input from a hash is extremely difficult. Individual data blocks are referred to as "leaf nodes," which are hashed to generate "non-leaf nodes." These non-leaf nodes can then be concatenated and hashed until a single root hash can represent all of the data blocks.

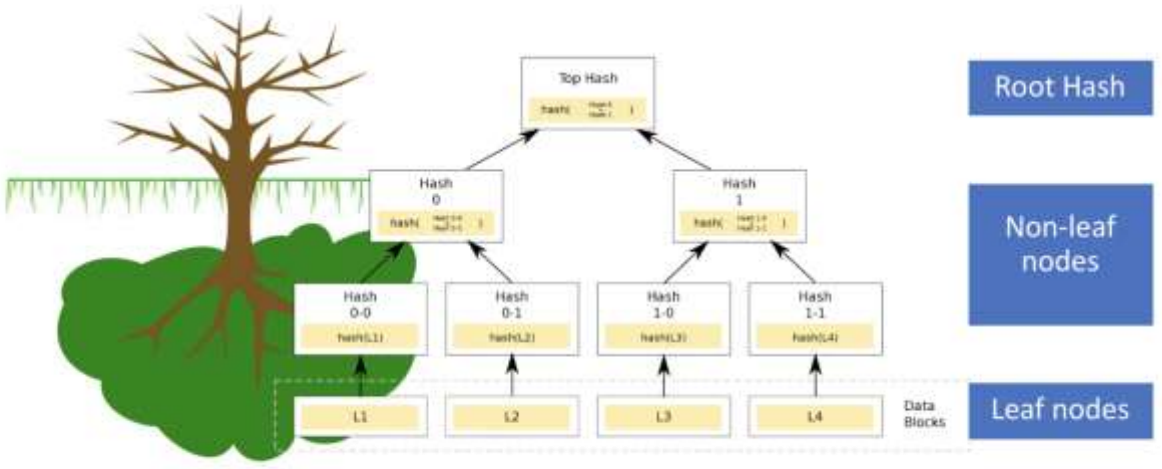


Fig 1 Merkle Dag Illustration V.Saini (2020)

A DAG is a mechanism to model topological information sequences with no cycles. A family tree is a simple illustration of a DAG. A merkle DAG is a data structure that uses hashes to refer to data blocks and objects in the DAG. This results in a number of valuable features: Because each data block has a unique hash, all material on IPFS may be uniquely identified. Furthermore, the data is resistant to tampering because changing it will change the hash, as illustrated below:

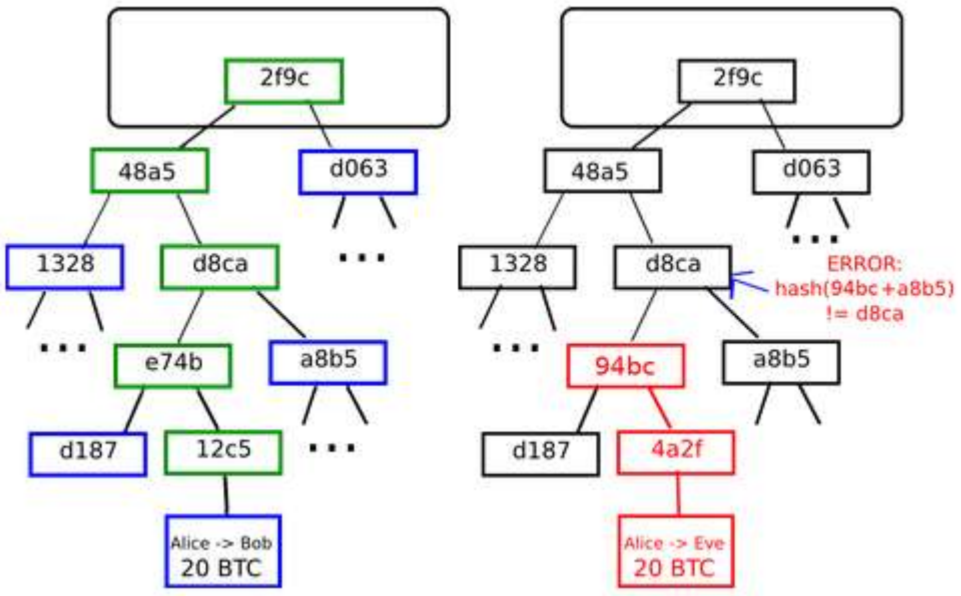


Fig 1.2 Merkle DAG self-certifying system. V.Saini (2020)

1.4.4 Self-certifying File System

The next key part of IPFS that we will examine is the Self Certifying System(SFS). It is a distributed file system which does not need specific data sharing rights. It is authenticated as data used by a client will be authenticated by the name of the file (which is signed by the server).

IPFS follows a similar approach for data objects: the full file history may be recovered as long as objects matching to the original data and any additional versions are available. Since data blocks are locally kept throughout the network and may be cached forever, IPFS objects can be permanently preserved.

IPFS also does not rely on Internet protocols for access. Overlay networks, which are essentially networks constructed on top of another network, can be used to distribute data. These characteristics are noteworthy because they are essential components of a censorship-resistant web. It might be a valuable instrument for encouraging free expression and combating the global prevalence of internet censorship, but we must be aware of the risk of abuse by unscrupulous actors.

1.5 Research and Modelling Plan

This project initially requires an extensive literature review which will form a context and provide necessary theoretical tools for the development of a Blockchain File Security System using Interplanetary File System (IPFS).

1.5.1 Literature Review

An extensive literature review will be undertaken spanning topics from Nakamoto, S: Bitcoin: A Peer-to-Peer Electronic Cash System. 2008. Crosby, M: Pattanayak, P.; Verma, S.;

Kalyanaraman, V. Blockchain technology: Beyond bitcoin. Appl. Innov. 2016, 2, 71.Wu, A.; Zhang, Y.; Zheng, X.; Guo, R.; Zhao, Q.; Zheng, D. Efficient and privacy-preserving traceable attribute-based encryption in blockchain. And Telecommunication. 2019, 74, 401–411.

Also, some free e-books would be consulted alongside free tutorials from Udemy to get knowledge of how to go about File security using Blockchain.

1.5.2 Implementation

I would be using Python alongside Flask library to implement the project. The AES encryption library will be employed along with Interplanetary File System web client API for encryption and file hashing.

HTML and JavaScript will also be employed to satisfy the UI needs.

CHAPTER TWO

LITERATURE REVIEW

This chapter discusses fundamental topics about the system. There exists a plethora of different peer-to-peer networks and technologies. The application implemented in this project is related specifically on file system security. Therefore, this chapter focuses on topics which are important in order to understand the concepts of decentralisation, peer-to-peer networking and file security.

2.1 Blockchain

Founded in 2009 by an an individual named Satoshi Nakamoto, Blockchain Technology is a breakthrough and has shown immense potential as being a fast, safe, and easy to use method for transferring or receiving funds.

Firstly, In a blockchain there is no “central entity” monitoring transactions like a Bank would. All details are kept private between the recipient and the sender and there is no “middleman” needed. A Blockchain is a “chain of Blocks”. Each block contains a “hash” an “index” and information about the particular transaction that took place. All the Blocks in the Blockchain are linked to each other with the “hash” variable. A “hash” contains information of the previous block in the chain and that’s what keeps the entire chain-linked and connected. S.Dante(2020).

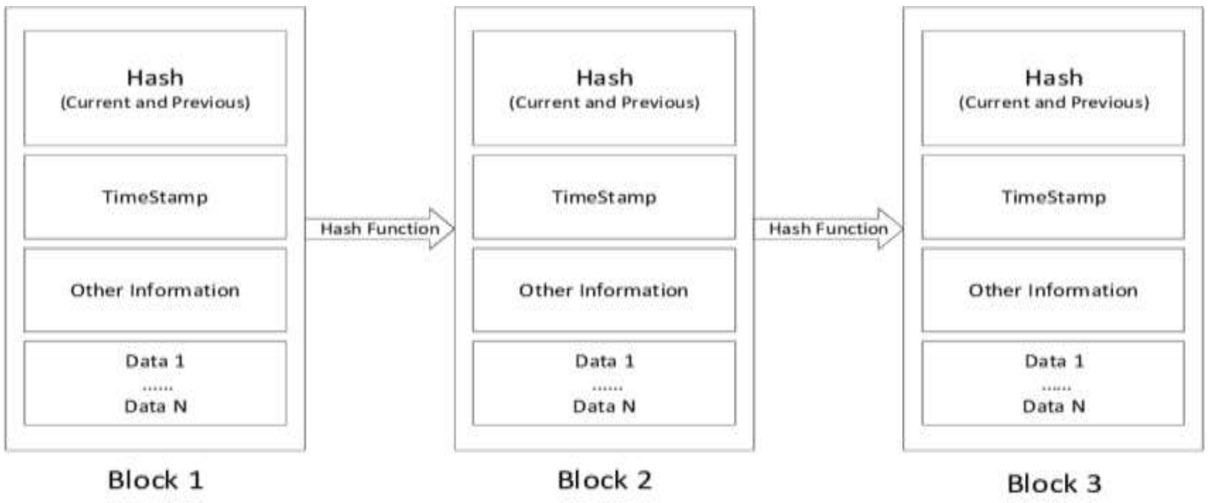


Fig 2.1 A Simple Blockchain S.Dante (2020)

Bitcoin was the first successful application of this system, and shortly after its rise in popularity, other cryptocurrencies were founded on the same principles. This system, however, is not restricted to storing financial information. Rather, the type of data being stored is inconsequential to and independent of the blockchain network.

Fundamentally, the data stored in a blockchain must have the following characteristics:

- Immutable
- Unhackable
- Persistent (no loss of data)
- Distributed

These qualities are necessary to maintain the integrity of the blockchain and the security of the network within which the transactions occur. S.Dante(2020)

Blockchain is seen as a distributed ledger that can be accessed globally by anyone to verify stored data with high integrity, resilience, credibility, and traceability. Distributed ledger technology can be used to write smart contracts which are self-executing contracts, and can be replicated, shared and supervised by a network of computers that run on blockchain. Smart contracts avoid middleman by automatically defining and enforcing rules and obligations made by the parties in the ledger. Blockchain, however is an expensive medium for data storage. For efficient storage of digital content, we can use InterPlanetary File System (IPFS) which is a peer-to-peer hypermedia protocol and distributed file system. Since IPFS is distributed, it has no single point of failure.

2.2 Network Architectures

The Internet has become the backbone of the modern society. Every day, more and more devices are being connected to it to be able share data and communicate more efficiently. The communication networks between different devices and services can be implemented in multiple different ways. These network architectures can be roughly split into two different types: centralized and distributed networks. The Figure 2.2 illustrates one possible approach to categorize different network models. The network model illustration in Figure 2.2 was created by Paul Baran in the 1960's. Nowadays, depending on context, the terms decentralized and distributed might be used to describe different things and sometimes they are used interchangeably. In this report the term decentralization refers to models where there is no central point of control or authority. One of the very first versions of the Internet, ARPANET, was fundamentally a distributed system. As the Internet evolved, the original distributed network was slowly replaced by a more centralized structure. Commonly, the

Internet connected devices run applications which are implemented by using client-server architecture L. Liu,(2010). The servers provide different services for the clients. The clients, for example mobile or desktop applications, make requests to one or more remote servers to access the services provided by the servers. The client-server architecture is a centralized architecture as illustrated in Figure 2.2 and it depends heavily on the servers to be available at any given time. As the client amount increases, server has to handle more requests. Consequently, server resource usage increases and the server might become overloaded. To prevent overloading, the amount of servers has to be increased or server has to be upgraded to one with more processing power and other resources.

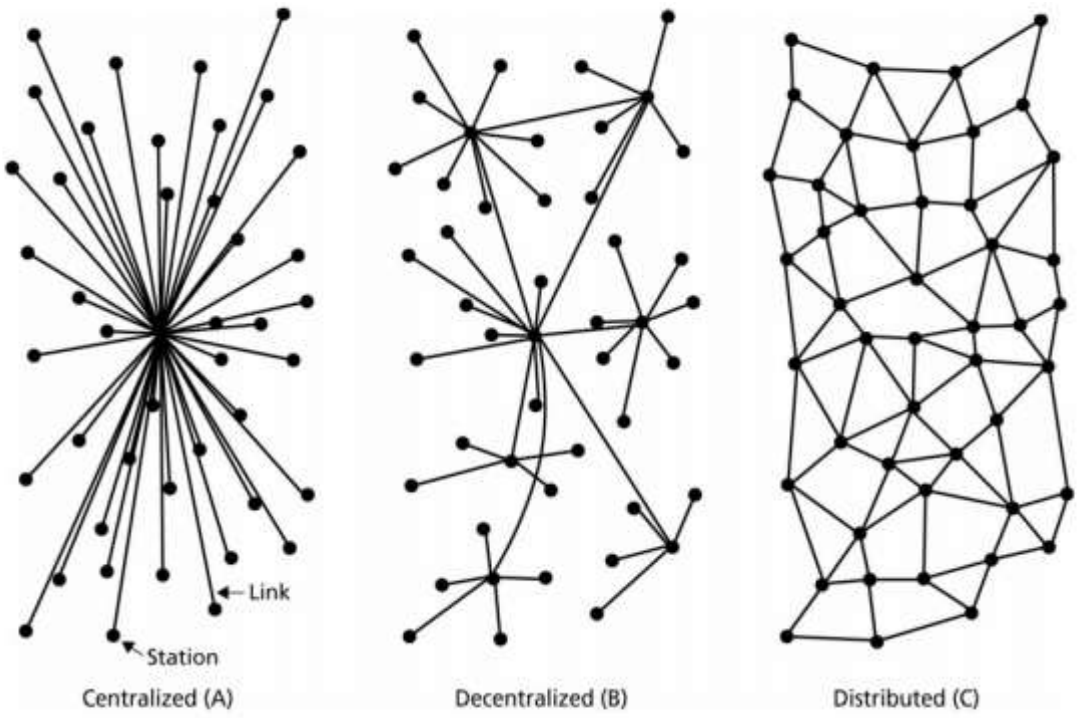


Figure 2.2 Differences between different network models J.Bufford (2009).

Due to the rise of cloud computing, physical on-premises servers have become rare. Cloud services allow to scale resources up easily or even automatically and thus make the scaling process relatively fast. This is often referred as serverless computing. A.Eivy ,(2017). However, hiding the servers from the users does not make the fundamental issue go away: the servers are still there and suffer from the very same centralization issues as before. Scaling resources up costs money. More importantly, there still exists single point of failure. The servers might become unavailable due to a number of reasons and the whole service provided by them is lost at that moment. Cloud has made this issue even more prominent. Large quantities of services depend on a handful of cloud providers which in turn rely on large centralized data centers. Amazon's AWS outage in North America in 2017 is a textbook example of single point of failure: an issue in one data center rendered hundreds of services unusable. S.Nichols,(2018) This issue could be avoided by distributing the centralized services which depended on that data center across the world to different data centers and effectively making the services less centralized. However, considering the issues caused by the AWS outage, geographical distribution is not often implemented.

In contrast to client-server architecture, peer-to-peer networking does not rely on centralized servers L.Liu (2010). Peer-to-peer networks are decentralized distributed systems in which every client, later referenced as node, acts as both client and server. Node can request resources from other nodes and in turn serve other nodes. Peer-to-peer networks form a new overlay network layer on top of the existing physical network infrastructure. Peer-to-peer networks avoid many drawbacks of the centralized client-server approach. There is no single point of failure which makes peer-to-peer networks resilient. Additionally, peer-to-peer networks are highly scalable: when the overlay network grows in size, the resource usage growth rate is less than linear J.Bufford and H.Yu(2010). Full decentralization also eliminates the need for central authority or service provider and effectively removes all hosting infrastructure expenses.

Peer-to-Peer Overlay Networks

Peer-to-peer networks are based on logical overlay networks. These overlays are implemented on the application level and they add an additional layer of abstraction on top of the existing physical networks. Overlays are virtual: they do not require any new equipment to be installed and there is no need to modify existing software or protocols. The overlay network topology determines the type of the peer-to-peer system. Peer-to-peer networks are often classified into unstructured and structured networks J.Bufford and H.Yu(2010).

The structure refers to the node and content placement in the network topology and on how the nodes communicate with each other. The peer-to-peer network properties such as performance and scalability depend on both the network topology and message routing algorithm. In unstructured overlay networks a node connects to its neighbor nodes which it knows about J.Bufford et al(2009). There is no predefined organized structure for node or data placement. Since a node knows about its neighbor nodes only it must communicate to the rest of the network trough the known nodes. Therefore, the structure of unstructured networks is often similar for example to random graphs or social networks. The content search in the unstructured overlay networks is implemented via some routing algorithm. Basic routing algorithms in unstructured networks are flooding and random walk J.Bufford et al(2009). Flooding works by sending a message, for example a search query, to all neighbor nodes. These nodes will then forward this message to their neighbors if they do not have the searched content available. This message has a time-to-live (TTL) value which is decreased in every node. The message propagation stops when this value reaches zero. Random walk works like flooding, but to decrease the message overhead, the message is sent only to one random neighbor. Unstructured peer-to-peer networks can be divided into three different categories with varying amount of decentralization L.Liu(2010).

These three network types are illustrated in the Figure 2.3. Centralized unstructured networks have central servers which contain information about other nodes. The nodes use this information to connect to each other. Hybrid unstructured networks use dynamic central entities called supernodes to form a unstructured network. Normal nodes then connect to these supernodes. Fully decentralized or pure unstructured peer-to-peer networks consist purely of nodes which are all considered equal.

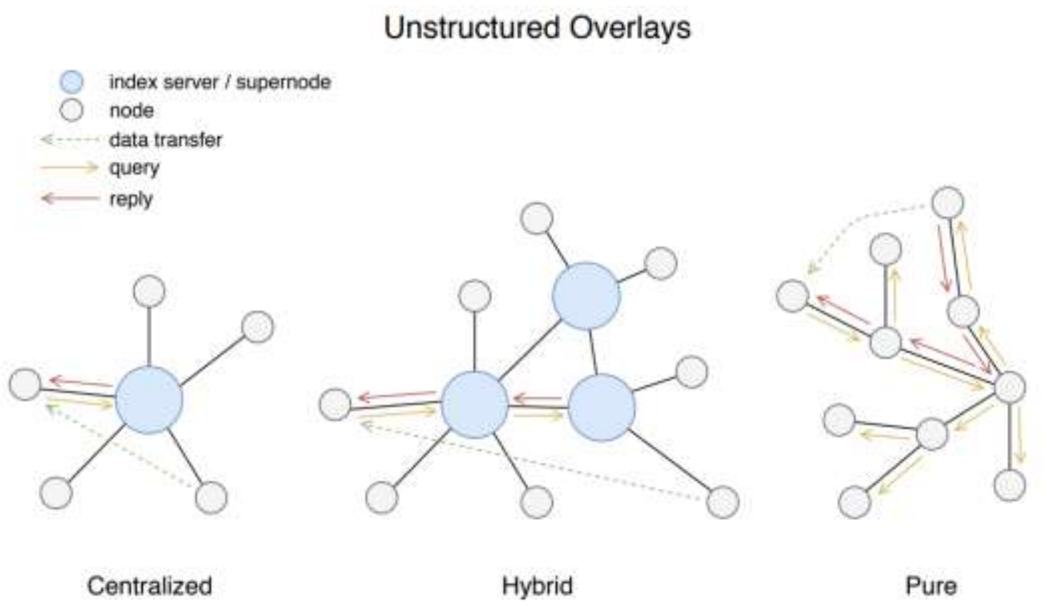


Figure 2.3 Unstructured overlay network types. J.Bufford (2009).

In contrast to unstructured networks, structured networks enforce some rules for the node and content placement in the network topology J.Bufford et al(2009). Structured overlays can be categorized by the overlay network geometry and routing algorithm. The geometry defines the graph structure between nodes in the network. Certain geometry may support multiple different routing algorithms. When a node joins to the structured network it will initialize a local routing table. This table is used by the routing algorithm to find content. Additionally, the node is given a virtual address from the virtual address space of the structured network. The address of node defines the place of the node in the overlay network geometry. The routing table and virtual structure enables deterministic routing and makes content search in the overlay network more efficient than in unstructured networks. However, the overlay network structure does not reflect the actual physical network structure. While this increases fault tolerance, the overlay network latency will increase if the nodes are distant in the physical network topology. J.Bufford and H.Yu(2010). Most of the structured peer-to-peer networks use key-based routing algorithms K.Dhara et al(2010). Distributed hash table (DHT) is often used to form structured overlays and to enable key-based routing. Each node in the network is responsible for a piece of the hash table according to some partitioning method. Every node is assigned an identifier (ID) which defines the range of the table it is responsible for. When some data is added to the network, a hash is computed for example from the filename of the data. The hash along with the data is stored in the overlay network to a node which is responsible for storing the content. This node is determined by the calculated hash and node ID. A simplified illustration of a structured network can be seen in Figure 2.4.

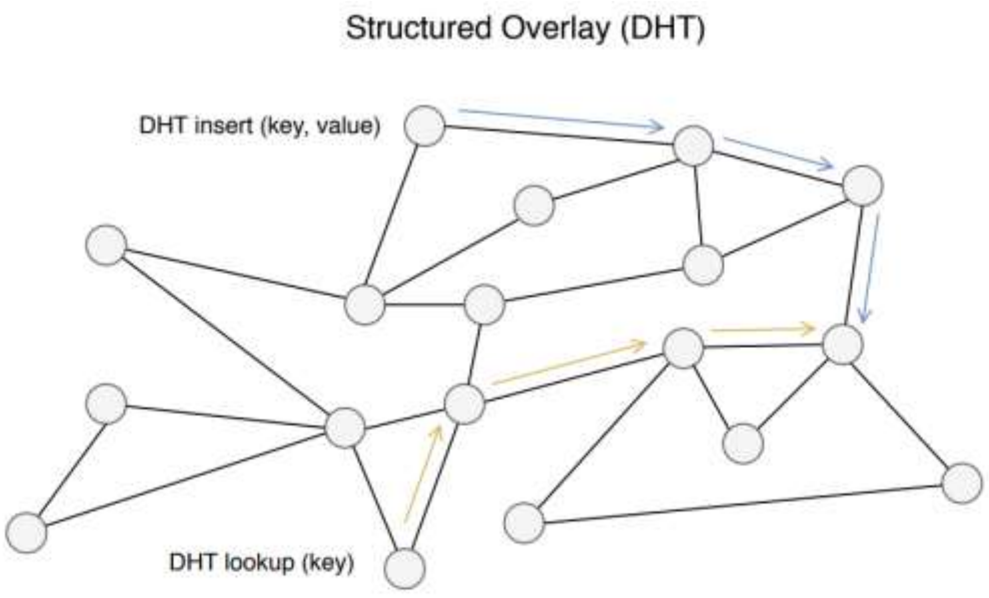


Figure 2.4 Structured overlay network J.Bufford (2009).

After the content has been added to the structured network, it can be searched with the hash K.Dhara et al(2010). The routing algorithm then finds the node which has the content by forwarding the query to a node which is closest to the target node's ID. DHTs drawback is that the routing tables must be maintained due to changes in the network. The process of constantly joining and leaving nodes is called as churn. Churn is common property for all peer-to-peer networks but in the case of DHT high churn rate means that the overlay network must constantly adjust the routing tables.

2.3 Peer-to-Peer File Sharing

Many of the original peer-to-peer network implementations were focused on file sharing. Many of them still exist and are being developed further. To understand the incentives behind modern peer-to-peer networks and decentralized solutions, some of the popular large scale file sharing peer-to-peer networks are introduced next in more depth.

One of the first large scale examples of peer-to-peer networks was file sharing system Napster L.Liu(2010). Napster utilized a centralized unstructured approach where nodes transferred files between each other but the file directory was stored in a central server. The centralized file directory contained metadata about the files and the nodes which had those files available. Nodes could then query the server for the files and initiate a file transfer directly with the node which had the file.

2.3.1 BitTorrent

BitTorrent is another more advanced example of the unstructured yet partly centralized approach. There are central servers, trackers, which maintain node addresses which are sharing some file. To share a file, a special .torrent file must be created which contains the tracker address. These files can be published for example on a web site. BitTorrent client software then uses the .torrent file to initiate connection to the tracker. Using the information provided by the tracker server, client connects to the other nodes and starts the file transfer. The files are broken into smaller chunks in order to download the file from multiple nodes simultaneously. This enables also the possibility to start uploading the already downloaded chunks to other nodes before the whole file has been downloaded. B. Cohen(2003).

The availability of files in BitTorrent is short-lived, since when nodes stop sharing certain file, it becomes unavailable. A study has shown that 38 % of the torrents become unavailable within the first month S. Kaune et al(2009). In addition, the BitTorrent protocol has so called free riding problem: users download files but don’t share them. While the BitTorrent protocol has countermeasures against free riding, it is possible to download files without contributing anything back T.Locher et al(2006). Despite these issues, BitTorrent is still one of the most popular file sharing peer-to-peer networks.

Like in any other centralized architecture, central servers introduce single point of failure to the system also in partially centralized peer-to-peer networks. Similarly, the central metadata or tracker servers cannot be scaled efficiently. . To address the issues of the partially centralized approach, fully decentralized peer-to-peer networks were developed.

2.3.2 Gnutella

One of the first fully decentralized peer-to-peer networks was Gnutella To connect to other nodes in the Gnutella network a client application is needed. The client application tries to establish connection to a predefined amount of nodes upon first startup and assigns found nodes as neighbors. L. Liu, N. Antonopoulos(2010).

Gnutella was not the only fully decentralized peer-to-peer network which emerged in the early 2000’s. Freenet is a decentralized file sharing system which has a special focus on encrypted data storage and strong anonymity [13]. One major difference to the Gnutella procotol is that all the nodes store not only their own content but also other content from the network. Due to the increased redundancy the network becomes more fault-tolerant. Freenet does not support arbitrary file searches. One-way secure hashes are used to uniquely identify different files. The file contents are encrypted: nodes are aware only of the file hashes but not the actual content of the files. To get a file from the Freenet network one must know the hash of the file. Freenet's routing algorithm resembles distributed hash tables but is less structured. I. Clarke (2001).

2.3.3 Freenet

In Freenet each node maintains a routing table with known file hashes and node addresses. A node uses the table to forward the search query to a node which it assumes to be closest to the owner of the file determined by the lexicographical distance of the hash. The nodes are aware only of their immediate neighbor nodes. This makes the network more secure, because the nodes never know where the data is stored. The actual file transfer is not done directly with the node which has the file but through the Freenet network to protect the identity of the data owner node.

2.3.4 Kazza

Kazaa is another example of the hybrid unstructured network L.Liu(2010) . In Kazaa,Supernodes act as temporary index servers for the normal nodes. Kazaa client software is needed to join the network. The client offers a list of supernodes to connect to. After connecting to some supernode, the normal node sends a list of files it is sharing to the supernode. When a node requests some file, supernodes look first their local index for the file. If the file is not found, the supernode will query the file from the other supernodes. L. Liu, N. Antonopoulos(2010).

2.4 Trust, Security and Privacy in Peer-to-Peer

Networks In the client-server architecture most of the security and data privacy concerns rely on the central authority. The central authority is trusted to keep for example private data safe and to otherwise protect the data which have been stored in the central servers. In peer-to-peer networks there is no central authority and thus every node trust the other nodes with which it is communicating. Mutual trust among nodes is an important aspect of peer-to-peer systems.

In content addressed networks such as Freenet there is a smaller change to receive wrong content due to the one-way hashes which identify the content cryptographically. If the content is identified by some other means, a malicious actor could easily spoof the content and distribute for example bogus data instead of the real file [9, p. 338]. Even in content addressed networks one must trust the original distributor of the content hash since it is impossible to know what the file behind the hash contains in reality.

Privacy is another important aspect of peer-to-peer networks and is closely related to both trust and security. The identity of the user is not usually protected in peer-to-peer networks in any way which enables for example user tracking J. Bufford (2009).Some networks such as Freenet have been built so that they offer enhanced privacy and protection. Regardless, users must take care when adding content to the peer-to-peer network to avoid exposing sensitive information with the rest of the world.

Johnson et al.(2009) argue that many issues related to security and privacy can be mitigated with good user interface design and user education. According to the authors, sensitive information gets commonly exposed for example due to a misplaced file, confusing user interface design or automated wizards. The user interface must be designed in a way which clearly shows what files are being shared. In the end, users must be educated to understand the risks of peer-to-peer file sharing and to use file sharing applications with care.

The early peer-to-peer file sharing systems were based on the unstructured approach. Nevertheless, numerous modern peer-to-peer file sharing systems utilize DHT-based structured networks.

2.5 Modern Peer-to-Peer File Sharing

Many new peer-to-peer content sharing networks have emerged in the last few years. Most prominent of these systems are InterPlanetary File System (IPFS) J.Benet (2014), Dat and Swarm. These three systems were selected to this comparison due to their popularity, open source implementations and the large development efforts behind them. Based on this comparison one of them will be selected to be used as the back end in the application implemented in this project. InterPlanetary File System is a peer-to-peer distributed file system J.Benet (2014). Its ultimate target is to replace Hypertext Transfer Procotol (HTTP) and build a new decentralized Internet infrastructure. IPFS combines multiple different proven techniques of past successful distributed systems to create a new solution which attempts to connect all computing devices under a single worldwide file system. Among these are for example Kademlia DHT, BitTorrent and distributed version control system Git. The main idea of IPFS is to model all data in a single Merkle directed acyclic graph data (Merkle DAG) structure. Dat is a peer-to-peer protocol which is designed for syncing versioned data.M Ogden (2017). It attempts to replace cloud based storage services such as Dropbox and Google Drive due to their centralized nature, high costs, slow transfer speeds and privacy issues. Dat has many similarities with IPFS: it is inspired by projects such as Git and BitTorrent. However, Dat’s main idea is narrower: it offers a free and encrypted versioned peer-to-peer data syncing system. Similarly to IPFS and Dat, Swarm is a distributed platform which provides peer-to-peer data storage services. Swarm is closely related to the Ethereum blockchain ecosystem. Its primary target is to serve as decentralized and redundant storage system for all data related to Ethereum. Unlike Dat and IPFS, Swarm does not require that the uploaders host the content themselves. This is possible due to a built-in incentive system which rewards nodes that are hosting content.Swarm (2018).

The main implementations of IPFS and Swarm are written in Go. Go is an open source programming language. It makes it easy to run both IPFS and Swarm almost on any device. In addition to Go, IPFS has also a JavaScript implementation which makes it possible to run IPFS in web browsers. The development efforts of Dat are focused solely on JavaScript. Like IPFS, Dat can be run on web browsers. Outside browsers Dat needs some JavaScript runtime or browser based environment to be able to run. This is a drawback, since it makes it harder to develop applications especially for mobile devices due to the special runtime environment needs. From portability point of view Go is a lot better choice than JavaScript: compiled Go applications produce a single binary and if needed, applications written in Go can be compiled into standard shared libraries. This allows Go applications to be used easily as a library directly from other programming languages. It is also straightforward to cross-compile Go programs to other processor architectures which is often the case when targeting mobile environments. While all the three projects are similar on higher level, their approaches to peer-to-peer content sharing are very different when observed more closely. Swarm is part of the large Ethereum ecosystem but it means that it requires Ethereum to work properly. Additionally, Swarm is not as mature as the other two projects. Dat has had multiple stable releases, but JavaScript makes it hard to integrate it into resource constrained mobile environments. The remaining candidate is IPFS. Go-ipfs, the reference implementation of IPFS, can be considered stable enough for this project’s application

2.6 InterPlanetary File System

InterPlanetary File System is a peer-to-peer network protocol which uses has been originally developed by Juan Benet (2018). IPFS integrates multiple different previously proven techniques together to create a distributed and versioned file system with no single point of failure.

The IPFS Protocol is divided into a stack of sub-protocols responsible for different functionality:

Identities - manage node identity generation and verification.

Network - manages connections to other peers, uses various underlying network protocols.

Routing - maintains information to locate specific peers and objects. Responds to both local and remote queries. Defaults to a DHT, but is swappable.

Exchange - a novel block exchange protocol (BitSwap) that governs efficient block distribution. Modelled as a market, weakly incentivizes data replication. Trade Strategies are swappable.

Objects - a Merkle DAG of content addressed immutable objects with links. Used to represent arbitrary data structures, e.g.file hierarchies and communication systems.

Files - versioned file system hierarchy inspired by Git.

Naming - A self-certifying mutable name system.

The IPFS stack can be divided into five sections.

At the bottom there are three different layers: networking, routing and data exchange. These layers have been implemented in a separate library called libp2p. The library provides modularized and extensible networking stack for peer-to-peer application development. The library was originally developed as a part of IPFS, but it was moved into external library to be able to leverage it also on other systems.

The first layer, networking, contains support for point-to-point transport between peers via protocols such as transmission control protocol / internet protocol (TCP/IP) and user datagram protocol (UDP).

The next layer is the routing layer. The routing layer is responsible for enabling mechanisms to locate other nodes and content in the network. A solution based on Kademlia DHT has been selected in IPFS as the routing layer implementation. However, the routing layer is implementation agnostic and the distributed hash table could be replaced with some other routing implementation. J.Benet (2014).

The third and last layer included in libp2p is the data exchange layer. This layer is responsible for negotiating how the actual data transfer between nodes works. In IPFS the data block exchange protocol is BitSwap which is a new solution inspired by BitTorrent. Like the routing layer, also this layer is implementation agnostic and could be replaced with some other data exchange protocol. On top of the three bottom layers there are the data structure layer and the naming system layer. The data structure layer is implemented as a Merkle DAG. Additionally, the files are versioned with a object model similar to the one used in Git. The underlying data model which describes these structures is implemented as a common hash-chain format called InterPlanetary Linked Data (IPLD). Merkle DAG is the main concept of IPFS: it creates links between the stored objects via cryptographic hashes. Merkle DAG makes the content in IPFS immutable and content addressed. Due to this, content authenticity verification is simple. Additionally, objects with the same content are considered equal and stored only once.

In the naming system layer, Interplanetary Name Space (IPNS), enables mutable naming of the stored objects. This is done by creating mutable pointers to the Merkle DAG. The naming scheme is inspired by Self-certifying File System (SFS). Every node is assigned a mutable namespace defined by the hashed node ID. Users can then publish an object under this namespace which points to the actual content in the Merkle DAG. The content is signed with the node's private key and thus the authenticity of the content can be checked with the public key and node ID. J.Benet (2014)

Applications can be implemented on top of the IPFS protocol. Go-ipfs is the main reference implementation of the IPFS protocol. Besides being able to run a full IPFS node, go-ipfs can used also as an application to control the node. For this purpose go-ipfs includes a command line client, an application programming interface and a web-based graphical user interface. An overview of the described architecture is illustrated in the Figure 2.5

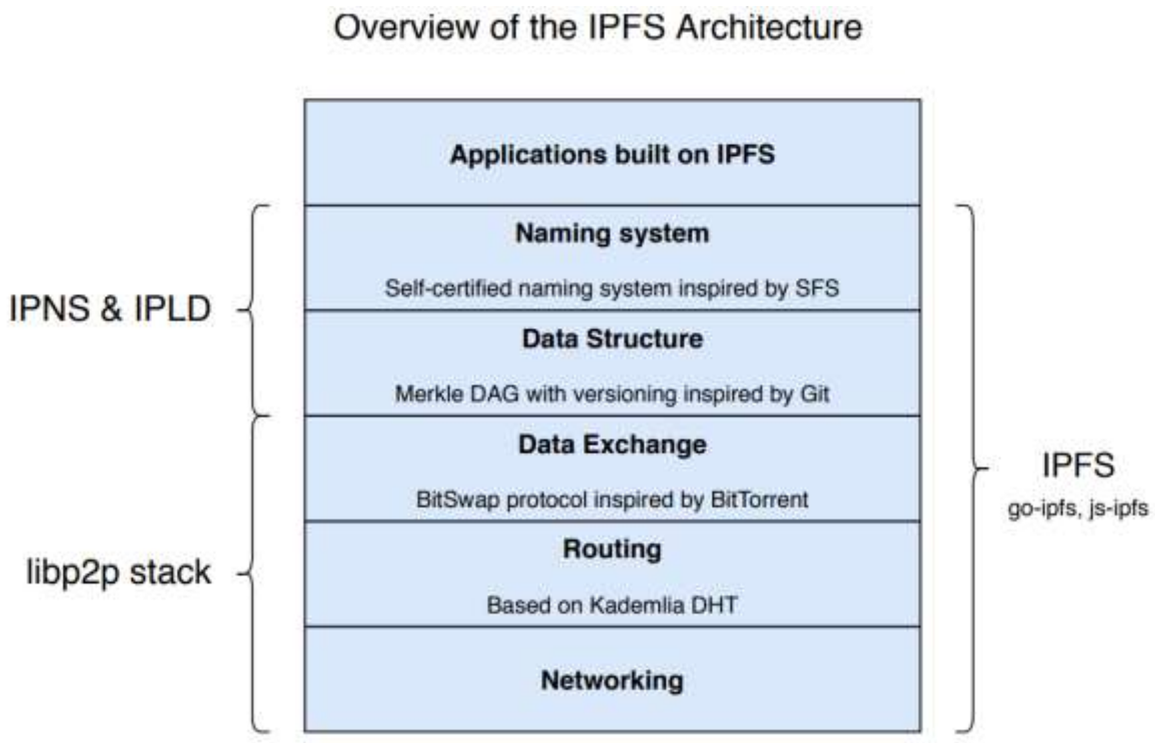

Overview of the IPFS Architecture
Applications built on IPFS
Naming system
Self-certified naming system inspired by SFS
Data Structure
Merkle DAG with versioning inspired by Git
Data Exchange
BitSwap protocol inspired by BitTorrent
Routing
Based on Kademlia DHT
Networking
IPNS & IPLD
libp2p stack
IPFS
go-ipfs, js-ipfs

Figure 2.5 Overview of the IPFS architectural stack. J. Bufford (2009).

2.7 Related Works

In recent era, blockchain such as; bitcoin and Ethereum are considered as hot and fundamental technologies of cryptocurrency. As a result, researchers and industrialists are paying more attention to establish a trust based model in a decentralized manner. In this section, few studies regarding blockchain are presented.

The need for peer-to-peer content sharing applications was first recognized in a user study conducted by Matuszewski et al(2007). in Finland. In the study the attitude towards mobile peer-to-peer content sharing applications was analysed. The survey results suggested that there is a need for such applications. However, the study revealed that free riding attitude is common even if the only cost of running a peer-to-peer application was higher battery usage. Most of the users were willing to run the peer-to-peer application only for short periods of time given that they are familiar with the people they are sharing content with.

Peer-to-peer usage in Finland was measured by Heikkinen et al. in 2007. TCP/IP traffic was measured in the GSM/UMTS networks of three major operators. Different peer-to-peer applications were identified by transport protocol port numbers. As a result, direct peer-to-peer traffic was not observed but instead 9-18 % unidentified traffic was detected. This unidentified traffic was categorized to be possibly originating from peer-to-peer applications. In addition to direct network traffic analysis, a panel study was conducted. The study revealed that only one mobile peer-to-peer client application, Fring, had significant usage levels across all participants.

In order to preserve the privacy for traceable encryption in blockchain, Wu et al. in (2019). proposed a system in which authenticity and non-repudiation of digital content is guaranteed. The problem tackled by authors is the secret key of user, which when shared with other entities does not hold the specific information of user. In case the shared key is corrupted or abused, it makes difficult to analyze the source of secret key. Moreover, leakage of confidential information in access control is a bottleneck for existing systems. Therefore, authors have integrated the privacy protection algorithm such as attribute based encryption (ABE) to secure the secret keys. However, decryption mechanism does not show improved efficiency.

Management of digital data rights is a fundamental requirement to achieve protection of digital data. Existing techniques for data rights lack transparency, decentralization, and trust. In response to above mentioned problems, Zhang and Zhao (2018) proposed blockchain-based decentralized solution. Information regarding the use of digital content, such as transaction and license information is transparent to everyone. Smart contract is designed for the automatic assignment of license. In this mechanism, owner can set the prices for selling the license to other customers. However, peers of the network have to possess high computational power to perform key acquisition.

The BitTorrent protocol and different implementations based on it has been studied widely in mobile environments in many different contexts. These studies include the creation of a mobile BitTorrent application by Ekler et al. (2008), power consumption study by Nurminen et al. (2008), proxy-based approaches on content downloading , the analysis of BitTorrent protocol reliability, hybrid BitTorrent based peer-to-peer content sharing solution and BitTorrent client usage and content lifecycle analysis.

Ekler et al. (2008). implemented a mobile BitTorrent client application for low end Nokia Series 40 based devices. The results show that even simple mobile devices are capable of running a full peer-to-peer node utilizing existing peer-to-peer networks

Bori et al. (2013) studied the reliability of BitTorrent protocol in mobile environment. The study was based on a mobile BitTorrent application called DrTorrent. The application had 5000 active users at the time of the evaluation of the statistics. Bori et al. conclude that BitTorrent is generally realiable, but the client applications must be capable of handling different anomalies such as failed TCP connections and corrupted data.

A hybrid BitTorrent based content sharing solution has been also studied: Ekler et al. (2008). proposed a new content sharing solution based on BitTorrent and central servers controlled by mobile operators. The solution supports both mobile and PC clients. The central server and operator's central unit help to distribute the content more efficiently which in turn helps to reduce service provider costs.

In current digital era, size of data is increasing every day with a significant amount. This brings a storage issue at users' terminals. The solution to this could be leveraging these storage limited nodes with cloud services. However, this solution comes with a cost of compromised trust and security, as it is difficult to rely on third party. In consideration to afore mentioned problems, a secure distributed storage architecture is proposed by Li et al (2018), in which data is divided into multiple encrypted chunks, and randomly stored on peer to peer nodes of network, which voluntarily offer storage services. A genetic algorithm is applied for placement of file blocks among multiple users.

The general performance of DHT-based peer-to-peer overlays in mobile environments was measured by Chowdhury et al. (2017). Five different DTHs were analyzed: Chord, Pastry, Kademlia, Broose and EpiChord. The measurements were done in an simulated environment. According to the results, the best choice for peer-to-peer overlay under heavy churn is Kademlia due to its good 97 % lookup success ratio while consuming 316 bytes/s of bandwidth. Additionally, EpiChord performs also well by achieving a 63 % success ratio while consuming only 70 bytes/s of bandwidth.

Cherbal et al. (2017). presented an improved Chord-based structured peer-to-peer approach to minimize physical distance of the nodes. The approach is based on two-tier structure where main Chord ring contains supernodes and secondary Chord ring contains cellular nodes. The authors provide a mathematical analysis of the approach which proves that the approach is more efficient due to reduced physical distance of the nodes.

Clouds are centralized servers which provide services for data storage and sharing between different stakeholder. In Xia et al’s (2017) work, sharing of medical data of patients between hospital management is secured using blockchain. End-to-end delivery of medical data needs to be encrypted to handle any misuse by third party. A blockchain-based data sharing platform is designed for multiple services providers on cloud. Smart contracts are written to automate secure sharing and distribution by employing access control strategies by data owners. This scheme also provides transparency and traceability. Performance of proposed scheme is analyzed by making a fair comparison with the existing schemes. However, there is still a scope to improve security and fine grained access control through ABE.

Data sharing is a crucial step to gain maximum benefit from the strengths of research. A lot of data sharing mechanisms are proposed and discussed in literature. There is no sufficient work available that focuses on the incentive mechanism to promote data sharing. To cover these limitations, Rowhani et al (2017) conducted a review on medical and health data to uncover the incentive mechanisms with the pre- and post- results after empirical analysis. According to survey, a single incentive is tested for medical and health data to analyse the rate of data sharing. Therefore, it is concluded that more incentive based researches need to be performed to encourage data sharing.

In summary, the previous work has studied peer-to-peer networking in various different perspectives including topics such as peer-to-peer usage level measurements, mobile BitTorrent clients, power consumption measurements and different DTH-based systems. This projet aims to extend the previous research by providing a fresh practical approach on peer-to-peer networking for file security. Further, by creating an actual application on top of an existing modern peer-to-peer network the behavior and performance of the system can be evaluated in real environment.

CHAPTER THREE

METHODOLOGY

This chapter focuses on defining the client application implemented as a part of this project. First, the use cases for the application are discussed. Based on the use cases a list of requirements are derived. The requirements are evaluated from the standpoint of this thesis and from the standpoint of end users.

3.1 System Architecture

As Blockchain is the core technology to be adopted in this project, the platform will be built using a proof of work blockchain, where smart contract will be designed and implemented to control the files added to the blockchain. To satisfy the principle of decentralization, these data will not be stored centralized in the platform, decentralized storage will be implemented. The platform will not store data onto the blockchain, instead, objects of data will be stored through distributed storage by adding the file hash to the blockchain to the to initiate or add to the existing blocks present.

Currently, there are protocols and services available in the market for distributed object storage and distributed file system. The considered system architecture is comprised of three layers: Client layer, Consensus layer and Data storage layer.

3.1.1 Client Layer

The major responsibility of this layer is to govern the interactions between user and the system, which mainly serves as a user interface. There will be a web interface for the user to interact with the system.

3.1.2 Consensus Layer

Consensus Layer is implemented with interaction with the Interplanetory File System(IPFS). This layer is responsible for encrypting the file and adding it to the blockchain.the proof of work verifies the file before adding to the blockchain.

3.1.3 Data Storage Layer

This layer is responsible for data storage, where data are transferred to this layer through consensus layer and the client layer. The technology which will be used in this project in this layer is Interplanetory File System (IPFS).

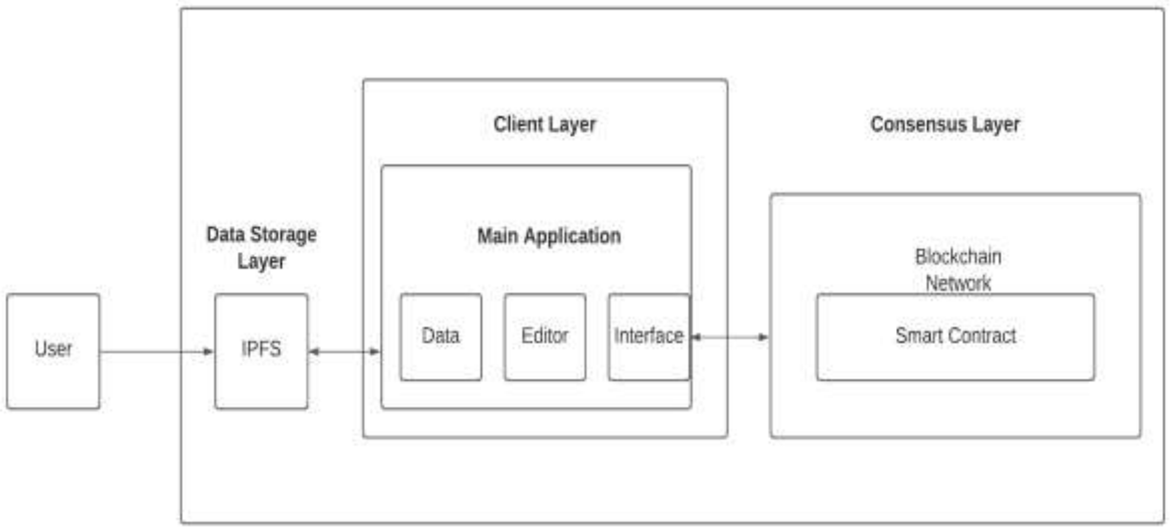


Fig 3.1 System Architecture Diagram

3.2 System Development Life Cycle

For the system that is to be implemented we firstly define the needs for the development of the system and what will be needed in the to make it tackle the existing problem effectively.

The present system would be studied thoroughly to assess the strengths and weaknesses. Here, the use of facts finding techniques can be re applied to determine the needs of the new system either from the users

The SDLC models are considered as the foundation software engineering tools to help facilitate better delivery of our software project. SDLC models are predefined set of phases to lay out the common understanding of the complete picture of the development process, from conceiving the concept to final delivery and continued maintenance of the applications. SDLC delineate and pre-establish the way in which the software project will be comprehended and then developed deriving from the business insights and requirement analysis phase to materialise them into features and functions of a software or application – until its operation and application to satisfy and achieve the business needs. Consequently, to assure successful completion of the software application, it is very important to select the most appropriate Software Testing life cycles and SDLC model and act in accordance with it, following the requirements and concerns of the project.

3.2.1 Incremental Model

Iterative and Incremental SDLC Model is a cyclical process blended approach – The combination of iterative design/method with incremental build model, After the completion of the inceptive planning stage, a small handful of steps are recurrently repeated with the aim to incrementally improve the software at each iteration or completion of cycle.

The iterative process begins with a simple implementation of a small set of the software requirements and iteratively enhances the evolving versions until the complete system is implemented and ready to be deployed.

In incremental model the whole requirement is divided into various builds. During each iteration, the development module goes through the requirements, design, implementation and testing phases. Each subsequent release of the module adds function to our previous release. The process continues till the complete system is ready as per the requirement.

During development, a model is designed, implemented and tested incrementally until the final product is finished. The method being implemented combines the elements of waterfall model and iterative philosophy of prototyping, also with gradual implementation we provide ability to monitor the effects of incremental changes to our system.

The various components are decomposed into a number of smaller components which are designed and built separately. Each component is then used together to achieve final project solution.

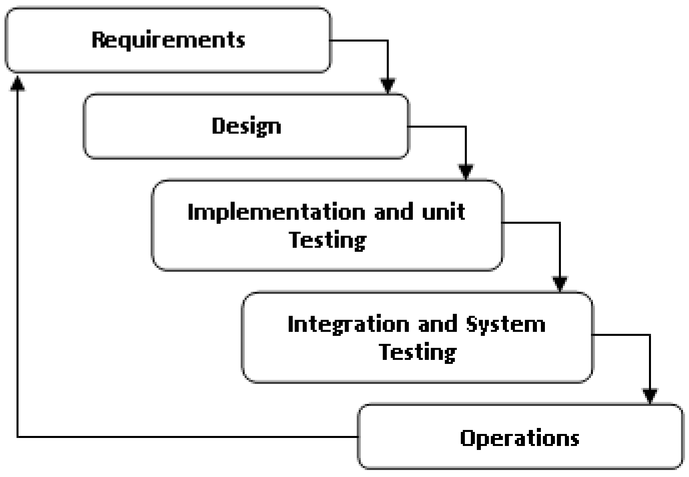

Requirements
Design
Implementation and unit Testing
Integration and System Testing
Operations

Fig 3.2 Incremental Model

3.3 Use Cases

A use case is a system used which will be employed to identify, clarify and organize our system requirements. The use case is made up of a set of possible sequences of interactions between systems and users in the system interface and related to a particular goal. The use case is a method of description of all the ways an end-user would make use of our proposed system

3.3.1 Uploading Files

File storage is the main use case for a common user. A user must be able to add a file or a folder into the IPFS via the application. After adding the file, user must be able to see and copy the content identifier of the added content to be able to retrieve the added content another time.

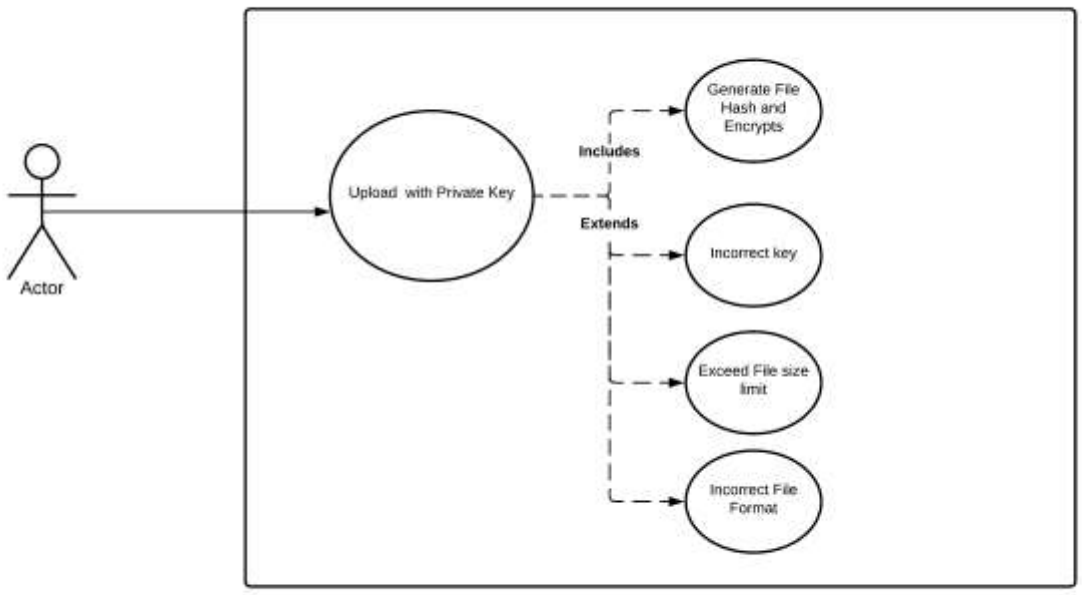


Fig 3.3 Upload Use Case Diagram

3.3.2 Retrieving Files

Given an IPFS content identifier i.e the file hash, user must be able to retrieve the data behind it through the application. The user must be able to insert a private key that the user used to upload the file to download it securely.

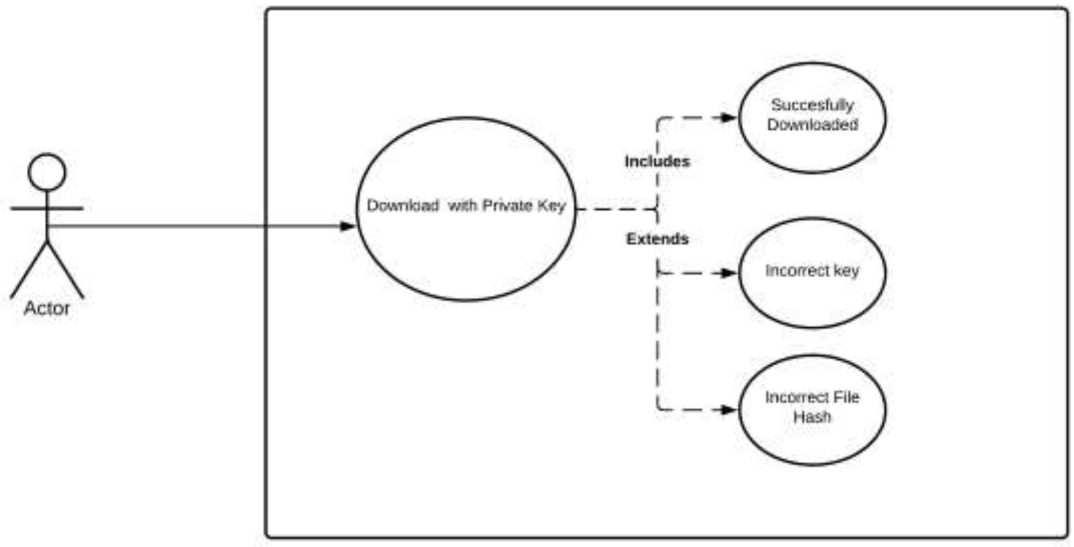


Fig 3.4 Download Use Case

3.3.3 Viewing the blockchain

User will be able to see the blockchain and view all the blocks of data that are present in it.

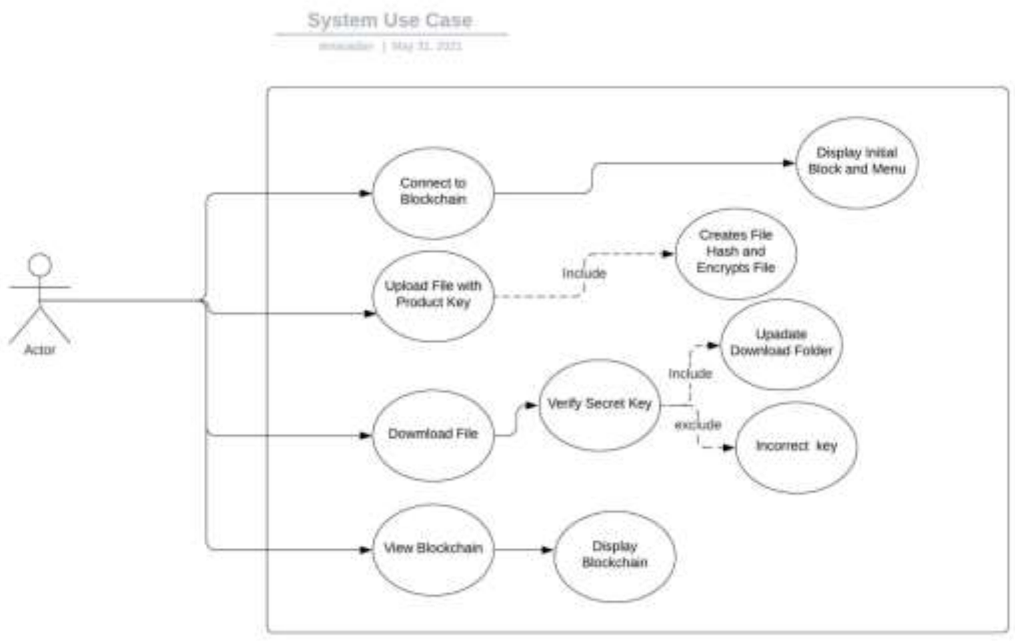


Fig 3.5 System Use case

3.4 Activity Diagrams

The activity diagram visually represents the series of actions or flow of control the system similar to a flowchart or a data flow diagram. They can also describe the steps in a use case diagram.

3.4.1 Description of System Activities

In the systems activity, the user will be introduced to the system user interface with the home page. The User can decide to learn about the functions of he system by using the navigation bar to access the services information or navigate to other features.

If the user decides to securely upload a file they will click the “Connect to the Blockchain” menu and the web browser will take them to the specified page.

The following interface will give the user options to upload a new file, download a new file or disconnect from the blockchain. The user can add a new file to the blockchain by interacting with the “Upload a new file”option.They will input their names and their unique secret key that will be used to encrypt the file, when the user clicks on upload the file is encrypted and Ipfs turns it into a unique hash that is then added to the blockchain. The user can view their file in the blockchain as a unique file hash which they can copy to access the file at a later time.

The user can retrieve their file from the blockchain by imputing the unique file hash that was automatically generated when they uploaded the file along with their unique private key in the “Download File” menu to get the file downloaded to their laptop

If the user wishes to disconnect from the blockchain they can access the “Disconnect from the blockchain” menu to disconnect from the blockchain. The blocks in the blockchain will still be available if the user wants to connect again

3.4.2 Activity diagrams

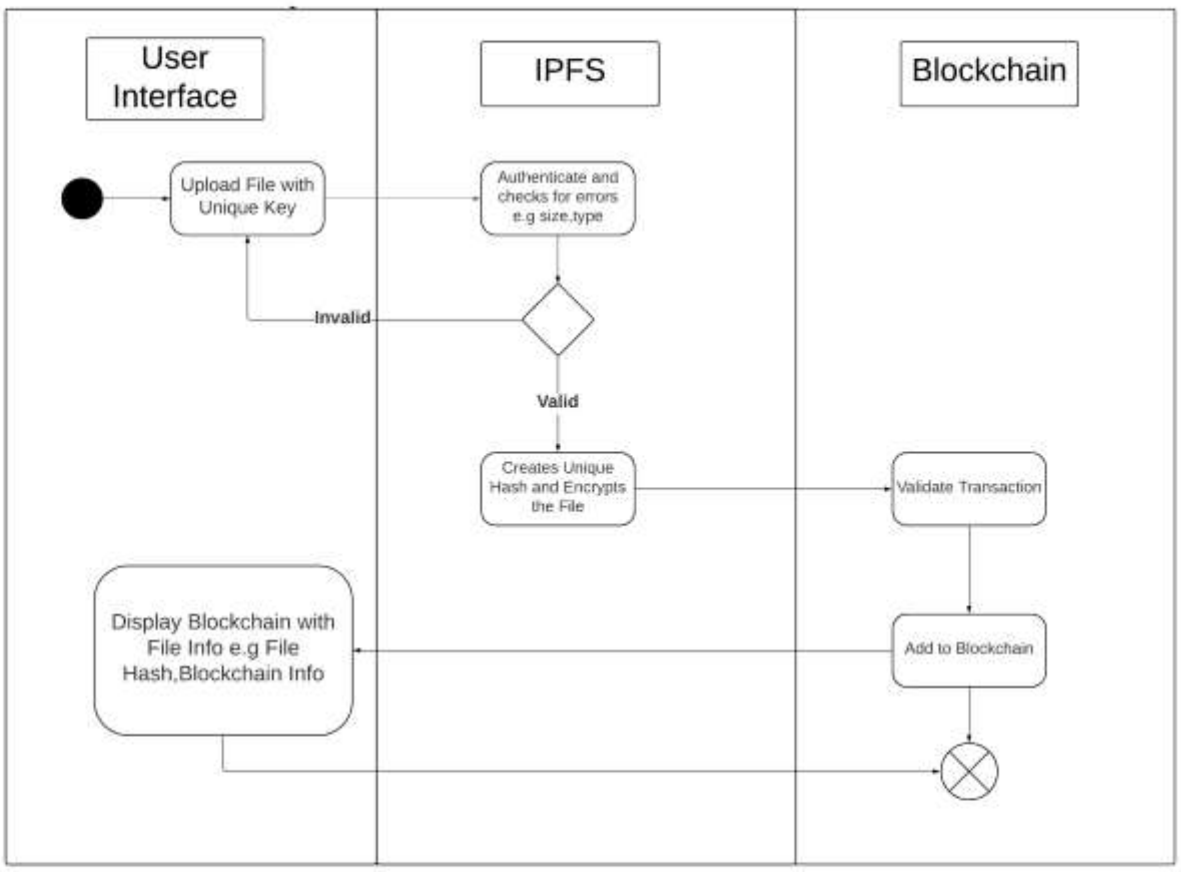

User Interface
IPFS
Blockchain
Upload File with Unique Key
Authenticate and checks for errors e.g size,type
Invalid
Valid
Creates Unique Hash and Encrypts the File
Validate Transaction
Add to Blockchain
Display Blockchain with File Info e.g File Hash,Blockchain Info

Fig 3.6 Activity for File Upload

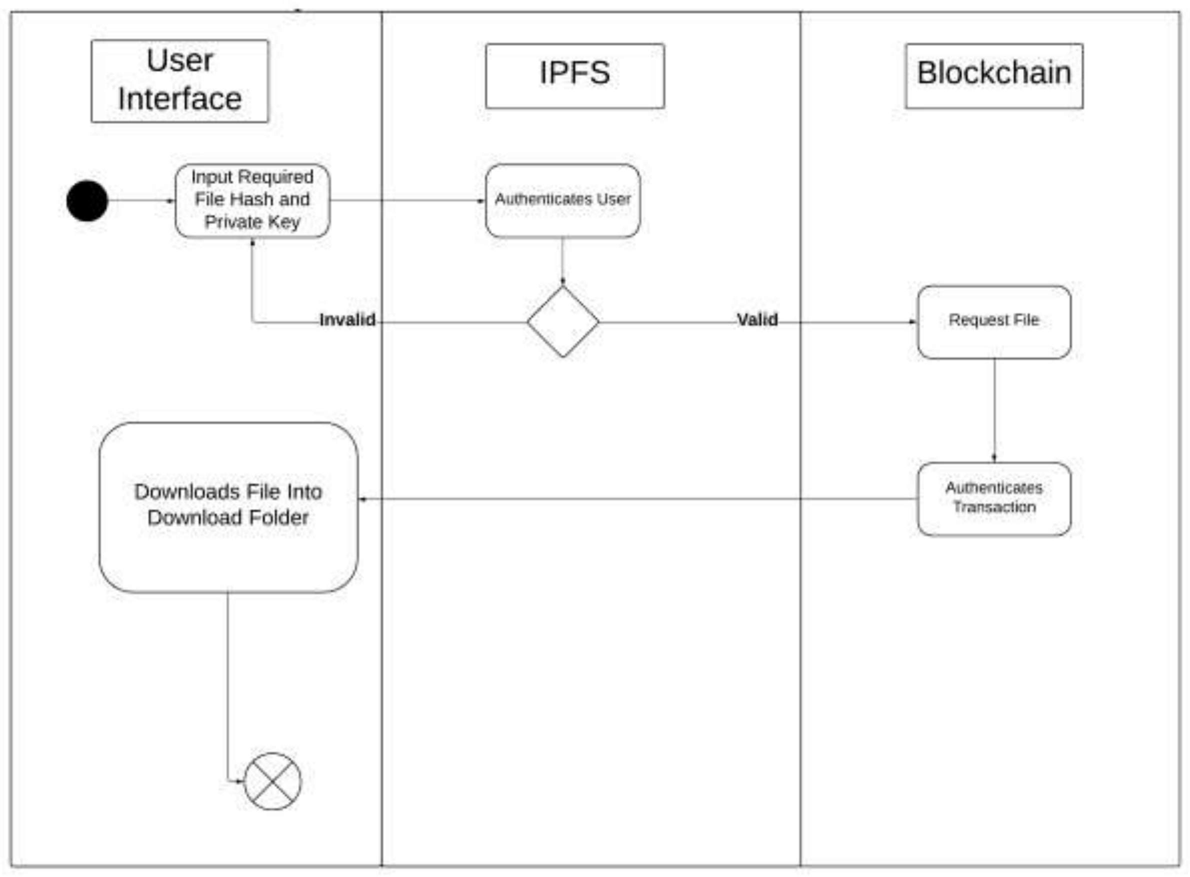

User Interface
IPFS
Blockchain
Input Required File Hash and Private Key
Authenticates User
Invalid
Valid
Request File
Authenticates Transaction
Downloads File Into Download Folder

Fig 3.7 Download Activity Diagram

3.5 Future Enhancement

The system can further be improved by adding more functions in it. Thus, system can be used for as a regular social networking application. Sharing photos, videos and communication facilities.

CHAPTER FOUR

4.1 Implementation

This chapter goes through the implementation of the IPFS and blockchain based file security application and its architecture. This chapter will give a high level overview of the application architecture.

The system developed utilises various library in the development, This chapter will also describe these various libraries their functionalities and their relevance to the project.

4.2 Software Implementation

4.2.1 Creating the Blockchain

Block Structure

In our system, a single block in a blockchain has the following structure:

The Block contains:

Block number: Simply displays the index number of the block. Block 0 refers to the genesis block.

Timestamp: This field indicates as to when the block was created and added to the blockchain.

Proof: Also called a nonce, it stands for "number only used once," which is a number added to a hashed or encrypted block in a blockchain that, when rehashed, meets the difficulty level restrictions. i.e by varying the proof we can vary the hash generated so that a new block can be created.

Previous hash: This field represents the hash of the previous block. (In this case block index 2). The hash of the entire block is generated using the SHA-256 hashing algorithm. This field creates a chain of blocks and is the main element behind blockchain architecture’s security.

Sender: The person who uploads the file enters his identity proof or name when he uploads the file.

Receiver: Displays who the intended receiver of the file shall be.

Hash of the file shared: The uploaded file is first encrypted with the file key given by the uploader using the AES encryption mechanism and subsequently using the SHA-256 hashing algorithm when it is uploaded to ipfs. The hash, then received from the IPFS after the encryption is the hash of the shared file which is added to the block.

4.2.2 Creating the Peer to Peer Network

In order to create a peer to peer network (p2p) for the blockchain to function, all the connected nodes must be in the same network. Only those users who are connected to the blockchain’s p2p network should have access to the blockchain’s data. This p2p network is created using Socket Programming.

We are working on a permissioned blockchains which require access to be a part of the blockchain. This access is granted when a user clicks on ‘Connect to the blockchain’ displayed on the home screen. Using socket programming, the list of connected nodes gets updated as soon as a new user gets connected or disconnected to the network and the updated list is broadcasted to the whole p2p network. As soon as all the connected nodes get the updated list of the nodes in the network, the consensus protocol works smoothly whenever a new block is added or the blockchain gets updated. Thus, the peer to peer network works effectively.

4.2.3 File Key

The unique key/password shared between the sender and the receiver of the shared file to increase the security of the file(s) on the blockchain network.

The upload page is to be filled out by the uploader eager to share the file. The file key entered here will be used to encrypt the file using AES encryption before uploading it to the IPFS network. The uploader will have to share the key only with the intended receiver(s) so he/she can download the file. The type of files that can be uploaded are .pdf , .png , .jpeg and .txt. As of now the size of the file that can be uploaded to the network is limited to 16 Megabytes.

The download page is to be filled by the receiver who has the valid file key shared by the sender and intends to download the shared file from the blockchain to their local computer. The file key here is used to decrypt: the AES encrypted file downloaded from the IPFS network so that the file can be interpretable. Make sure you enter the correct file key and hash for a successful download.

Integrating with IPFS

Our blockchain relies on IPFS for keeping it lightweight and scalable. If the files were stored directly on the blockchain, it would render the blockchain very heavy and inefficient. Combining IPFS and blockchain, we get to access the IPFS's power of decentralized storage and enhance the blockchain's security and accessibility. Instead of storing the file directly on the blockchain, we store the files on the IPFS network while the blockchain stores only the file' hash. Each file will have a unique hash as IPFS employs the SHA-256 hashing algorithm. Thus, the file is stored in a secure decentralized network and is easily accessible through the blockchain. The file can be retrieved using its generated hash easily. Hence IPFS eliminates the bottleneck of storing entire files on the blockchain.

4.2.4 Using Cryptographic Encryption

4.2.4.1 SHA-256 Hashing Algorithm

We use the SHA-256 algorithm to generate a unique hash of the entire block that is used by the corresponding blocks to form the chain (via the previous hashes). IPFS as well uses this algorithm to generate the hash of the shared file. The SHA-256 hashing algorithm is employed because of the following advantages:

One-way:- Once the hash is generated, we can't revert to the original data from the hash.

Deterministic:- For a particular input, the hash generated, always remains the same i.e. same input always gives the same hash.

Avalanche-effect:- Even a slight change in the input will bring about a large change in the final hash, making it untraceable

Withstand collisions:- There is a very rare chance that the hash generated for two different inputs will be the same. Think of it as a human fingerprint!

4.2.4.2 AES Encryption

AES Encryption is a form of symmetric, cryptographic encryption that depends on a shared key between the sender and receiver to access any file (here the file key) . If we had not employed the AES encryption, any connected user to the blockchain can access the file hash and thus, the shared file using the hash, directly from the IPFS. Using the AES Encryption, we encrypt the file using the file key of the uploader. Thus, if any user tries to download the file directly from the IPFS, all they get is a non-readable file. Thus, only users with a valid file key can access the readable file contents, thereby enhancing the security of the blockchain and the file contents.

4.3 Softwares Used

4.3.1 Development Environment

Visual Studio Code

A lightweight cross-platform code editor designed by Microsoft for multiple operating systems. It allows powerful debugging with a variety of tools, highlighted syntax, intelligent completion of code, build in Git control and code refactoring

4.3.2 Web Framework

Flask

Flask is a web framework written in Python. It is classified as a micro framework because it does not require particular tools or libraries.it has no database abstraction layer, form validation, or any other components were pre-existing third-party libraries provide common functions.

4.3.3 Front-End

HTML5

HTML5 is a markup language used for structuring and presenting content on the World Wide Web.

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

BOOTSTRAP 3

This is a front-end open source toolkit, featuring Sass variables and mixins, responsive grid system, extensive prebuilt components, and powerful JavaScript plugins.

4.3.4 Back-End

Blockchain

Blockchain is a system of recording information in a way that makes it difficult or impossible to change, hack, or cheat the system. A blockchain is essentially a digital ledger of transactions that is duplicated and distributed across the entire network of computer systems on the blockchain.

4.3.5 Simulation Setup

The specifications for implementation setup are: Intel® core™ i3-6100U CPU@2.30 GHz, 6 GB RAM, 64 bit operating system and X64-based processor. The programming language is python to write the blockchain. Front-end web GUI is developed using Bootstrap 4.0 and Javascript for user interactive forms. The primary tools used to develop this system are given below:

Interplanetary File System (IPFS)

The Interplanetary File System (IPFS) is a protocol and peer-to-peer network for storing and sharing data in a distributed file system. IPFS uses content-addressing to uniquely identify each file in a global namespace connecting all computing devices.

4.4 Graphical User Interface

4.4.1 Home Page

The Home page of the system displays the information about the system functions in the about menu.it also displays services menu which when the user accesses prompts them to connect to the blockchain.

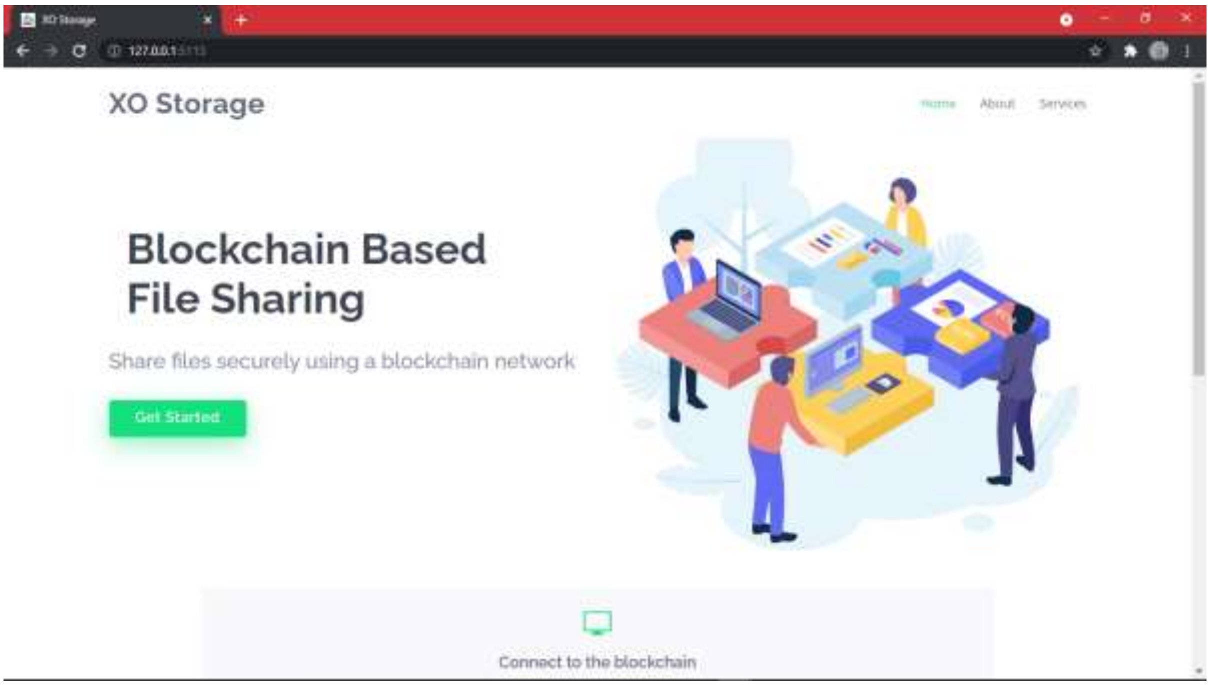
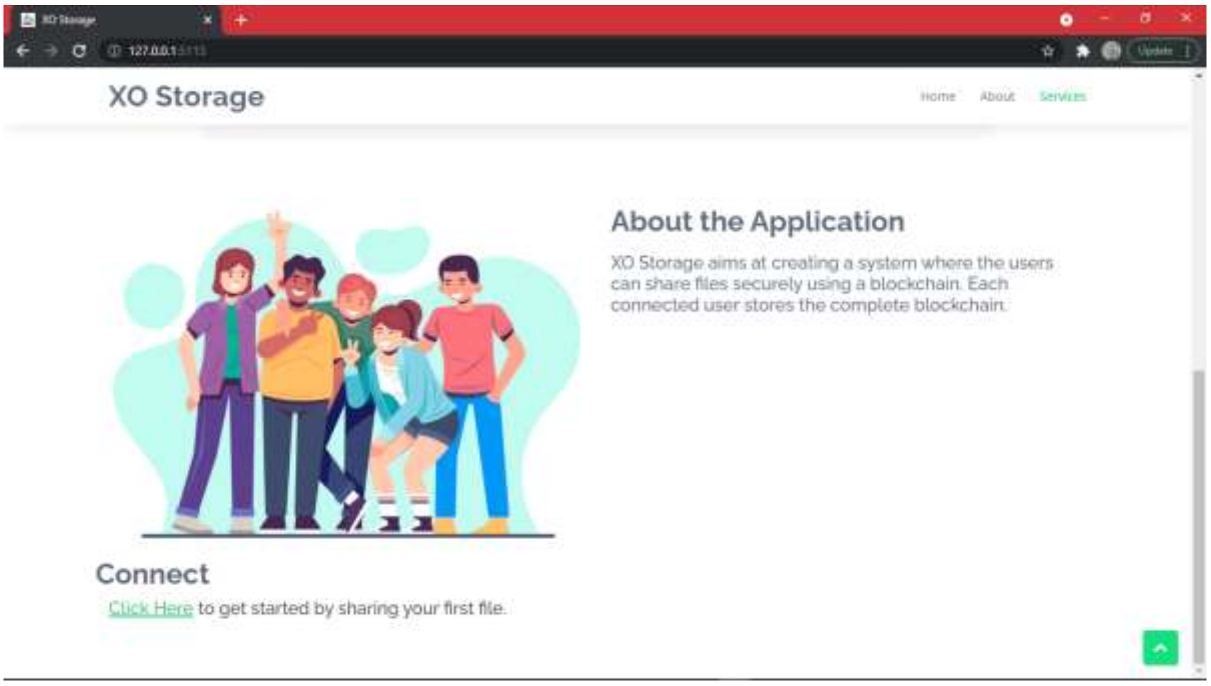

XO Storage
127.0.0.1
XO Storage
Home
About
Services
Blockchain Based File Sharing
Share files securely using a blockchain network
Get Started
Connect to the blockchain

Fig 4.1 Home Page of the system

The page also includes a back to top button to return user to initial point the about menu and services are all displayed on one page to better improve functionality of user instead of moving from page to page.


XO Storage

About the Application

XO Storage aims at creating a system where the users can share files securely using a blockchain. Each connected user stores the complete blockchain.

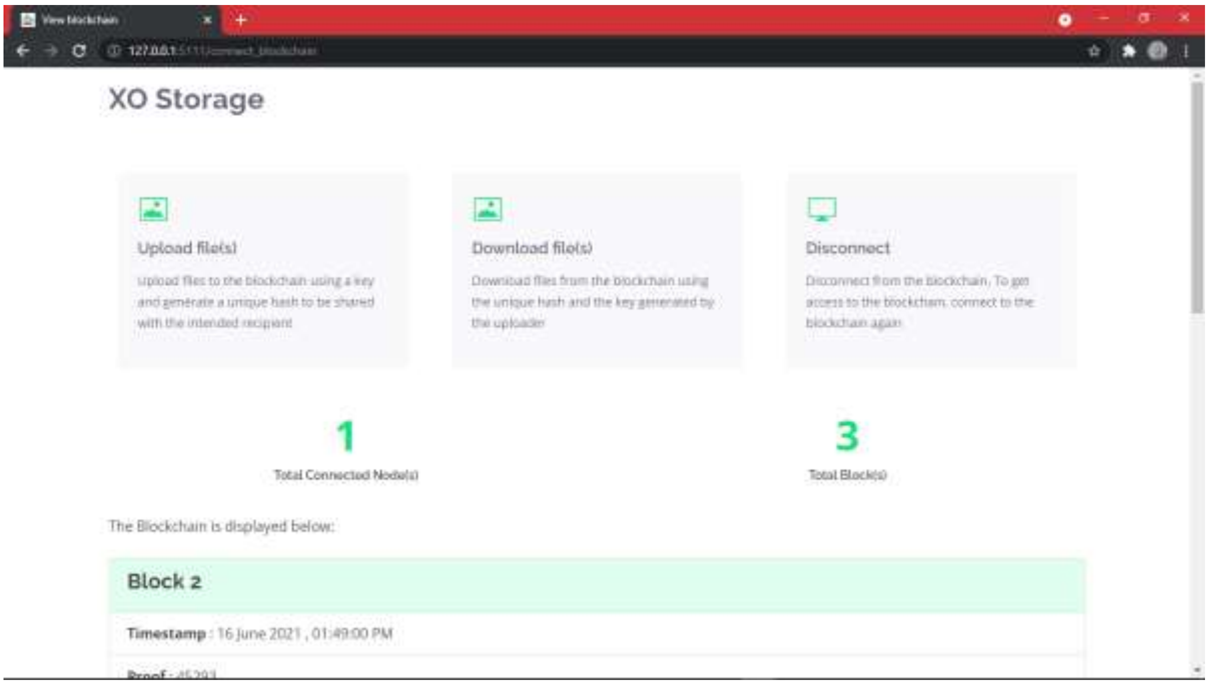
Connect

Click Here to get started by sharing your first file.

Fig 4.2 Home Page Showing system functions

4.4.2 The Blockchain Page

When the user clicks the connect to blockchain button it takes them to the blockchain page that showcases system features that include upload file, download file and disconnect from the blockchain.


XO Storage

Upload file(s)

upload files to the blockchain using a key and generate a unique hash to be shared with the intended recipient

Download file(s)

Download files from the blockchain using the unique hash and the key generated by the uploader

Disconnect

Disconnect from the blockchain. To get access to the blockchain, connect to the blockchain again

1

Total Connected Node(s)

3

Total Block(s)

The Blockchain is displayed below:

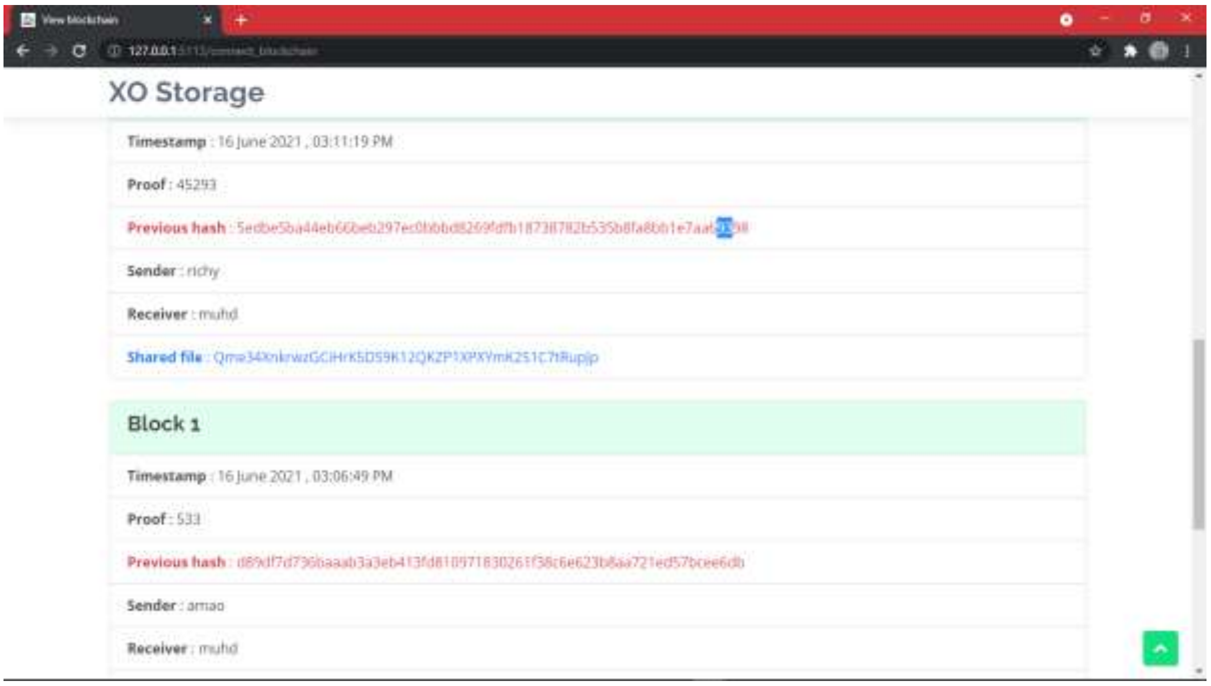
Block 2

Timestamp : 16 June 2021 , 01:49:00 PM

Proof : 45293

Fig 4.3 Blockchain Page

The page displays the system blocks already present in the system along with how many nodes are currently connected to the blockchain. A single block displays the sender name, receiver name, file hash, block name and the time stamp of when the file was uploaded.


XO Storage

Timestamp : 16 June 2021 , 03:11:19 PM

Proof : 45293

Previous hash : 5edbe5ba44eb66beb297ec0bbbd8269fdfb18738782b535b8fa8bb1e7aa0398

Sender : richy

Receiver : muhd

Shared file : Qme34XnkrwzGCiHrK5DS9K12QKZP1XPXYmK251C7tRupjp

Block 1

Timestamp : 16 June 2021 , 03:06:49 PM

Proof : 533

Previous hash : d89df7d736baaab3a3eb413fd810971830261f38c6e623b8aa721ed57bcee6db

Sender : amao

Receiver : muhd

Fig 4.4 Blockchain Page Showing blocks

4.4.3 Upload Page

When the user accesses the upload page by clicking the upload file option, they will be forwarded to the upload page, this is the page where the user can upload file with private key and unique file hash will be generated for it.

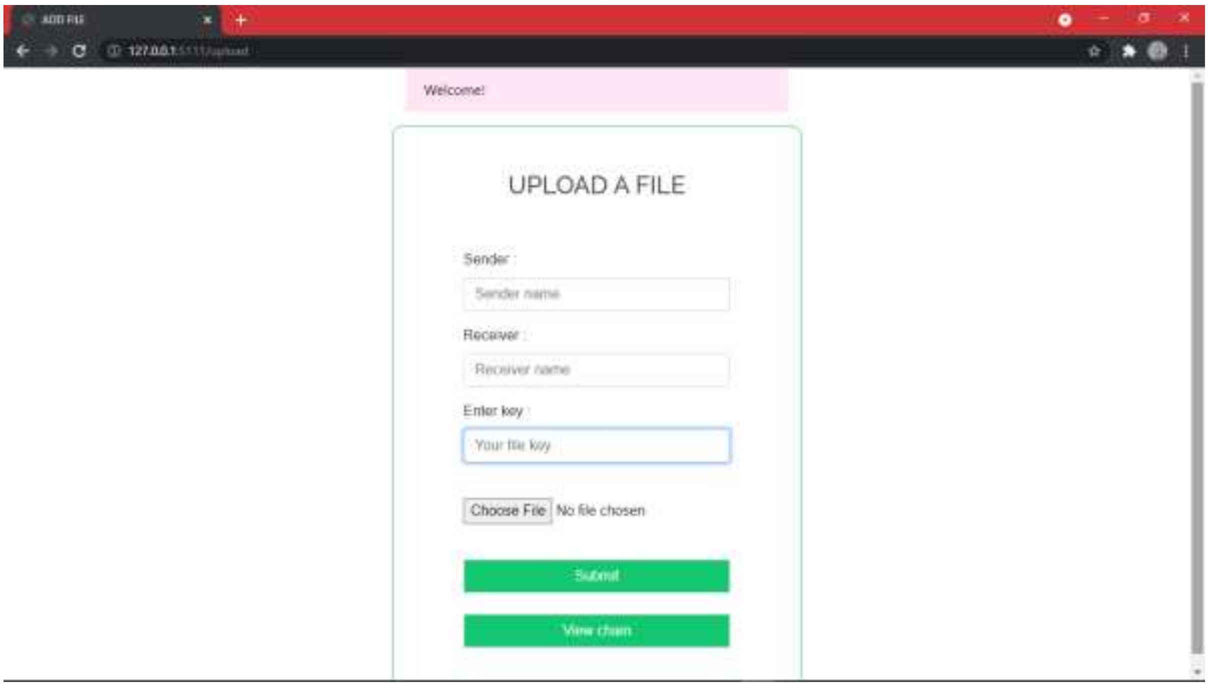
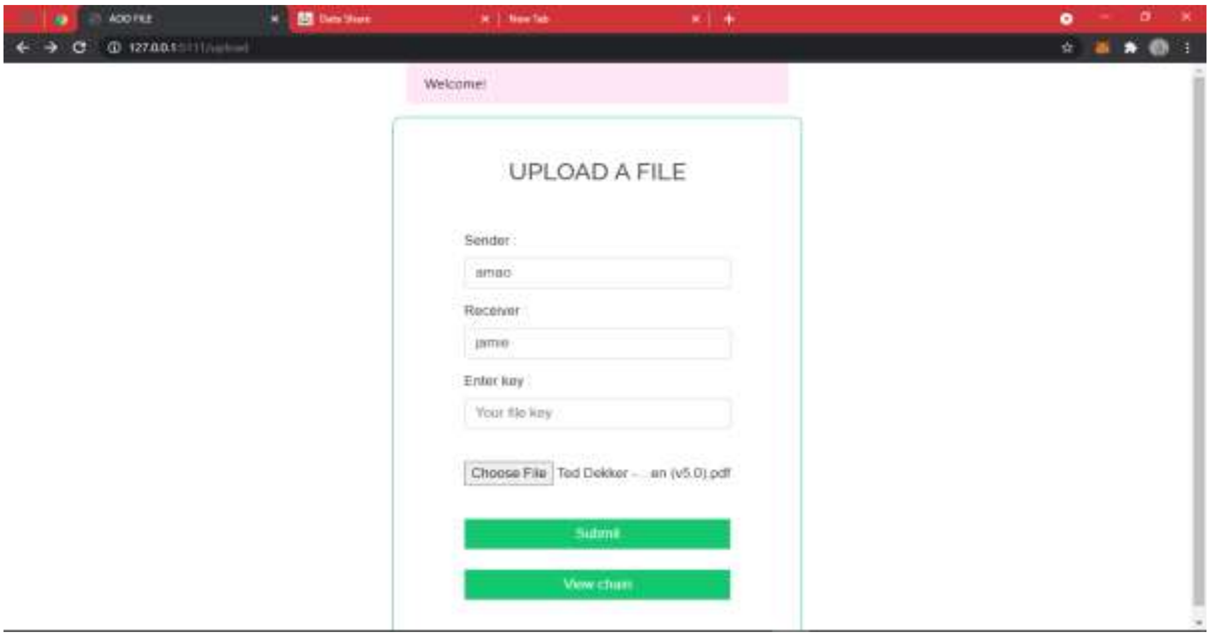


Fig 4.5 Upload Page

The system will update the user with prompts as to whether the file has been updated successfully or it as encountered some errors like invalid file type or alert the user if they are disconnect from internet and cannot access the blockchain to upload files at current time.

The User can click on view chain option to return to previous page and view the file they just uploaded included in the latest block


Welcome!

UPLOAD A FILE

Sender : amao

Receiver : jamie

Enter key : Your file key

Choose File Ted Dekker - ... en (v5.0).pdf

Fig 4.6 Upload Page with user input

4.4.4 Download Page

When the user accesses the download page by clicking the download file option, they will be forwarded to the download page,the download is the page where the user can download file with private key and unique file hash that has been generated for it.

The system will prompt user when there is successful download of file shown in fig 4.7 .The system has also been designed to prompt the user if the download has encountered some error while downloading the file.Errors could include bad Internet connection, Incorrect private key or file hash

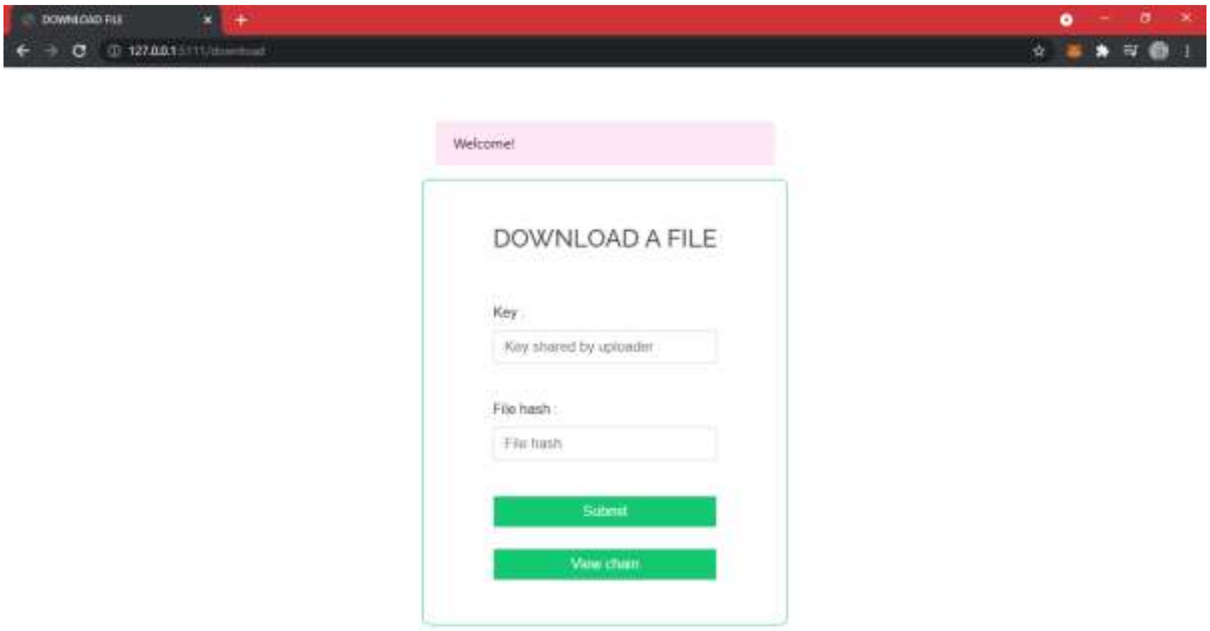


Fig 4.7 Download Page

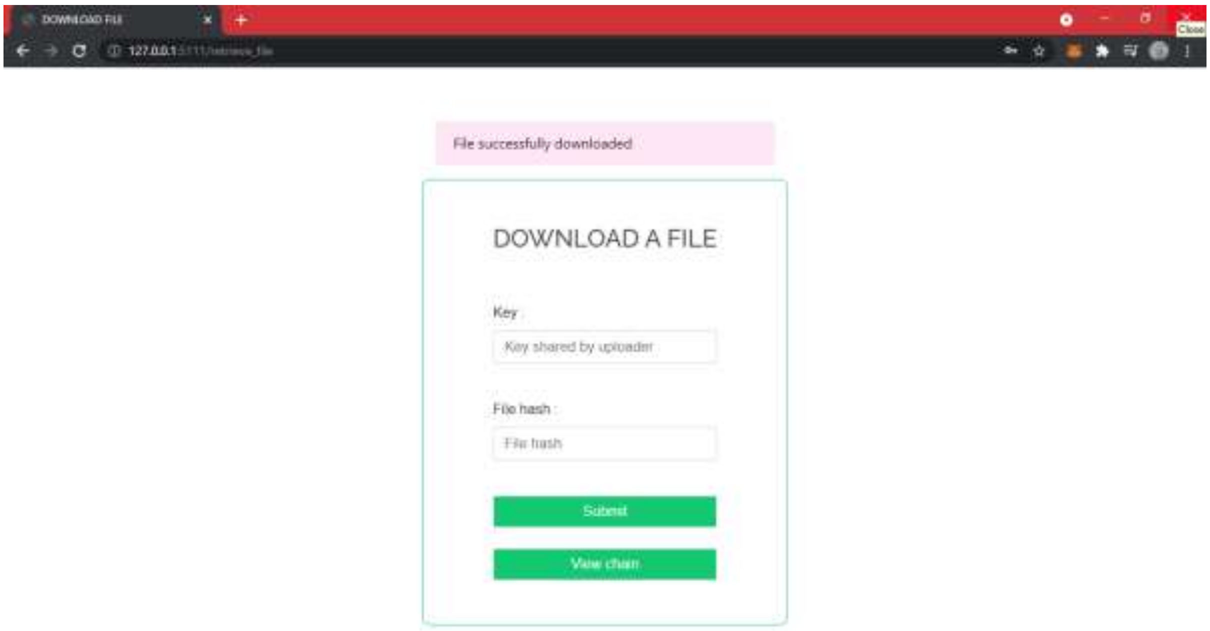
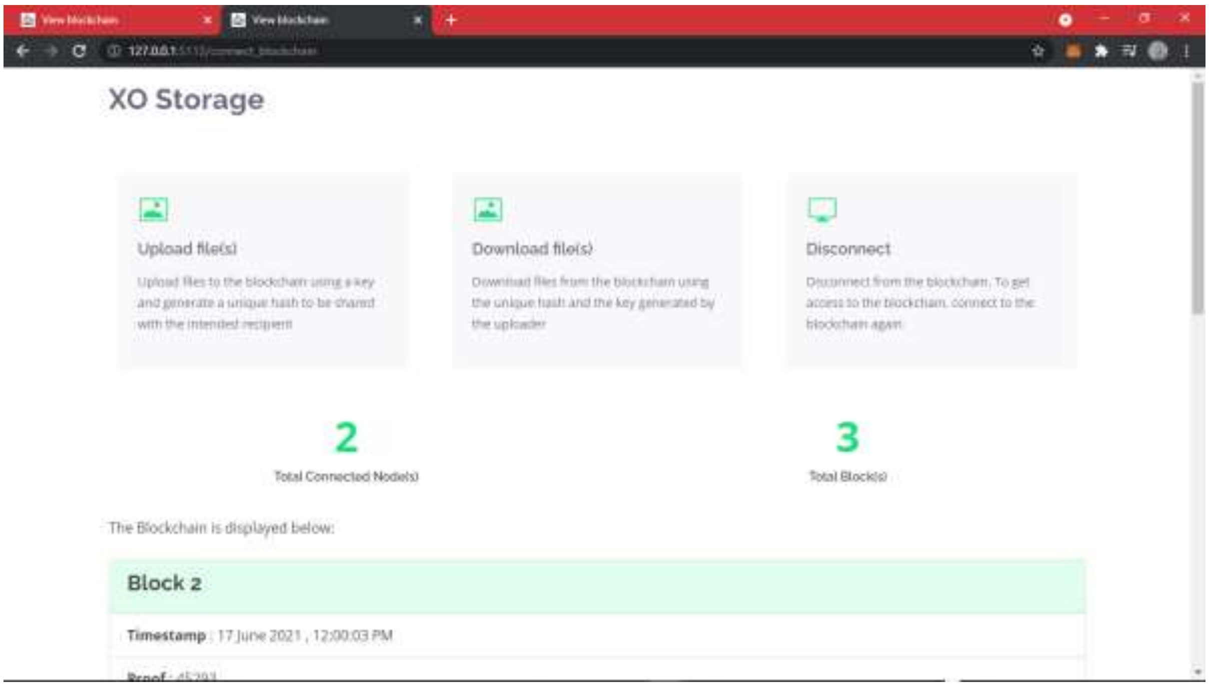


Fig 4.8 Download Page with user successfully uploaded file

4.4.5 New peers in System

For the system to satisfy peer to peer functionality, the inclusion of additionally node is simulated. When a new user on another node joins the blockchain they would see all connected nodes currently in system along with all blocks currently in the system as shown in fig 4.9 .The new user can add their own files to the blockchain and it’ll reflect in other users blocks.



XO Storage

Upload file(s)

Upload files to the blockchain using a key and generate a unique hash to be shared with the intended recipient

Download file(s)

Download files from the blockchain using the unique hash and the key generated by the uploader

Disconnect

Disconnect from the blockchain. To get access to the blockchain, connect to the blockchain again

2

Total Connected Node(s)

3

Total Block(s)

The Blockchain is displayed below:

Block 2

Timestamp : 17 June 2021 , 12:00:03 PM

Proof : 45293

Fig 4.9 Blockchain showing two nodes connected.

4.5 System Testing

After the implementation of the system the system was tested to determine if it satisfies the design goals. The system is tested with the three widely used methods of software testing-Unit Testing, Regression Testing and System Testing

Unit testing: After the system was developed according to design specifications each subsystem was taken individually and accessed to see if they performed in accepting input and generating desired output.it was also evaluated from an end user's perspective.

Regression Testing: The regression testing was done to confirm and identify if there is any error in the system due to modification in another subsystem.It makes sure any changes made in modification doesn't affect and introduce new defects in the entire system.

System Testing: The system was then tested as a whole with positive and negative values to observe system performance for errors and make actions that will allow the system obtain our defined goals

CHAPTER FOUR

4.1 Implementation

This chapter goes through the implementation of the IPFS and blockchain based file security application and its architecture. This chapter will give a high level overview of the application architecture.

The system developed utilises various library in the development, This chapter will also describe these various libraries their functionalities and their relevance to the project.

4.2 Software Implementation

4.2.1 Creating the Blockchain

Block Structure

In our system, a single block in a blockchain has the following structure:

The Block contains:

Block number: Simply displays the index number of the block. Block 0 refers to the genesis block.

Timestamp: This field indicates as to when the block was created and added to the blockchain.

Proof: Also called a nonce, it stands for "number only used once," which is a number added to a hashed or encrypted block in a blockchain that, when rehashed, meets the difficulty level restrictions. i.e by varying the proof we can vary the hash generated so that a new block can be created.

Previous hash: This field represents the hash of the previous block. (In this case block index 2). The hash of the entire block is generated using the SHA-256 hashing algorithm. This field creates a chain of blocks and is the main element behind blockchain architecture's security.

Sender: The person who uploads the file enters his identity proof or name when he uploads the file.

Receiver: Displays who the intended receiver of the file shall be.

Hash of the file shared: The uploaded file is first encrypted with the file key given by the uploader using the AES encryption mechanism and subsequently using the SHA-256 hashing algorithm when it is uploaded to ipfs. The hash, then received from the IPFS after the encryption is the hash of the shared file which is added to the block.

4.2.2 Creating the Peer to Peer Network

In order to create a peer to peer network (p2p) for the blockchain to function, all the connected nodes must be in the same network. Only those users who are connected to the blockchain's p2p network should have access to the blockchain's data. This p2p network is created using Socket Programming.

We are working on a permissioned blockchains which require access to be a part of the blockchain. This access is granted when a user clicks on 'Connect to the blockchain' displayed on the home screen. Using socket programming, the list of connected nodes gets updated as soon as a new user gets connected or disconnected to the network and the updated list is broadcasted to the whole p2p network. As soon as all the connected nodes get the updated list of the nodes in the network, the consensus protocol works smoothly whenever a new block is added or the blockchain gets updated. Thus, the peer to peer network works effectively.

4.2.3 File Key

The unique key/password shared between the sender and the receiver of the shared file to increase the security of the file(s) on the blockchain network.

The upload page is to be filled out by the uploader eager to share the file. The file key entered here will be used to encrypt the file using AES encryption before uploading it to the IPFS network. The uploader will have to share the key only with the intended receiver(s) so he/she can download the file. The type of files that can be uploaded are .pdf , .png , .jpeg and .txt. As of now the size of the file that can be uploaded to the network is limited to 16 Megabytes.

The download page is to be filled by the receiver who has the valid file key shared by the sender and intends to download the shared file from the blockchain to their local computer. The file key here is used to decrypt: the AES encrypted file downloaded from the IPFS network so that the file can be interpretable. Make sure you enter the correct file key and hash for a successful download.

Integrating with IPFS

Our blockchain relies on IPFS for keeping it lightweight and scalable. If the files were stored directly on the blockchain, it would render the blockchain very heavy and inefficient. Combining IPFS and blockchain, we get to access the IPFS's power of decentralized storage and enhance the blockchain's security and accessibility. Instead of storing the file directly on the blockchain, we store the files on the IPFS network while the blockchain stores only the file' hash. Each file will have a unique hash as IPFS employs the SHA-256 hashing algorithm. Thus, the file is stored in a secure decentralized network and is easily accessible through the blockchain. The file can be retrieved using its generated hash easily. Hence IPFS eliminates the bottleneck of storing entire files on the blockchain.

4.2.4 Using Cryptographic Encryption

4.2.4.1 SHA-256 Hashing Algorithm

We use the SHA-256 algorithm to generate a unique hash of the entire block that is used by the corresponding blocks to form the chain (via the previous hashes). IPFS as well uses this algorithm to generate the hash of the shared file. The SHA-256 hashing algorithm is employed because of the following advantages:

One-way:- Once the hash is generated, we can't revert to the original data from the hash.

Deterministic:- For a particular input, the hash generated, always remains the same i.e. same input always gives the same hash.

Avalanche-effect:- Even a slight change in the input will bring about a large change in the final hash, making it untraceable

Withstand collisions:- There is a very rare chance that the hash generated for two different inputs will be the same. Think of it as a human fingerprint!

4.2.4.2 AES Encryption

AES Encryption is a form of symmetric, cryptographic encryption that depends on a shared key between the sender and receiver to access any file (here the file key) . If we had not employed the AES encryption, any connected user to the blockchain can access the file hash and thus, the shared file using the hash, directly from the IPFS. Using the AES Encryption, we encrypt the file using the file key of the uploader. Thus, if any user tries to download the file directly from the IPFS, all they get is a non-readable file. Thus, only users with a valid file key can access the readable file contents, thereby enhancing the security of the blockchain and the file contents.

4.3 Softwares Used

4.3.1 Development Environment

Visual Studio Code

A lightweight cross-platform code editor designed by Microsoft for multiple operating systems. It allows powerful debugging with a variety of tools, highlighted syntax, intelligent completion of code, build in Git control and code refactoring

4.3.2 Web Framework

Flask

Flask is a web framework written in Python. It is classified as a micro framework because it does not require particular tools or libraries.it has no database abstraction layer, form validation, or any other components were pre-existing third-party libraries provide common functions.

4.3.3 Front-End

HTML5

HTML5 is a markup language used for structuring and presenting content on the World Wide Web.

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

BOOTSTRAP 3

This is a front-end open source toolkit, featuring Sass variables and mixins, responsive grid system, extensive prebuilt components, and powerful JavaScript plugins.

4.3.4 Back-End

Blockchain

Blockchain is a system of recording information in a way that makes it difficult or impossible to change, hack, or cheat the system. A blockchain is essentially a digital ledger of transactions that is duplicated and distributed across the entire network of computer systems on the blockchain.

4.3.5 Simulation Setup

The specifications for implementation setup are: Intel® core™ i3-6100U CPU@2.30 GHz, 6 GB RAM, 64 bit operating system and X64-based processor. The programming language is python to write the blockchain. Front-end web GUI is developed using Bootstrap 4.0 and Javascript for user interactive forms. The primary tools used to develop this system are given below:

Interplanetary File System (IPFS)

The Interplanetary File System (IPFS) is a protocol and peer-to-peer network for storing and sharing data in a distributed file system. IPFS uses content-addressing to uniquely identify each file in a global namespace connecting all computing devices.

4.4 Graphical User Interface

4.4.1 Home Page

The Home page of the system displays the information about the system functions in the about menu.it also displays services menu which when the user accesses prompts them to connect to the blockchain.

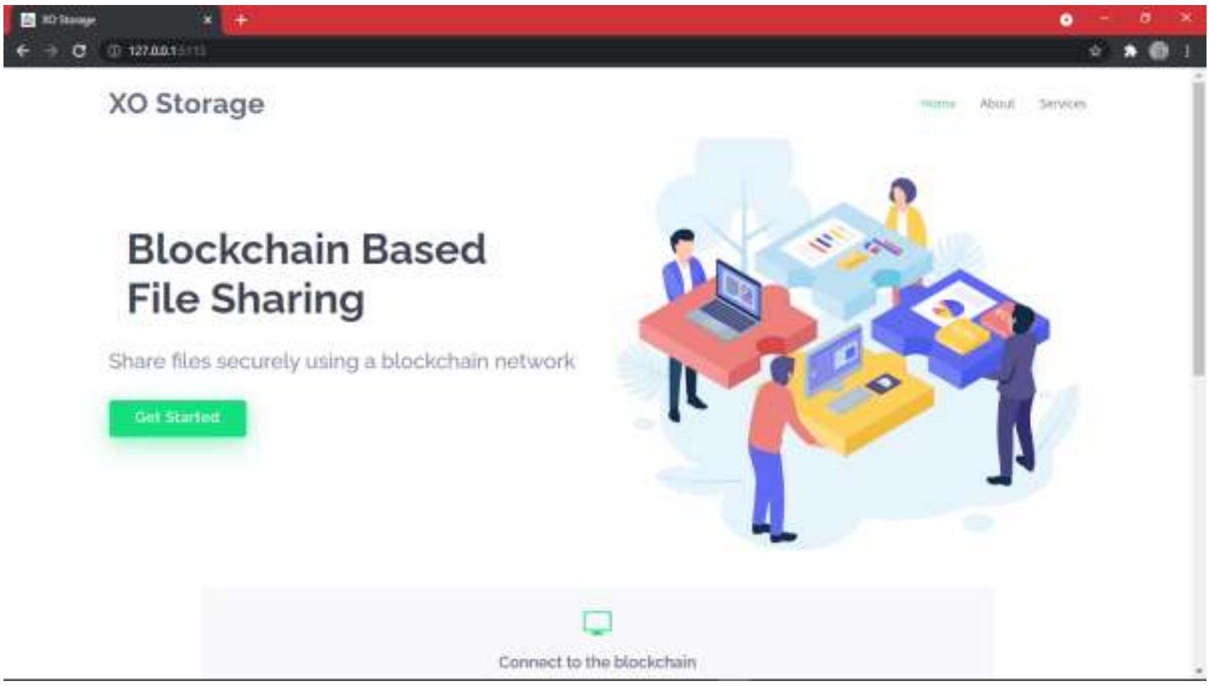
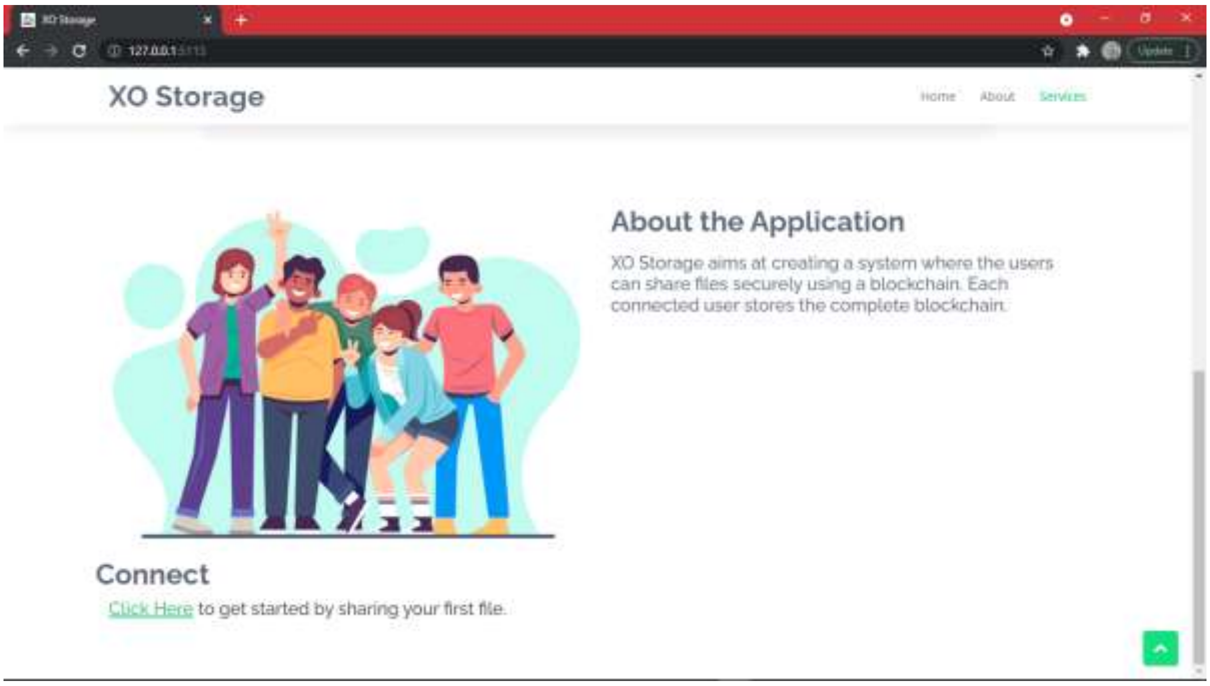

XO Storage
127.0.0.1
XO Storage
Home
About
Services
Blockchain Based File Sharing
Share files securely using a blockchain network
Get Started
Connect to the blockchain

Fig 4.1 Home Page of the system

The page also includes a back to top button to return user to initial point the about menu and services are all displayed on one page to better improve functionality of user instead of moving from page to page.



About the Application

XO Storage aims at creating a system where the users can share files securely using a blockchain. Each connected user stores the complete blockchain.

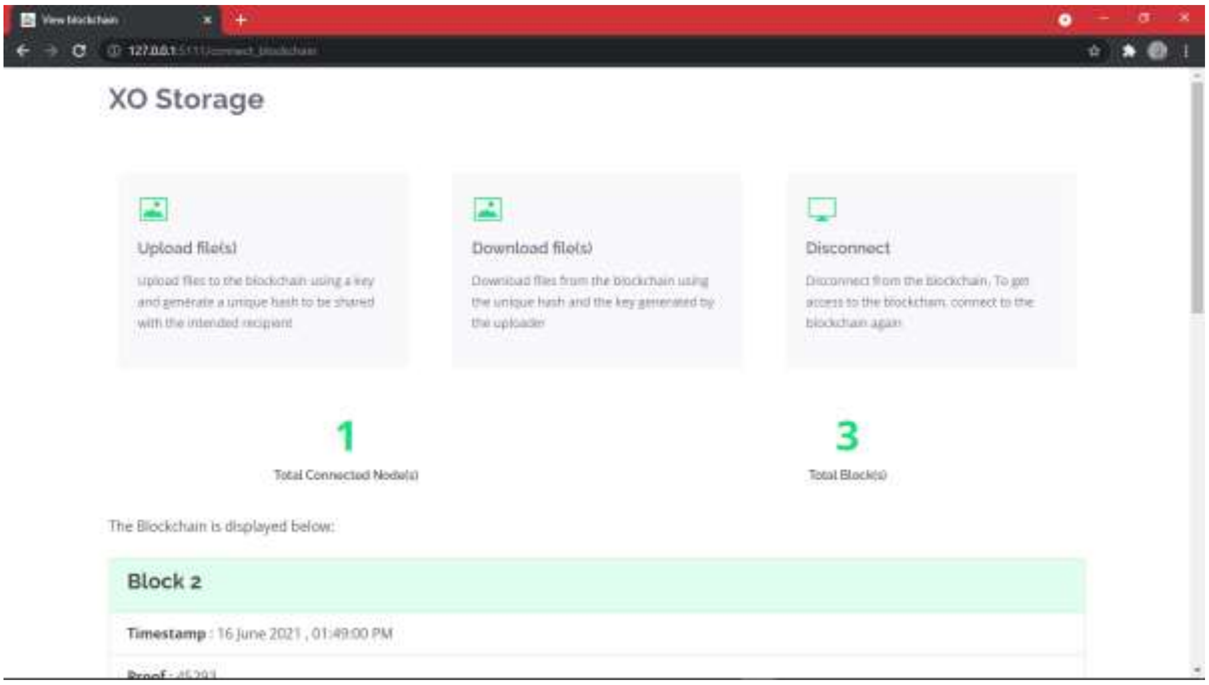
Connect

Click Here to get started by sharing your first file.

Fig 4.2 Home Page Showing system functions

4.4.2 The Blockchain Page

When the user clicks the connect to blockchain button it takes them to the blockchain page that showcases system features that include upload file, download file and disconnect from the blockchain.


XO Storage

Upload file(s)

upload files to the blockchain using a key and generate a unique hash to be shared with the intended recipient

Download file(s)

Download files from the blockchain using the unique hash and the key generated by the uploader

Disconnect

Disconnect from the blockchain. To get access to the blockchain, connect to the blockchain again

1

Total Connected Node(s)

3

Total Block(s)

The Blockchain is displayed below:

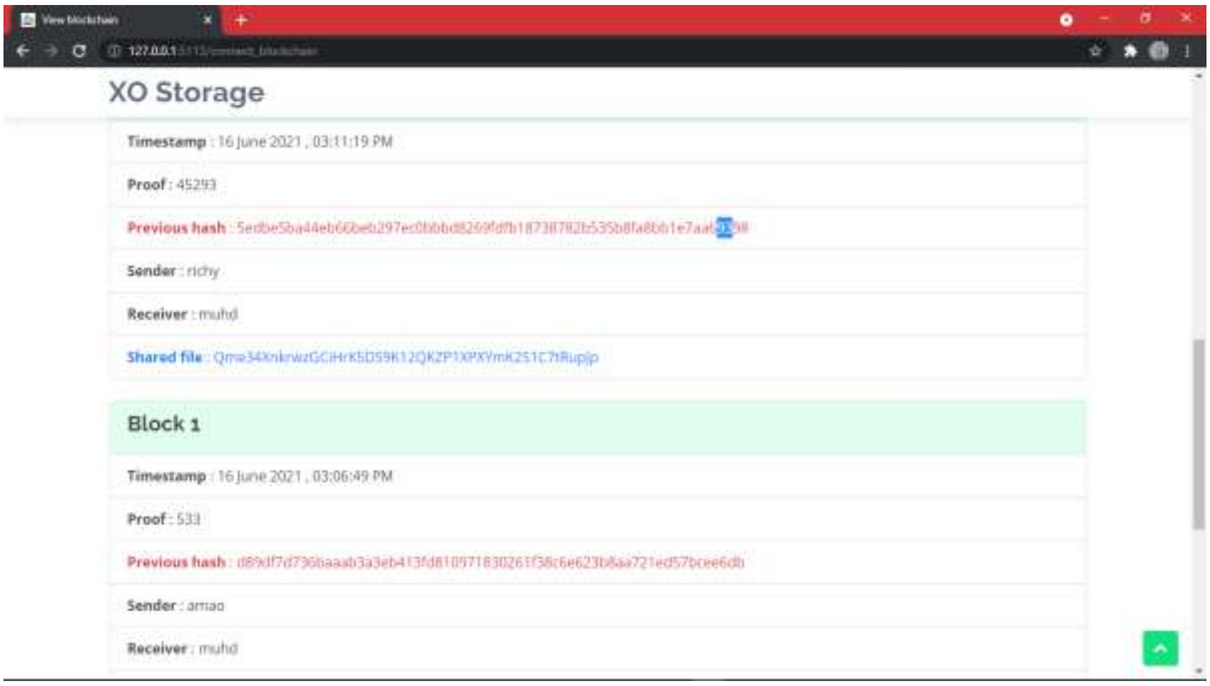
Block 2

Timestamp : 16 June 2021 , 01:49:00 PM

Proof : 45293

Fig 4.3 Blockchain Page

The page displays the system blocks already present in the system along with how many nodes are currently connected to the blockchain. A single block displays the sender name, receiver name, file hash, block name and the time stamp of when the file was uploaded.


XO Storage

Timestamp : 16 June 2021 , 03:11:19 PM

Proof : 45293

Previous hash : 5edbe5ba44eb66beb297ec0bbbd8269fdfb18738782b535b8fa8bb1e7aa2398

Sender : richy

Receiver : muhd

Shared file : Qme34XnkrwzGCiHrK5DS9K12QKZP1XPXYmK251C7tRupjp

Block 1

Timestamp : 16 June 2021 , 03:06:49 PM

Proof : 533

Previous hash : d89df7d736baaab3a3eb413fd810971830261f38c6e623b8aa721ed57bcee6db

Sender : amao

Receiver : muhd

Fig 4.4 Blockchain Page Showing blocks

4.4.3 Upload Page

When the user accesses the upload page by clicking the upload file option, they will be forwarded to the upload page, this is the page where the user can upload file with private key and unique file hash will be generated for it.

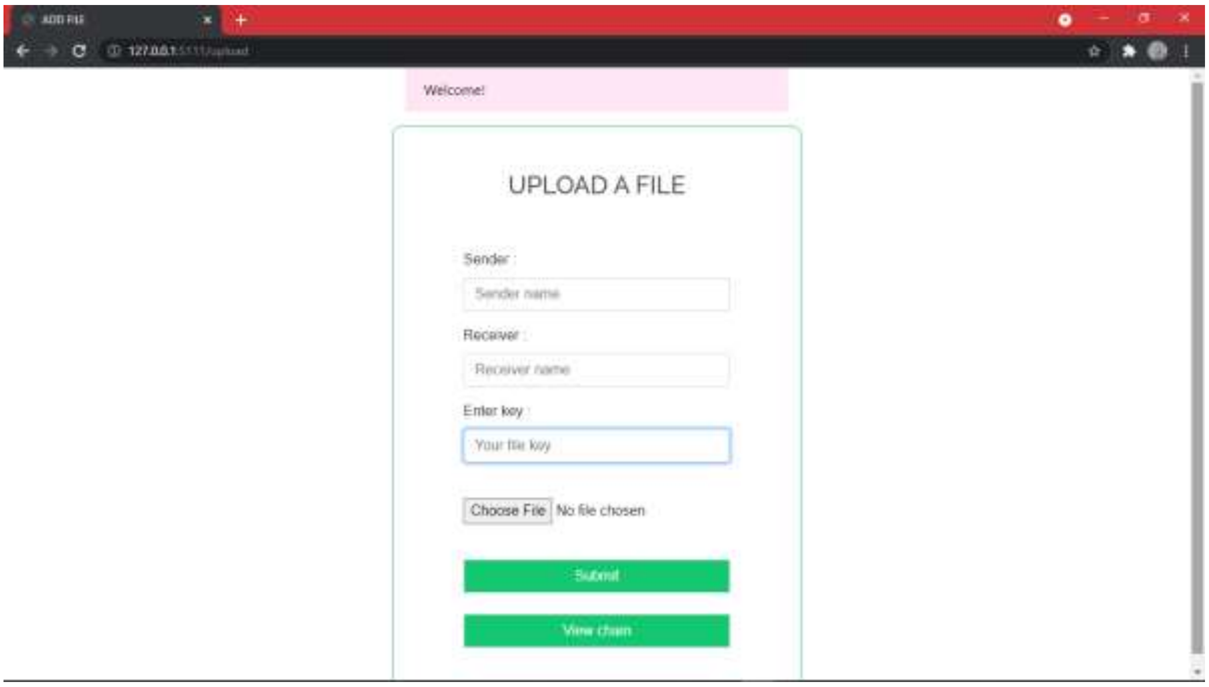
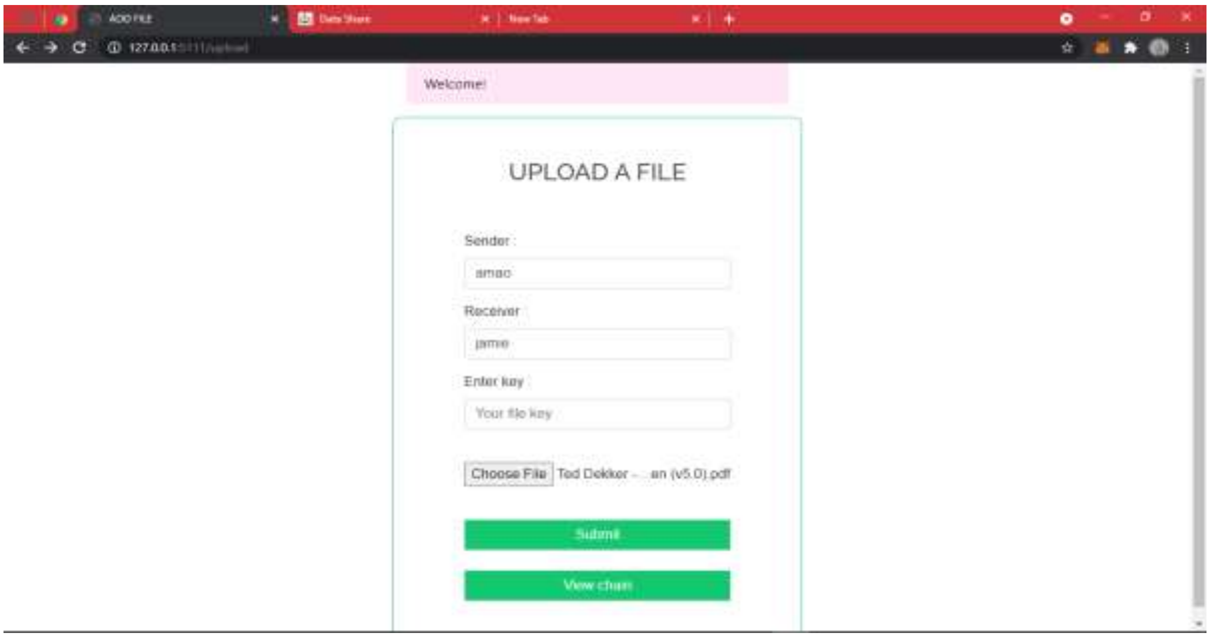


Fig 4.5 Upload Page

The system will update the user with prompts as to whether the file has been updated successfully or it as encountered some errors like invalid file type or alert the user if they are disconnect from internet and cannot access the blockchain to upload files at current time.

The User can click on view chain option to return to previous page and view the file they just uploaded included in the latest block


Welcome!

UPLOAD A FILE

Sender : amao

Receiver : jamie

Enter key : Your file key

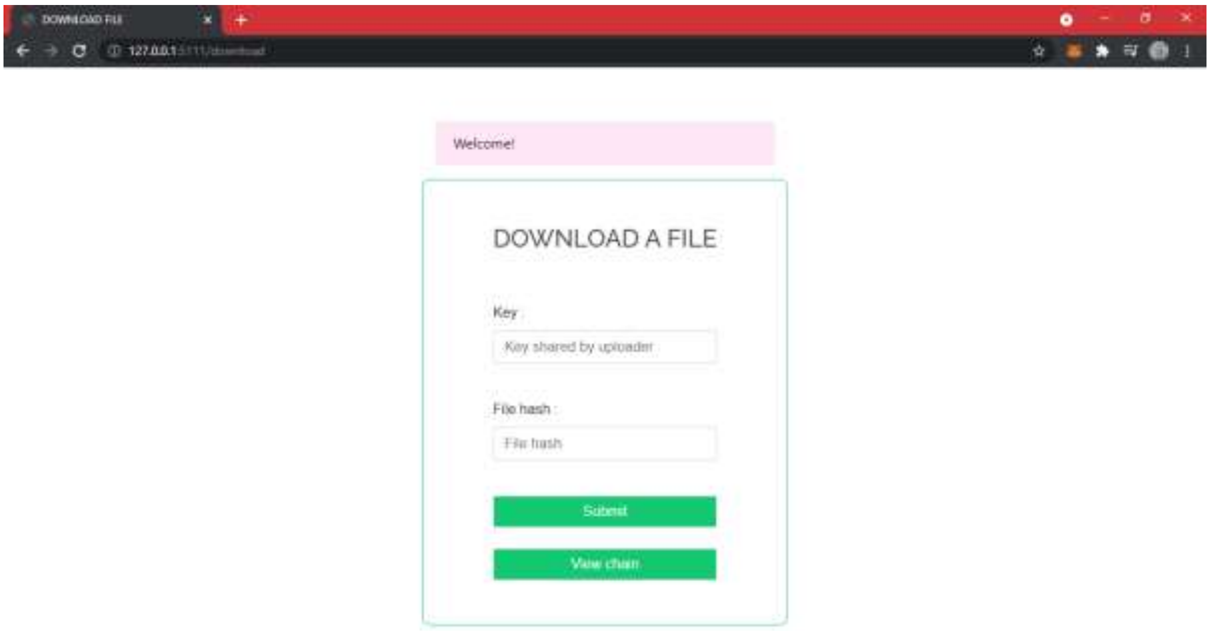
Choose File Ted Dekker - ...an (v5.0).pdf

Fig 4.6 Upload Page with user input

4.4.4 Download Page

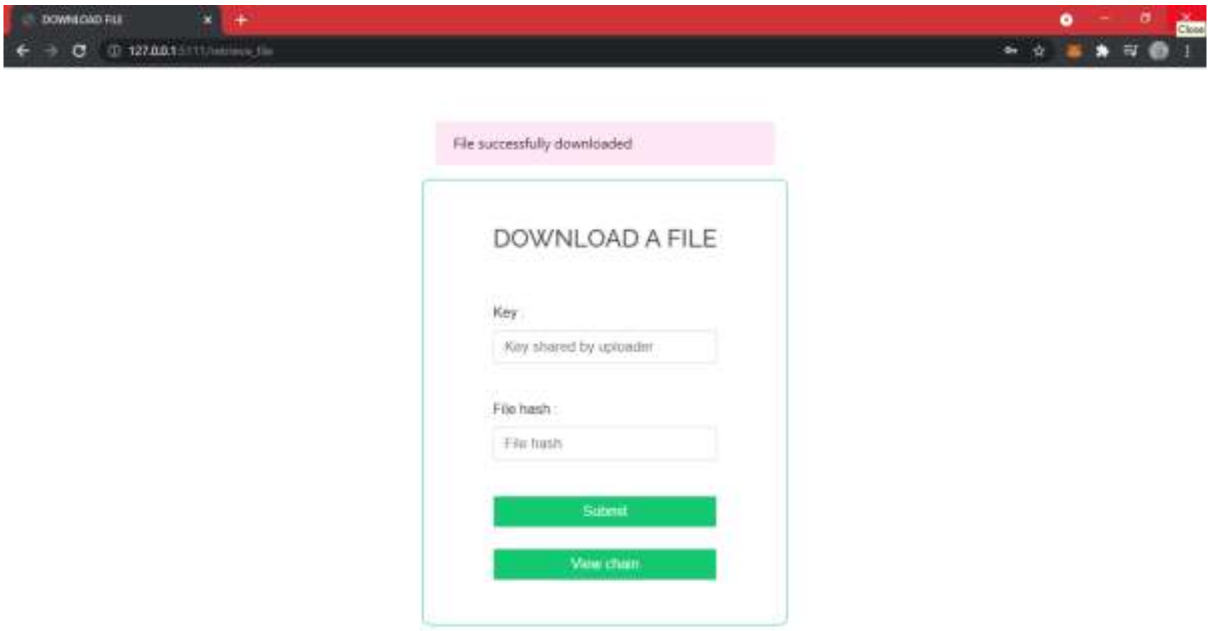
When the user accesses the download page by clicking the download file option, they will be forwarded to the download page,the download is the page where the user can download file with private key and unique file hash that has been generated for it.

The system will prompt user when there is successful download of file shown in fig 4.7 .The system has also been designed to prompt the user if the download has encountered some error while downloading the file.Errors could include bad Internet connection, Incorrect private key or file hash



Welcome!

Fig 4.7 Download Page

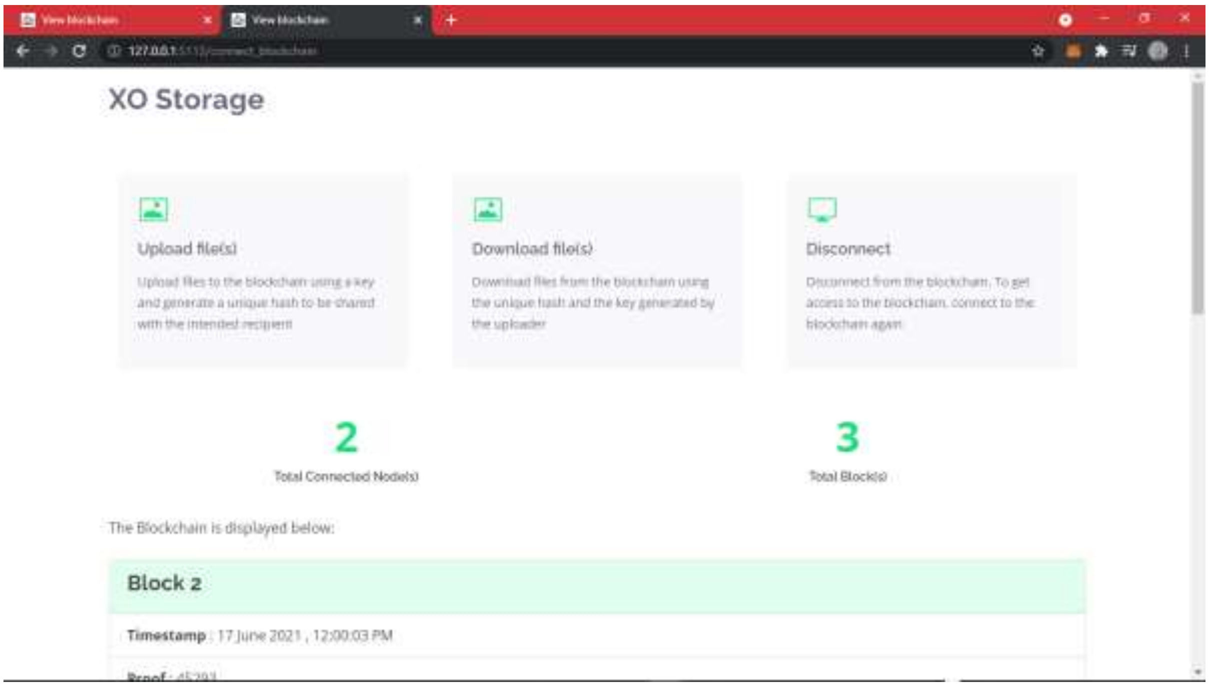


File successfully downloaded

Fig 4.8 Download Page with user successfully uploaded file

4.4.5 New peers in System

For the system to satisfy peer to peer functionality, the inclusion of additionally node is simulated. When a new user on another node joins the blockchain they would see all connected nodes currently in system along with all blocks currently in the system as shown in fig 4.9 .The new user can add their own files to the blockchain and it'll reflect in other users blocks.



XO Storage

Upload file(s)

Upload files to the blockchain using a key and generate a unique hash to be shared with the intended recipient

Download file(s)

Download files from the blockchain using the unique hash and the key generated by the uploader

Disconnect

Disconnect from the blockchain. To get access to the blockchain, connect to the blockchain again

2
Total Connected Node(s)

3
Total Block(s)

The Blockchain is displayed below:

Block 2

Timestamp : 17 June 2021 , 12:00:03 PM

Proof : 45293

Fig 4.9 Blockchain showing two nodes connected.

4.5 System Testing

After the implementation of the system the system was tested to determine if it satisfies the design goals. The system is tested with the three widely used methods of software testing- Unit Testing, Regression Testing and System Testing

Unit testing: After the system was developed according to design specifications each subsystem was taken individually and accessed to see if they performed in accepting input and generating desired output.it was also evaluated from an end user's perspective.

Regression Testing: The regression testing was done to confirm and identify if there is any error in the system due to modification in another subsystem. It makes sure any changes made in modification doesn't affect and introduce new defects in the entire system.

System Testing: The system was then tested as a whole with positive and negative values to observe system performance for errors and make actions that will allow the system obtain our defined goals

CHAPTER FIVE

SUMMARY,CONCLUSION AND RECOMMENDATIONS

5.1 Summary

In this project, a blockchain-based secure file security system is presented. The main aim of this proposed scenario is to provide data authenticity and quality of data to user. A decentralized storage IPFS provides the solution for bloating problem at owner's end. Data hashes returned by the IPFS are encrypted using AES encryption to encrypt the file hash with private key of the user and then it adds it to to previous hash in the blockchain to provide security so that another user cannot illegally access the data.

In chapter one, various problems associated with security in centralised storage systems looked at. The projects aim scope and objectives were then clearly defined. In chapter two, related literatures on blockchain, decentralised peer to peer networking and file security were extensively discussed and reviewed as well as a review of related works.In chapter three, a comprehensive analysis of the system requirements was undertaken.In chapter four,the proposed system was implemented and evaluated in accordance with the project goals.

5.2 Conclusions

The issue of file security was critically analysed. A file security system was developed to address these issues and provide a secure, tamper proof model for storing and sharing files in a distributed file system where a user can upload a file and secure it with a private key, view the files stored in the system by them and other nodes,download a file uploaded by them and another node in the system with condition of using their respective private keys and view all nodes presently connected to the blockchain. . The system design was achieved using block diagrams, architectural patterns, flow-chart diagram and use-case diagrams.The system was then implemented along the project goals and tested for any errors that might be incurred during this process.

5.3 Recommendations

During development and research of the system certain observations were made and the following are recommended for future projects:

- The current system method of requiring the hash of files is not feasible for future projects. Using a suitable naming system so the average user can input the file name directly.
- The system can be improved to be used as or implemented alongside regular social networking application. Sharing photos, videos and communication facilities.
- The current system is implemented as a web application but can further be improved for mobile implementation.

REFERENCES

A. Bori, P. Ekler, The analysis of bittorrent protocol reliability in modern mobile environment, in: 3rd Eastern European Regional Conference on the Engineering of Computer Based Systems, 2013, IEEE, pp. 120–126.

A. Eivy, Be Wary of the Economics of ”Serverless” Cloud Computing, IEEE Cloud Computing, Vol. 4, Iss. 2, 2017, pp. 6–12.

Algorithms, in: Shen, X., Yu, H., Buford, J., Akon, M. (eds.), Handbook of Peer-to-Peer Networking, Springer Science & Business Media, 2010, pp. 223–254.

A. Muthitacharoen, R. Morris, T. M. Gil, and B. Chen, “Ivy: A Read/Write Peer-to-Peer File System”. OSDI 2002.

Baumgart and S. Mies. S/kademlia: A practicable approach towards secure key-based routing. In Parallel and Distributed Systems, 2007 International Conference on, volume 2, pages 1–8. IEEE, 2007.

B. Cohen, Incentives build robustness in BitTorrent, Workshop on Economics of Peer-to-Peer systems, Vol. 6, 2003, pp. 68–72.

Directions, in: Shen, X., Yu, H., Buford, J., Akon, M. (eds.), Handbook of Peer-toPeer Networking, Springer Science & Business Media, 2010, pp. 223–280.

F. Chowdhury, J. Furness, M. Kolberg, Performance analysis of structured peer-topeer overlays for mobile networks, International Journal of Parallel, Emergent and Distributed Systems, Vol. 32, Iss. 5, 2017, pp. 522–548.

https://www.theregister.co.uk/2017/03/01/aws_s3_outage/

http://www.theinstitutes.org/blockchain

.J. Benet, IPFS-content addressed, versioned, P2P file system, arXiv preprint arXiv:1407.3561, 2014.

J Benet Filecoin: A Decentralized Storage Network, 2017

J. Buford, H. Yu, Peer-to-Peer Networking and Applications: Synopsis and Research,2009.

J. Buford, H. Yu, E.K. Lua, P2P networking and applications, Morgan Kaufmann,2009.

J. Nurminen, J. Nöyränen, Energy-consumption in mobile peer-to-peer-quantitative results from file sharing, in: Consumer Communications and Networking Conference, 2008, IEEE, Las Vegas, NV, USA, pp. 729–733.

K. Dhara, Y. Guo, M. Kolberg, X. Wu, Overview of Structured Peer-to-Peer Overlay

L. Liu, N. Antonopoulos, From Client-Server to P2P Networking, in: Shen, X.,Yu, H., Buford, J., Akon, M. (eds.), Handbook of Peer-to-Peer Networking, Springer Science &

Business Media, 2010, pp. 72–89.

Li, J.; Wu, J.; Chen, L. Block-secure: Blockchain based scheme for secure P2P cloud storage. Inf. Sci. 2018, 465, 219–231.

M.E. Johnson, D. McGuire, N.D. Willey, Why file sharing networks are dangerous?, Communications of the ACM, Vol. 52, Iss. 2, 2009, pp. 134–138.

M.V. Heikkinen, A. Kivi, H. Verkasalo, Measuring mobile peer-to-peer usage:Case Finland 2007, in: International Conference on Passive and Active Network Measurement, 2009, Springer, pp. 165–174.

M. Ogden, Dat - Distributed Dataset Synchronization And Versioning, 2017.

Nakamoto, S. Bitcoin: A Peer-to-Peer Electronic Cash System. 2008. Crosby, M.; Pattanayak, P.; Verma, S.; Kalyanaraman, V. Blockchain technology: Beyond bitcoin. Appl. Innov. 2016, 2, 71.

P. Baran, On distributed communications networks, IEEE transactions on Communications Systems, Vol. 12, Iss. 1, 1964, pp. 1–9.

P. Ekler, J.K. Nurminen, A. Kiss, Experiences of implementing BitTorrent on Java ME platform, in: Consumer Communications and Networking Conference, 2008, Citeseer, pp. 1154–1158.

Rowhani-Farid, A.; Allen, M.; Barnett, A.G. What incentives increase data sharing in health and medical research? A systematic review. Res. Integr. Peer Rev. 2017, 2, 4.

Vaibhav Saini : Encyclopedia of storage platforms, 2020

S. Cherbal, A. Boukerram, A. Boubetra, An improvement of mobile Chord protocol using locality awareness on top of cellular networks, in: 5th International Conference on Electrical Engineering-Boumerdes (ICEE-B), 2017, IEEE, pp. 1–6.

S. Nichols, AWS's S3 outage was so bad Amazon couldn't get into its own dashboard to warn the world, The Register, 2017. Available (accessed on 21.6.2018)

Shrestha, A.K.; Vassileva, J. Blockchain-Based Research Data Sharing Framework for Incentivizing the Data Owners. In International Conference on Blockchain; Springer: Cham, Switzerland, 2018; pp. 259–266.

Swarm, Swarm, 2018. Available (accessed on 15.7.2018): http://swarm-gateways.

T. Locher, P. Moor, S. Schmid, R. Wattenhofer, Free Riding in BitTorrent is Cheap,2006.

Tim Olson :Blockchain Pulse: IBM Blockchain Blog

Vincent Tabora: Using IPFS For Distributed File Storage Systems ,2020

Wu, A.; Zhang, Y.; Zheng, X.; Guo, R.; Zhao, Q.; Zheng, D. Efficient and privacy-preserving traceable attribute-based encryption in blockchain. Ann. Telecommun. 2019, 74, 401–411.

Xia, Q.I.; Sifah, E.B.; Asamoah, K.O.; Gao, J.; Du, X.; Guizani, M. MeDShare: Trust-less medical data sharing among cloud service providers via blockchain. IEEE Access 2017, 5, 14757–14767.

Zhang, Z.; Zhao, L. A Design of Digital Rights Management Mechanism Based on Blockchain Technology.In International Conference on Blockchain; Springer: Cham, Switzerland, 2018; pp. 32–46.