TRIBHUVAN UNIVERSITY

INSTITUTE OF ENGINEERING

PULCHOWK CAMPUS

DECENTRALIZED SECURE CLOUD STORAGE USING BLOCKCHAIN

Project Supervisor

Prof. Dr. Subarna Shakya

By:

Anish Shrestha (071/BCT/504)

Bipin Khatiwada (071/BCT/512)

Sagar Bhusal (071/BCT/551)

Sumit Chhetri (071/BCT/546)

A PROJECT WAS SUBMITTED TO THE DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING IN PARTIAL FULFILMENT OF THE REQUIREMENT FOR THE BACHELOR’S DEGREE IN ELECTRONICS & COMMUNICATION / COMPUTER ENGINEERING

DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING

LALITPUR, NEPAL

August 05,2018

LETTER OF APPROVAL

The undersigned certify that they have read, and recommended to the Institute of Engineering for acceptance, a project report entitled "Decentralized Secure Cloud Storage Platform Using Blockchain" submitted by Anish Shrestha (071/BCT/504), Bipin Khatiwada (071/BCT/512), Sagar Bhusal (071/BCT/551), Sumit Chhetri (071/BCT/546) in partial fulfilment of the requirements for the Bachelor's degree in Electronics & Communication/Computer Engineering.

Project Supervisor, Prof. Dr. Subarna Shakya Department of Electronics and Computer Engineering Institute of Engineering, Central Campus	Internal Examiner Dr. Aman Shakya Department of Electronics and Computer Engineering Institute of Engineering, Central Campus
External Examiner Mahesh Singh Kathayat Associate Professor Advisor at Kathmandu Engineering College (KEC),Kathmandu,Nepal	Head of Department Dr. Dibakar Raj Pant Department of Electronics and Computer Engineering Institute of Engineering, Central Campus

DATE OF APPROVAL:

COPYRIGHT

ACKNOWLEDGEMENT

We would also like to acknowledge Prof. Dr. Subarna Shakya, our project Supervisor who is providing us guidance and helping us with different views of application, development and scalability. His guidance has been corner stone in Cloud technology and Cryptography. We are able to accomplish this project with his valuable directions and suggestions.

We would like to express our notable thanks to Department of Electronics and Computer Engineering for providing us environment to dig the latest technology,investigate it and do research on those areas as Major Project.We are also grateful to Dr. Dibakar Raj Pant, HOD, Department of Electronics and Computer Engineering as well as our DHODs, Mrs. Bibha Sthapit for presenting useful suggestions regarding the queries put forward regarding the project.

Our special appreciation goes to Er. Sandip Pandey, Alumni of Pulchowk Campus for guiding us in this project and helping to better cope with the difficulties along with giving supportive hands.Without his valuable guidance and suggestions in Blockchain technology, it would have been difficult for us to come to a conclusion to work in this new technology.

Last but not least, we would appreciate all our teachers,seniors and friends for their support and constant inspiration which helped us to withstand in our dreams and make it come true.

Sincerely,
Anish Shrestha (071/BCT/504)
Bipin Khatiwada (071/BCT/512)
Sagar Bhusal (071/BCT/551)
Sumit Chhetri (071/BCT/546)

ABSTRACT

The project entitled "Decentralized Secure Cloud Storage Using Blockchain" is a P2P network where each node provides the storage service to the client's data. The developed system is concerned with storing parts of a single encrypted file, each part being stored on different nodes such that only client can retrieve the parts to remake the original file.

The purpose of the system is to experiment and demonstrate a working Secure File Storage System for small sized credential files, based upon the Cryptography and Blockchain technology. The P2P network is designed using the Kademlia protocol that allows each node to join and leave the network anytime, identify it's closest nodes, update it's routing table and search for any node. Using different cryptography and network algorithms a new protocol for such a system has been implemented in this project. Using blockchain to apply Service Term Agreement, files can be chunked and distributed to a completely decentralized storage network without any loss of accountability and integrity.

The project has been successfully deployed and tested for Android client by maintaining a P2P network of several nodes. However it is evident that a web application can be more feasible for large size files and Android application for small sized files and portability.

Keywords:*Decentralize Network, P2P, Shamir's Secret Sharing, Secure File storage, Cryptography, Blockchain*

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF SYMBOLS / ABBREVIATIONS

AES	Advanced Encryption Standard
DHT	Distributed Hash Table
ECC	Elliptical Curve Cryptography
ECDSA	Elliptic Curve Digital Signature Algorithm
EVM	Ethereum Virtual Machine
ICO	Initial Coin Offering
ID	Identification
SHA	Secure Hash Algorithm

1. INTRODUCTION

1.1 Background

With the growing fields of Information Technology, Internet Of Things and Digitization of every business, organizational work and projects, Information has become the biggest valuable asset for anyone. Data has become the most powerful thing in today's world. With the abundance of data and it's ever growing nature, it's equally important to store it in an organized way such that it's easily accessible and secure. For this purpose Databases are being used as a warehouse to store data.

Database play a crucial role for any individual as well as any organization and business to store its data. Realizing the importance of data and insufficiency of storage, databases are replicated, distributed and backed up in different ways. Individuals store data in the cloud provided by different privately companies. Organizations set up their data centers at different part of the globe to store its data. For the security and bandwidth, data are scattered and replicated to different servers at different places. This seems to provide a good solution for the management of rapidly increasing data. And also ensures data safety. In future, the rate of increment of data is sure to reach high. To cope with it, the current database system needs to be more reliable, safe and available all the time.

1.2 Problem Statement

1.2.1 Lack of Privacy of Data

Different cloud service companies and distributed data centre of organization ensures the data availability and safety. However,most of them have terms that allow the company to edit, modify, access, delete, view and analyze your content. This can be done to provide the best possible service to the client, create advertisement, manipulate it in some way to generate income or use for their own purpose or analysis. Data stored to a private owned database gives several access rights to that company and thus, is not always secure.

1.2.2 Data Loss

Storing sensitive data only on local machine or drives can sometime be very lamenting because once they are stolen, lost or destroyed by any other means, user cannot make a recovery. Moreover, most of the personal accounts of Cloud Storage also do not cover the insurance of data, take responsibility in case of data loss due to catastrophic failure as well as ensures data availability all the time. This is well stated on Terms and Conditions of Dropbox, Box, RapidShare,Google Drive, Amazon Cloud Drive, MS Onedrive etc. So, completely relying on data storage on your local machine only or on the cloud storage is just not always safe and genuine.

1.2.3 Financial and other losses due to Data hack

Furthermore, storing users sensitive data to cloud is not considered a good option when it comes to the high potential value of that data. Sensitive information here means: user passwords, secret keys of cryptocurrencies wallet, secret and confidential documents, sensitive informations, papers, records of banking transactions etc whose loss or hack can bring a huge disaster for any organization or individual.

1.3 Solution Proposed

For the above problem statement, the solution we propose here is to establish a distributed database system which will store data in peer to peer network such that there'll be no any central body with right to use and modify clients data.

The data will be shred to multiple chunks, encrypted using different cryptography algorithms, stored at different nodes. No any node will know what data and whose data they are storing. Even if the any attacker hacks into any node and pulls data, it'll only receive part of the data which is encrypted. So, it's efficiently hard to grab complete data in decrypted form by any attacker. Thus is more secure than Cloud.

For the data storing service that the storage nodes of the network provide, they will be rewarded, and the client will pay for that service. Furthermore, the client and Postman will be bound by the "Smart Contracts" stored in the Blockchain that will act as a proof and trust layer for data availability and storage. This way the network will sustain by reward and compensation mechanism. The integrity of the data and trust of data storage is managed by using Blockchain.

1.4 Objective

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

- To fulfill the requirements of the partial fulfillment of the Bachelor's degree in Computer Engineering, Institute of Engineering, Tribhuvan University
- To research about cryptography,P2P network,web technology and blockchain.
- To contribute in the active research on decentralized applications and cryptography.
- To develop a distributed cloud storage platform.

1.5 Scope of the Project

In this project, we propose a limited version of a working decentralised database system using decentralized network and blockchain. This project however does not include the complete bandwidth optimization algorithms, sophisticated NAT traversal algorithms and complete analysis of the security in each layer of the network stack. The scope of the project is limited to show viability of decentralized database system using blockchain and smart contracts. We'll discuss the optimization techniques in the report at the end of the project but the implementation is out of scope.

Decentralized Secure Storage is a very needed feature in today's world where high value data needs to be securely stored in web.

- The scope of this project can be found in any organisation or individual needs where security of the data or information is paramount. For example: Banks, Individuals, different Organizations etc.
- It can also be used in saving the Bandwidth of the Central Server.

- User can register for the Postman and earn reward coins by renting their Storage Devices. Thus can be also used as a earning source by any node.
- Most of the features developed here can be used fully or partially in many other applications like End to End Messenger, different Storage Service etc.

1.6 Related Theory

It is a method of changing the plain message or sentences to a scrabble sequence of characters such that it do not possess any sense without decrypting back to original form. This method is applied to convert plain message into secure form so that only the intended party can decrypt it, read and process on it. Cryptography is the major backbone of today's information security. It mainly focuses on four objectives : Confidentiality, Integrity, Non-repudiation and Authentication. Several algorithms exists to perform the cryptography. Their security level, computation power and efficiency also varies. Cryptography can be categorized broadly into three parts : Public Key Cryptography ,Secret Key Cryptography and Hashing. We will be dealing with all three parts in our project one way or another.

1.6.1 Public and Private Key Pairs

Pubic and Private Key pair are two uniquely related cryptographic keys. Neither public key nor private key will ever have multiple pairs. Private keys are often called "Secret" or "Secret Key". They are kept only by the user and is not shared to any one. Private key can generate signature of a message and only it's corresponding public key can verify it. Public keys are often called "Sharing Key" or "Address" as it can be made public to other users. Any message encrypted using public key can only be decrypted by it's private key. That means to securely send the message to a user, we can lock it using his public key. This locked message can only be unlocked by that particular user since it can be only unlocked using his private key. This is the main feature because of which this has been more popular nowadays. Elliptic Curve algorithm is one of the most used algorithm to generate such key pairs.

1.6.2 Elliptic Curve (EC) Cryptographic Algorithm

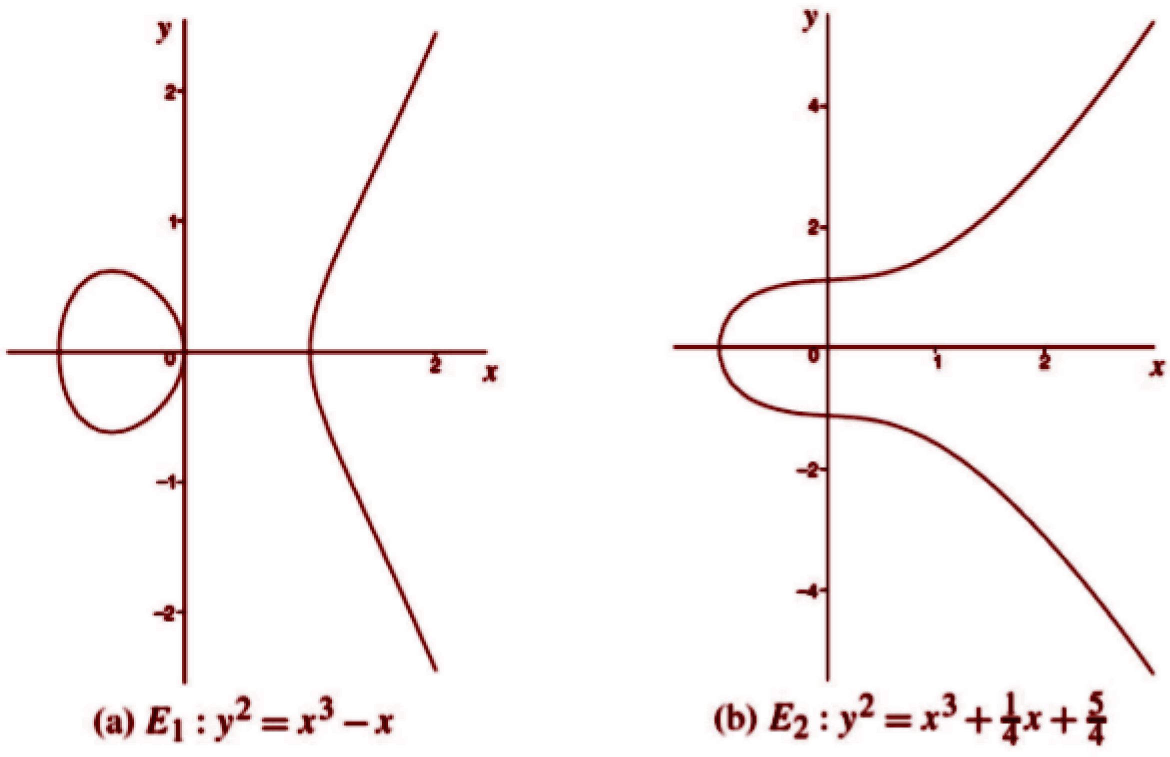


Figure 1.1: Elliptic Curve

Elliptical curve cryptography (ECC) is a public key encryption technique based on elliptic curve theory that can be used to create faster, smaller, and more efficient cryptographic keys. ECC generates keys through the properties of the elliptic curve equation instead of the traditional method of generation as the product of very large prime numbers.Because ECC helps to establish equivalent security with lower computing power and battery resource usage, it is becoming widely used for mobile applications. ECC is based on properties of a particular type of equation created from the mathematical group (a set of values for which operations can be performed on any two members of the group to produce a third member) derived from points where the line intersects the axes. Multiplying a point on the curve by a number will produce another point on the curve, but it is very difficult to find what number was used, even if you know the original point and the result. Equations based on elliptic curves have a characteristic that is very valuable for cryptographic purposes: they are relatively easy to perform, and extremely difficult to reverse.

1.6.3 Shamir's secret sharing Algorithm

As the name implies, Shamir's secret sharing was created by Adi Shamir, an famous Israeli cryptographer, who also contributed to the invention of RSA algorithm. Shamir's secret sharing is an algorithm that divides a secret into shares. Secret can be recovered by combining certain numbers of shares. **Secret** is a secret message or number that you want to share with others securely. **Share** is a piece of secret. Secret is divided into pieces and each piece is called share. It is computed from given secret. In order to recover the secret, certain number of shares are needed. **Threshold** is the number of shares needed at least in order to recover the secret. Secret can be restored only when number of secrets are more than or equal to the number of threshold.

Share Computation

- Decide secret Initially the message is converted into the byte array , thus can be treated as the number.
- Decide threshold Next thing to decide is the threshold. This is the minimum number of secrets required to recover the message or the byte array.
- Create polynomial A polynomial is chosen with any numbers for coefficient, but the degree of your polynomial must be threshold -1 . If threshold is 3, the degree must be 2. The polynomial of degree of 2 should take the form of y=ax2+bx+c. For any a and b selected, the value of c will be the secret.
- Draw graph Note that drawing graph is not necessary to do computation for Shamir's secret sharing. However, for understanding the graph of our polynomial looks like this a normal polynomial.These points are the shares. The value in x is called share number and the value of y is a share.

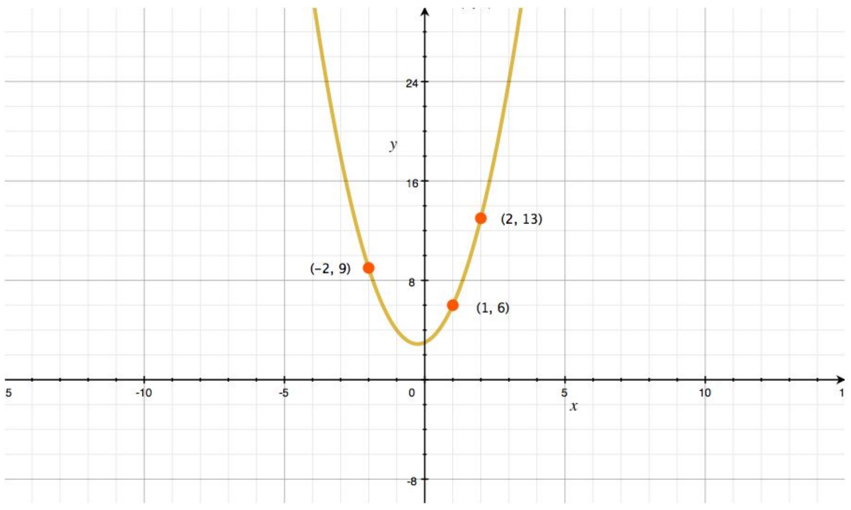


Figure 1.2: Constructing Polynomials From Secrets

Secret Reconstruction

Plotting points of shares, (x,y)(1, 6), (x,y)(2,13), (x,y)(-2, 9), in this case on the graph. Points are plotted and connected by an imaginary line.

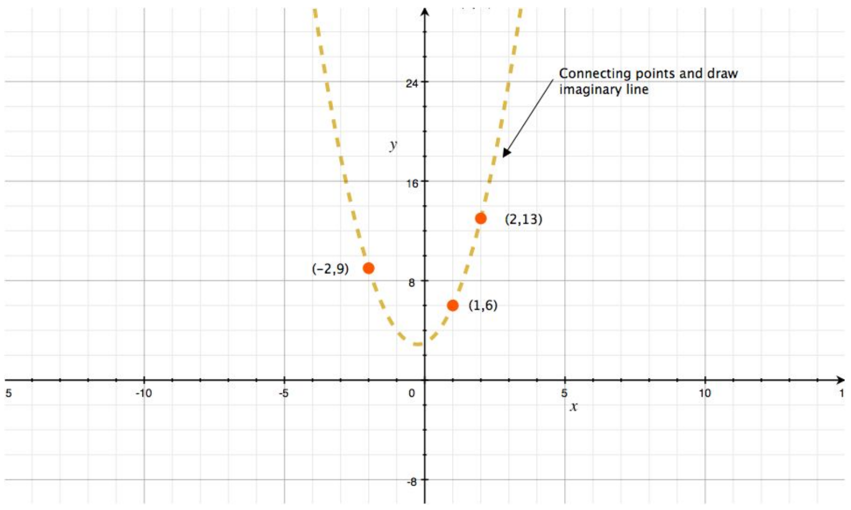


Figure 1.3: Recovering Secrets From Polynomials

1.6.4 Hashing

Hash is a unique set of characters or an array of bytes derived from a function which intakes certain message or plain text. Each message has unique hash that depends on the hash function used to derive the hash. A slight change in any character of input will produce an entire different hash, meaning that it is impossible to trace any pattern in hashing and also impossible to obtain the original text from it hash value. The hash value is always of same size in most of the cases. For our project, we will be using SHA-256 hashing.

The SHA (Secure Hash Algorithm) is one of a number of cryptographic hash functions. A cryptographic hash is like a signature for a text or a data file. SHA-256 algorithm generates an almost-unique, fixed size 256-bit (32-byte) hash. Hash is a one way function. It cannot be decrypted back. This makes it suitable for password validation, challenge hash authentication, anti-tamper, digital signatures. SHA-256 is one of the successor hash functions to SHA-1, and is one of the strongest hash functions available.

1.6.5 Distributed Hash Table(DHT)

Hash Table is data structure capable of storing (key,value) pairs and performing value lookup when key is provided.The insertion and lookup of (key,value) pair is very fast in the order of O(1) as they internally use arrays to keep the data.But the array is always larger than the total no. (key,value) pair which results in slower iteration over the items in hash table.

Distributed Hash Table is a distributed service that provides (key,value) pair like Hash tables. Each node participating in the distributed network maintains only a fixed number of (Key,Value) pair. A method or rule is made such that when a key is known, the peer which might have the value can be determined. The value is asked with one or more such peers. If the peer doesn't have the value it asks with other peers until the value is retrieved.

1.6.6 Kademlia

Kademlia is a communications protocol for peer-to-peer networks. It is one of many versions of a DHT, a Distributed Hash Table.Kademlia uses a "distance" calculation between two nodes. This distance is computed as the exclusive or (XOR) of the two node IDs, taking the result as an integer number. Keys and Node IDs have the same format and length, so distance can be calculated among them in exactly the same way. The node ID is typically a large random number that is chosen with the goal of being unique for a particular node . It can and does happen that geographically widely separated nodes from Germany and Australia, for instance can be "neighbors" if they have chosen similar random node IDs.

- **Network Characterization**

A Kademlia network is characterized by three constants, which we call **alpha**, **B**, and **k**. The first and last are standard terms. The second is introduced because some Kademlia implementations use a different key length.

- alpha is a small number representing the degree of parallelism in network calls.
- B is the size in bits of the keys used to identify nodes and store and retrieve data.
- k is the maximum number of contacts stored in a bucket
- **Node** A Kademlia network consists of a number of cooperating nodes that communicate with one another and store information for one another. Each node has a nodeID, a quasi-unique binary number that identifies it in the network.Within the network, a block of data, a value, can also be associated with a binary number of the same fixed length B, the valu's key.A node needing a value searches for it at the nodes it considers closest to the key. A node needing to save a value stores it at the nodes it considers closest to the key associated with the value.
- **NodeId** NodeIDs are binary numbers of length B = 160 bits. In basic Kademlia, each node chooses its own ID by some unspecified quasi-random procedure. It is important that nodeIDs be uniformly distributed; the network design relies upon this. While the protocol does not mandate this, there are possible advantages to the node's using the same nodeID whenever it joins the network, rather than generating a new, session-specific nodeID.
- **Key** Data being stored in or retrieved from a Kademlia network must also have a key of length B. These keys should also be uniformly distributed. There are several ways to guarantee this; the most common is to take a hash, such as the RIPEMD160 digest, of the value.
- **Kademlia Metrics** Kademlia's operations are based upon the use of exclusive OR, XOR, as a metric. The distance between any two keys or nodeIDs x and y is defined as distance(x, y) = x ŷ where r̂epresents the XOR operator. The result is obtained by taking the bytewise exclusive OR of each byte of the operands.
- **Kademlia Protocol** Kademlia has four messages.
    - PING: Used to verify that a node is still alive.

- STORE: Stores a (key, value) pair in one node.
- FIND_NODE: The recipient of the request will return the k nodes in his own buckets that are the closest ones to the requested key.
- FIND_VALUE: Same as FIND_NODE, but if the recipient of the request has the requested key in its store, it will return the corresponding value

1.6.7 Blockchain

Blockchain is the growing list of records called blocks, containing structured information linked using the art of cryptography distributed globally . Each block in blockchain contains the cryptographic hash of previous block along with timestamp and data which typically varies with use-cases.

Merkle trees are the fundamental part of blockchain.Every block contains the block header which is outcome of recursive cryptographic hashes of all the data nodes or transaction from bottom to up approach. During hash generation, the order of data matters for the final hash of the block. If single detail of transactions or order of transaction changes then changes the merkle hash.Therefore, Merkle Root summarizes all of the data in the related transactions, and is stored in the block header. This feature makes blockchain resistance to modification which is considered secure by design.

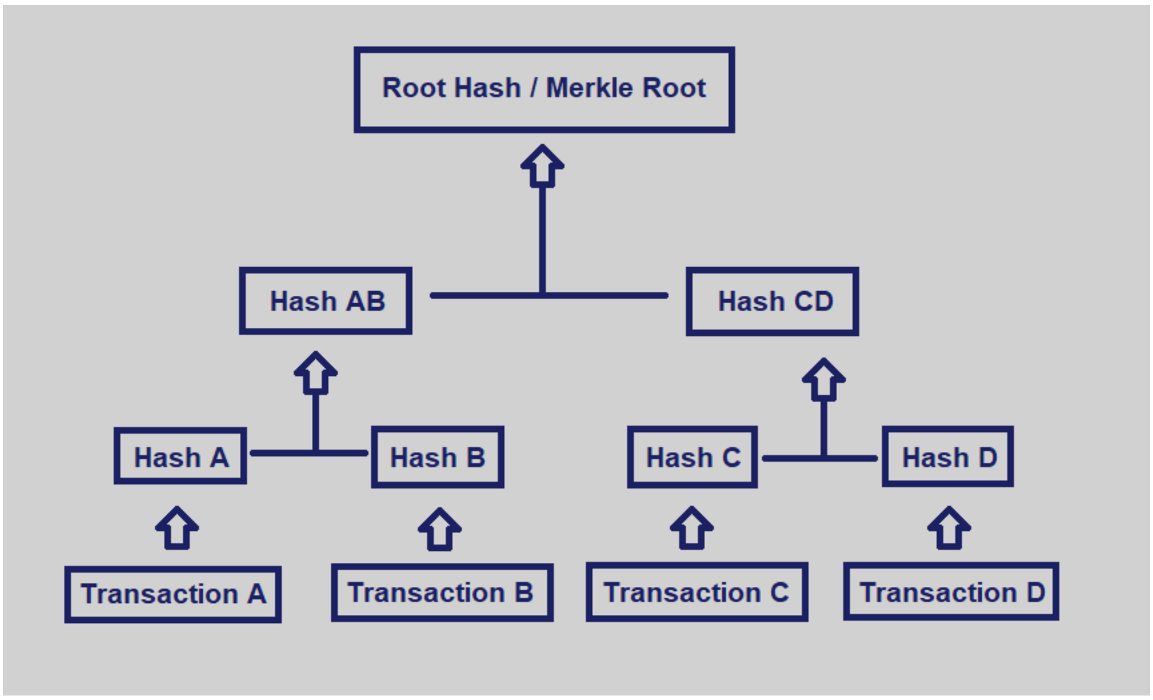


Figure 1.4: Merkle tree

Blockchain is P2P network which relies on protocol for inter-node communication and validating the blocks. Blockchain was invented by Satoshi Nakamoto in 2008 to serve as the public transaction ledger of the cryptocurrency bitcoin. Blocks in blockchain are the holder of valid transactions that are hashed and encoded in Merkle tree. Every block contains the cryptographic hash of previous blocks along with its own data which therefore forms the chain. This iterative mechanism in each block conforms the integrity of previous block all the way back to genesis block.

Block time is the average time it takes in the network to generate 1 extra block in blockchain.By the time block is generated, the data of that block is verified. This means lesser the block time, fasre the transactions. A hard fork is a rule change such that the software validating according to the old rules will see the blocks produced according to the new rules as invalid. Storing data in P2P network allows Blockchain to eliminate the pitfalls of centralization. There will be no central point vulnerability ,no center point of faliure in blockchain. It is open to public which makes it more user- friendly than traditionally owned records. Being permissionless and open, there is no need to guard against bad actors.

- Ethereum Ethereum is an open-source, public, blockchain-based distributed computing platform and operating system featuring smart contract(scripting) functionality. Ether is a cryptocurrency whose blockchain is generated by the Ethereum platform. Ethereum provides a decentralized Turing-completevirtual machine, the Ethereum Virtual Machine (EVM), which can execute scripts using an international network of public nodes. "Gas", an internal transaction pricing mechanism, is used to mitigate spam and allocate resources on the network. Ethereum address are composed of the prefix”0X” a common identifier for hexadecimal, concatenated with the rightmost 20 bytes of the Keccak-256 (SHA-3)hash (big endian) of the ECDSA(Elliptic Curve Digital Signature Algorithm) public key.

- Smart Contract Ethereum’s smart contracts are based on different computer languages, which developers use to program their own functionalities. Smart contracts are high-level programming abstractions that are compiled down to EVM bytecode and deployed to the Ethereum blockchain for execution. They can be written in Solidity (a language library with similarities to Cand JavaScript), Serpent (similar to Python, but deprecated), LLL (a low-level Lisp-like language), and Mutan (Go-based, but deprecated). There is also a research-oriented language under development called Viper (a strongly-typed Python-derived decidable language).
- ERC20 Token ERC-20 is a technical standard used for smart contracts on the Ethereum blockchain for implementing tokens. ERC stands for Ethereum Request for Comment, and 20 is the number that was assigned to this request. The clear majority of tokens issued on the Ethereum blockchain are ERC-20 compliant.
  The ERC20 token standard describes the functions and events that an Ethereum token contract has to implement.

2. LITERATURE REVIEW

2.1 Existing Data Storage System

Currently, user uses his/her offline storage devices and other secondary storages for data backup and protection. Most of us often use the cloud service of Amazon, Google, Dropbox, Microsoft and others. A huge amount of users data are stored in cloud which is in fact someone's computer or storage devices. Such organization has complete authority over users data. In recent years, trend has increased rapidly in using those data without users acknowledge and permission by those company for their uses and pursue higher benefits from it.

2.2 Need of a distributed database

With ever-growing technological advancement and shrinking size with more power devices, there has never been a better time for advancements in an information system made up entirely of distributed devices. Of course we have the internet as an example. However, internet itself maintains a general hierarchy of client-server and a lot of middle men which may or may not be trusted. The devices grow, data grow and so do the need for physical as well as logical means to hold the data. With a new race for powerful organizations to gather as much data as possible for future manipulation and understanding of information in a gigantic scale, there has been an unprecedented search and store of data like never before. Even in personal scale, we associate data to our personal lives as digital data has never been more of our personal life's metadata like today. Hence, they say data is the new currency, data is the new knowledge. [1] [2]

However, how do we store our personal data? We store it locally, we burn it into DVDs, we use cloud services, we store it as much as we can and then leave the rest for trusted servers to put. We have come to an age where our crucial data - we store it in some server's storage space provided by a company with a promise security and integrity. People now pour more trust in such services than themselves to store our data with security and integrity. However, in every system with central authorities there is a hierarchy of power and in this case such misuse directly affects our crucial data we use in our lives. Giving the key to an entity of higher authority in a system can not last long without an imminent risk. One can not guarantee the replication and misuse of such data by such authorities.

However, with the technological advancements, computation power, data transfer rates and storage space distributed to each and every people in forms of many devices. there need not be networks with central authorities and hierarchies to where we trust our data for safe keeping and transfer. Instead we can do what we have been doing for the past decade now in completely distributed systems maintained by parties involved using the system and with Blockchain technology we can do this with trust.

2.3 Necessity of blockchain for a distributed storage system

In 2008, after the global financial crisis, a new paper brought about by person or a group named Satoshi Nakamoto introduced a concept of a peer-to-peer distributed currency with a paper titled “Bitcoin: A peer-to-peer Electronic Cash System” [2]. With there usually being a central authority bearing a higher power in a system may it be in economics, finance, technological or general, there exists a risk of the central authority misusing its power for one’s gain while other’s loss. The paper talked about a distributed digital crypto-currency which could improve how we did things and many regarded as a better system.

Hence, Blockchain with its social perceptions related to Bitcoin not only solves the problem of “Double Spending” but provides us a solution for a problem with this specific criterias:

- Possibility of Fraud.
- Intermediaries or a middle man.
- Throughputs (Number of transactions/sec).
- Stable data.

Hence, called the FITS model which defines a environment where there is a possibility of Fraud, a middle man for transactions and exchange of other resources, transactions occurring in many number per seconds and a stable data i.e. data that is not constantly changing.

However, taking this step further, Blockchain allows for what is known as a Distributed Autonomous Organization where it is an organization self-sustained by members without a central authority and trust built around what is known as “Smart contracts”. [3]

Hence, using blockchain technology we can build a system far exceeding the typical client server system in terms of application and scope but with the same amount of trust we put in a centralized server. Like a cloud platform where you are storing your data. You will trust the server of the given cloud service provider to hold the data without tampering and betrayal of trust and make it available for use any time you want. This trust is built by the company itself with continuous service and a good policy. However, in a distributed service this is hard to do as there is no central authority

However, with blockchain we can implement a distributed system with a trust system. The same kind of trust people put in data storage services like a dedicated server or even simple cloud storage applications like Google Drive. Hence, for an implementation of a distributed storage network maintained by the people and for the people to use without an authority that provides trust but have trust built into the system, technology like blockchain is crucial. [4]

2.4 Blockchain against data tampering

In blockchain, we store the hashes of files - metadata and its hash, the transactions between users, who the data belongs to, who is storing the data, which other parties are involved during the replication and storage process and access control data.
Encryptions with Asymmetric keys paired and shared secrets, we will implement a layer of security and restriction. Using such approach, one can be sure that one's data is safe and not accessible and readable by undesirable parties.

2.5 Existing Decentralized Database System using Blockchain

Although blockchain technology has been developed more than 2 decades ago, its actual implementation in various technology fields is just blooming. There are few such examples of distributed database using blockchain. Some of the most popular applications of blockchain in distributed database management are storj.io, bigchaindb etc. We here discuss BigchainDB,Storj.io , Sia and MaidSafe as it is more somewhat related to our project.

BigchainDB: a scalable blockchain database that uses Blockchain technology to store the users data in various nodes. BigchainDB inherits characteristics of modern distributed databases: linear scaling in throughput and capacity with the number of nodes, a full-featured NoSQL query language, eïňČcient querying, and permissioning. Being built on an existing distributed DB, it also inherits enterprisehardened code for most of its codebase. [5]

Storj aims to become a cloud storage platform that can't be censored or monitored, or have downtime. It is the first decentralized, end-to-end encrypted cloud storage that uses blockchain technology and cryptography to secure users files. Storj is a platform, cryptocurrency, and suite of decentralized applications that allows you to store data in a secure and decentralized manner. Files are encrypted, shredded into little pieces called 'shards', and stored in a decentralized network of computers around the globe. [6]

Filecoin and Sia are other two decentralized database systems that both support smart contracts on the blockchain that set the rules and requirements for storage, while Storj does not; Storj users pay what they use. This particular payment model means that if a user disappears, the host will no longer be paid for lending their space, a potential problem for those who will be renting their storage space. [7]

MaidSafe also aims to do more on its network than trade storage; it markets itself as a "crowdsourced internet", on which not only data is stored but decentralized applications live. Miners rent out their unused computing resources to the SAFE network, including hard drive space, processing power and data connection and are paid in the native Safecoin. The SAFE network also supports a marketplace in which Safecoin is used to access, with part of the payment going to the application's developer. Miners can also sell the coins that they earn for other digital currencies, and these transactions can happen either on the network or directly between individuals. [7]

Storj, Sia, MaidSafe and Filecoin are all built with a native storage marketplace where users and hosts can buy and sell storage space. All use mining to provide computing power for the network. In Filecoin, not only are miners given token rewards for hosting files, but they must prove that they are continuously replicating the files for more secure storage. They are also rewarded for distributing content quickly the miner that can do this the fastest ends up with the tokens. In Maidsafe's network dubbed SAFE Safecoin are paid to the user as data is retrieved; however, this is done in a lottery system where a miner is rewarded at random. The amount of Safecoin someone can earn is directly linked to the resources they provide and how often their computer is turned on. [7]

2.6 How our Project will differ from Existing systems?

Many of the system workflow matches with the aforementioned system protocols. However, this system is more superior and is far better than other existing systems. Our system will be differ from them in following manners:

- It is more focused on storage of keys, passwords, secret keys of cryptocurrency wallets, credentials , important and secure file systems. So, our system will be focused more on securing users data than just providing mass data storage. However, it can be used to store any amount of data.
- Since we are developing it over Ethereum blockchain, Ethereum coins can be used to pay and receive in transaction. Since it is widely used currency. Users do not have to invest in ICO separately to buy coins specially for this system.
- Since the transaction period of Ethereum is less (20 seconds in ideal case) , it will be relatively faster than other systems like Storj.

3. METHODOLOGY

3.1 Building P2P network

3.1.1 Peer to Peer Network

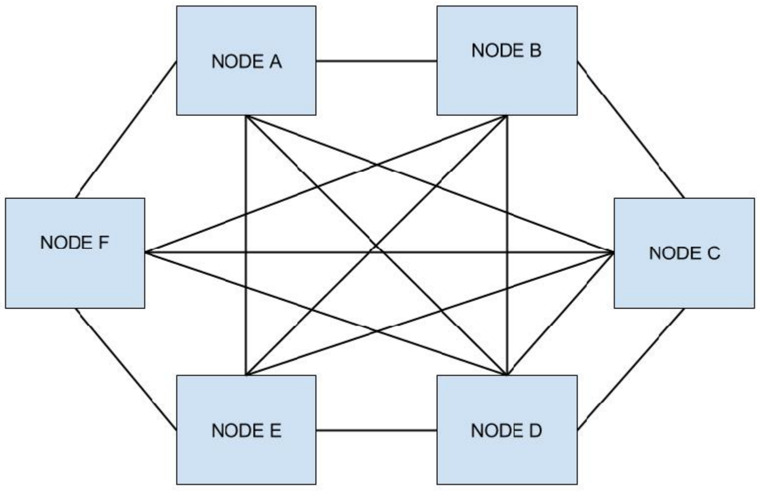


Figure 3.1: P2P Network

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

3.1.2 Creating Network and DHT

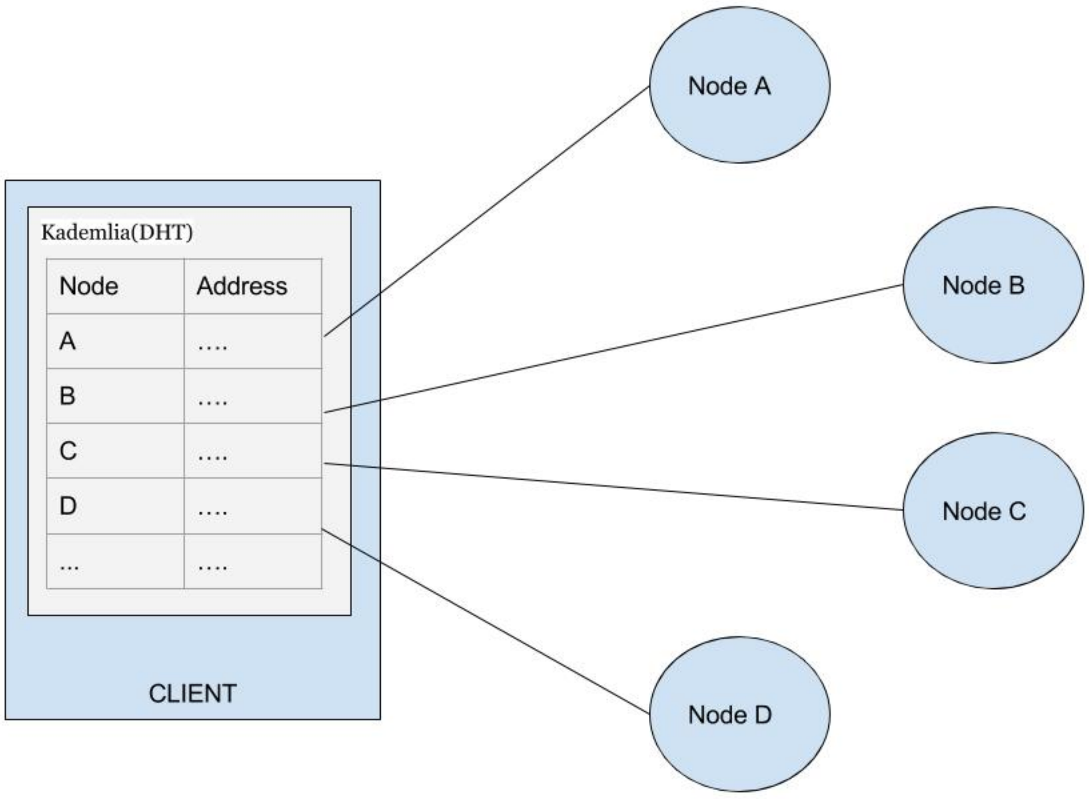


Figure 3.2: DHT

Initially user registers to the network. It’s address will be stored in the DHT. A Peer to Peer network will be formed combining all such nodes connected to the network. Each client will get his private and public key. The address can be generated from it’s public key or public key itself can be used as it’s address. The system will use Kademlia to store and find out the node address and thus create a channel for communication. The DHT will be used to identify the nodes required and store or retrieve the data. Thus, the network will basically be made of 2 parties: those who host space and those who use provided space for data storage. Operations of data flow and storage between such parties is bound by “Smart Contracts”. Such transactions are done with the exchange of tokens which is a virtual currency of the network.

- Configuring Kademlia Our DHT implementation mainly uses kademlia protocol. Few

DHT operation are implemented with different approach for efficiency. Each Node in the DHT network is identified by a 160 bit identifier. The identifier comes from hash of publicKey of the node. The value is same as a ethereum address. Thus the identifier of user or a node is same in DHT Network and in the Ethereum Blockchain network

- Components of Kademlia The components of Kademlia are listed below:
  - **ContactBucket**

    Contact Bucket class is responsible for storing the details about nodes in the DHT network. The information can be retrieved from identifier of node. The contact bucket consists of 160 slots and each slot can contain multiple no of nodes. The no. of nodes that can go into one slot is given by a parameter 'K' of the network. Commonly 'K' parameter is set to 2 or 3. Higher values of K allows to store more contacts. A contact always goes into a specific slot in the bucket. The position of a contact in bucket is determined from following calculation.

    Let,
    Id of local node = L
    Id of peer node = P
    XOR_value = L (XOR) P

    Position of peer in the bucket = Position_of_most_significant_bit_in (Xor_value).

    Since the xor value is symmetric, the position of L calculated by P and the position of P calculated by L is same. Thus negative distances need not be dealt with. Similary the total no of possible nodes that can go into a slot is higher for higher index for slot. The possible total no. of contacts in each slot is :

    For Slot 0 : 2^0 - 1 = 0
    For Slot 1 : 2^1 - 1 = 1
    For Slot 2 : 2^2 - 1 = 3
    For Slot 3 : 2^3 - 1 = 7
    For Slot 4 : 2^4 - 1 = 15
    For Slot 5 : 2^5 - 1 = 31
    For Slot 159 : 2^{159} - 1 = 7.3 * 1047

    Since the no of contacts stored in each contact is fixed, if the contact in DHT has higher distance, it might not be included in the Contact Bucket. For contacts with smaller distance, there's high possibility of having it in Bucket. A contact contains of following information.

  * Id of the node
  * Public IP and port of the node
  * Time when the node was last active

- **Message Dispatcher** Message Dispatcher class is responsible for sending and receiving messages from network. Message for each different purpose is available as a extension of MessageClass. The message is serialized in a predefined format and a TCP/Udp packet is made using the serialized data. Then it is sent to the respective destination. MessageDispatcher handles everything required for concurrently sending and receiving messages in a separate thread Pool. We have written both UDP based and TCP based Message Dispatcher classes, but we later chose to use the UDP based Dispatcher as there was no need to setup connections. Each message was represented by a byte array thus we didn't need the stream based protocol that TCP provided.

  Furthermore, using UDP makes it possible to use NAT traversing techniques. Nodes can then be setup without having a public IP.

  Message Dispatcher class also performs the task of updating Contact Bucket when a node sends message. It reports when a new connection is received and when the ip address of a node changes.

- **TimeStamped Store** TimeStamped Store is the HashTable part of DHT. TimeStamped Store keeps the (key,value) pairs along with the entry and expiry time. The "key" part is obtained from RIPEMD160 of the original key. Like the Contact Bucket, Timestamped Store contains 160 different slots labeled by unique index 0 - 159 . The slot in which (key,value) pair goes is determined by the xor based arithmetic between id of the node and they value of key.

  Xor_distance = node_id (xor) key

  Slot No. of (key,value) = Position of most significant bit in (Xor_distance)

  Let's say the xor_distance is calculated to be binary value of '1'. The Slot index to which it goes will be ‘0’, as the most significant bit is at ‘0’ position. For ‘1010’, the slot index will be 3. For higher index of slot the no of Keys it needs to accommodate grows exponentially. A Node can decide when to limit the storage.

- Operation In DHT
  - **Find Node** As name suggests, find node fetches information of a node based on the node ID. If the information is available in local Contact Bucket, the information is simply returned. Otherwise, the information must be fetched from other nodes in the DHT. The purpose of xor distance based positioning in Contact Bucket serves for this purpose. When searching for a node in DHT, we follow following steps.
    - select alpga A no of nodes from our Contact Bucket that are closest to the required node.
    - Ask each of nodes to return alpha nodes closest to the required node in their bucket. The message sent is serialize as ‘FindNode’ message
    - From the newly returned list of nodes, we again select alpha no. of nodes that are closest to the required node and yet not queried. The returned nodeList is serialized as ‘NodeListMessage’.

      Above process is repeated until we find the node, or we are out of nodes to query. In each iteration we find more and more closer nodes to the required nodes. If the node is not found, the node it is concluded that the node with given id has not entered in the network.
  - **Put** Put operation means to add a (key,value) in the DHT network. When putting a value in the network, we find closest nodes that are currently in the network to the key. The closeness is calculated from the xor distance metric. Similar to the findNode algorithm, we select fixed no of nodes initially and query them for closer nodes. A redundancy parameter is initially set to limit which nodes will store the values. Let’s say if the redundancy parameter is 3 then only 3 closest nodes in the network will be requested to store the (Key, Value) pair. The request is sent in the form of ‘DataMessage’. The (key,value) pair will be stored by each node in their own TimeStamped store structure.

- **Get** Get operation servs for finding a value from known key. When Getting a value from DHT network, first alpha no of nodes are selected from the local ContactBucket. Then the algorithm is as follows.
    - ∗ Send each alpha nodes a ‘LookUpMessage’.
    - ∗ If the node receiving ‘LookUpMessage’ has the (key,value) pair in it’s store, it will reply a ‘DataMessage’. Otherwise, It will send ‘NodeListMessage’ containing the nodes closest to given key in it’s own Contact Bucket.
    - ∗ Add receivd ‘DataMessage’ into a list.
    - ∗ Update closest nodes if ‘NodeListMessage’ was received.
    - ∗ If we have received more closer list of nodes, repeat the process
    - ∗ Finally, among the received ‘DataMessage’ select the latest one and update Local Store with it.
    - ∗ Return the latest value.
- **Ping** Ping operation is used to detect if a node in DHT is still active. If the node to ping doesn’t exist in our local Contact Bucket, FindNode() operation is carried out to find the node. If the node is found, we then send a ‘Ping Message’ instance to the node. If the node replies with a ‘Pong Message’ within a time limit, the node is considered alive in the network.
- **Join** Join operation is used when a node starts for the first time or after a shutdown. For join operation ip address and port of a node already Existing in the DHT must be known. Then the FindNode operation is used with the node’s own id. Thus Contact Bucket gets populated with new nodes.

3.2 Node Setup

The following diagram shows the basic workflow of node in our system.

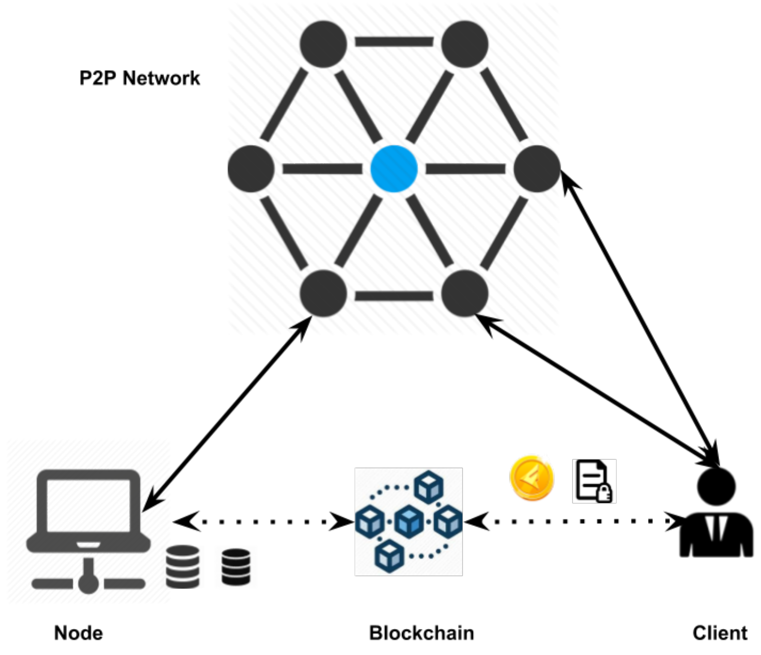


Figure 3.3: Node Workflow

3.2.1 Introduction of Node

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

3.2.2 Node Operations

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

3.2.3 Motivation For Node

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

3.3 Client Setup

3.3.1 Communication with Node

3.3.2 File Processing

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

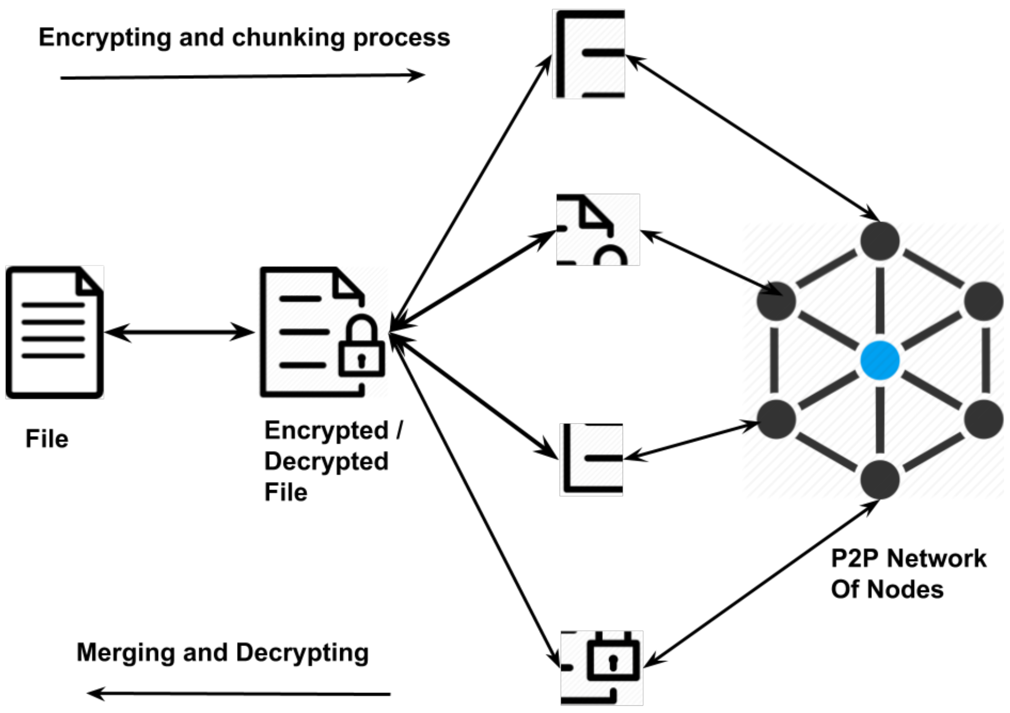


Figure 3.4: File Processing

3.3.3 AES encryption/ decryption

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

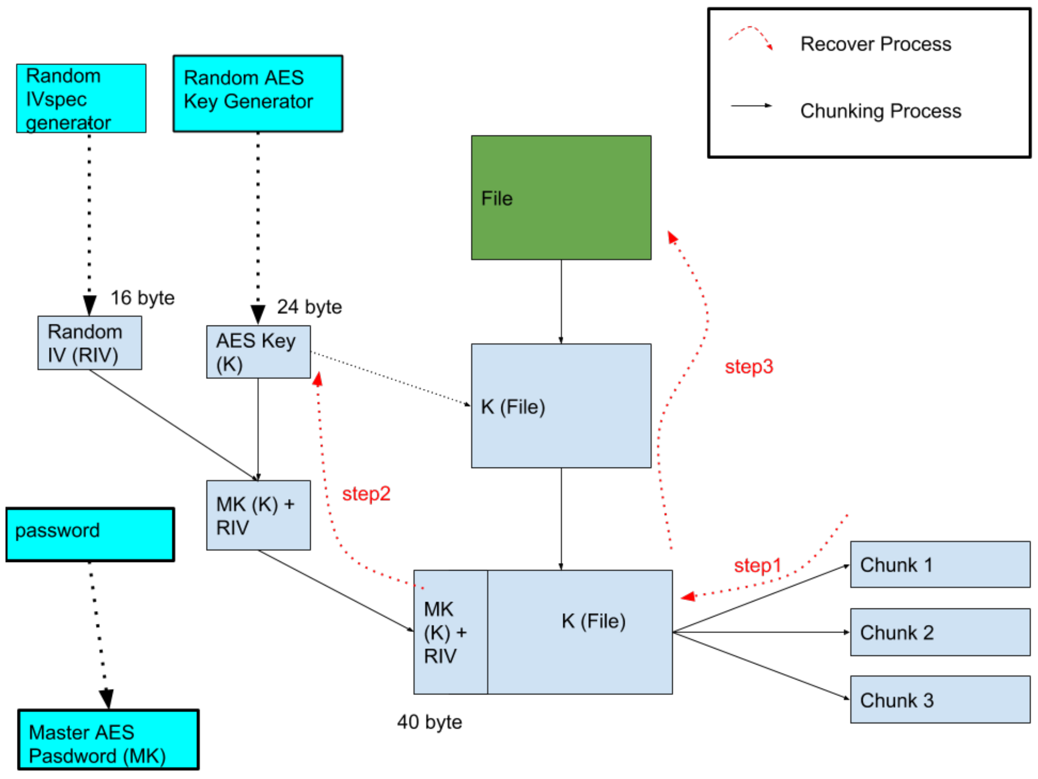


Figure 3.5: Secure File Storage Protocol

3.3.4 File splitting /merging

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

3.4 Blockchain Integration

3.4.1 Smart Contract Development/Contract Deployment

Smart contract acts as an agreement between client and postman during the transaction of data.Therefore, we have tried to ensure(look) from both parties during its development. We have tried to keep as less data as possible in blockchain network without compromising services.Smart Contract act as a global database which gets deployed in the blockchain.
There are 2 roles in our contract

- Node : stores client data,earns tokens .
- Client : stores data on node, pays tokens.

This diagram shows the workflow of client and node in smart contract.

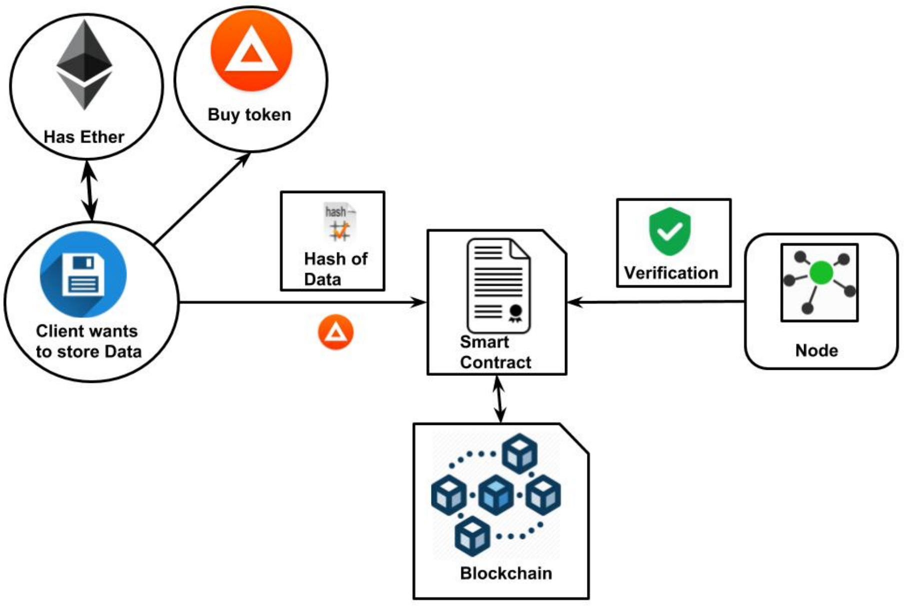


Figure 3.6: Smart contract flow diagram

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

3.4.2 File Details Record

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

- Address with FileDetail Array

  It tracks the owner of file along with array of owned file details by each individual.File Details contains the hash of file and it's filename.

- File Hash with Chunk Hash Array

  It tracks the chunk of file array associated with the main file so that when file hash is known, its chunk can be traced.

- Chunk Hash with File hash

  Every node doesn't have file hash where as smart contract needs file hash for getting chunk info. This data gap is fulfilled by this data structure where once chunk hash is known, file hash can be known immediately.

- Chunk hash to index/File hash to index

  Looping in smart contract is very expensive. Therefore for every index of chunk hash in array is mapped with it's chunk hash so that once chunk hash is known it's index can be immediately known. Like that, client can have number

- ChunkIndex to Download index

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

3.4.3 Token transfer / Payment

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

4. SYSTEM DEVELOPMENT METHODOLOGY

4.1 Software Development Approach

Making huge and dynamic system using traditional approach such as waterfall model of software development cost more time and manpower. Therefore to meet the requirements of the system with more flexibility and timely deliverly, we have choosen Scrum methodology under the Agile Development method.

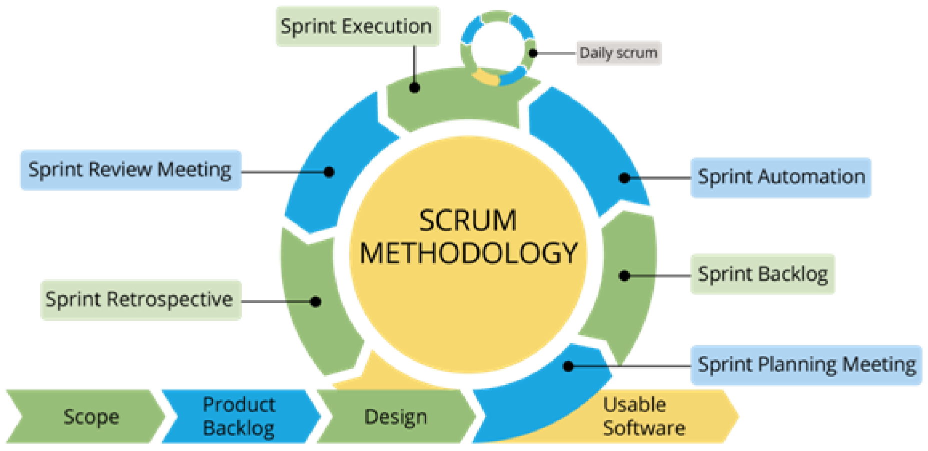


Figure 4.1: Scrum-methodology

Instead of providing complete, detailed descriptions of how everything is to be done on a project, much of it was left up to the project development team. The scrum team is self-organizing in that there is no overall team leader who decides which person will do which task or how a problem will be solved. So, each time every team member tried to solve problems and find solutions.And in Scrum, a team is cross functional, meaning everyone is needed to take a feature from idea to implementation. Project progress were shown via a series of sprints. In keeping with an agile methodology, sprints are timeboxed to no more than a month long, most commonly two weeks. We met at every start of sprint and figure out each individual commits of task and created backlog to keep track of work. We commonly had daily meetings duing college team for discussing project's problems and solutions. Meeting duration was no more than 10-15 minutes. This helped all team member to be fully synchro-

nized which is one of the main benefit of Scrum methodology. We made sprint burndown chart and release burndown chart which helped us to know remaining work and track the overall project schedule.
Hence scrum is adpative, has small repeating cycles and there is short-term planning with constant feedback, inspection and adaption and is therefore chosen as the software development methodolgy

4.2 Requirement Analysis

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

4.2.1 Function Requirement

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

4.2.2 Non Function Requirement

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

4.3 System Level Architecture

4.3.1 User Level Architecture

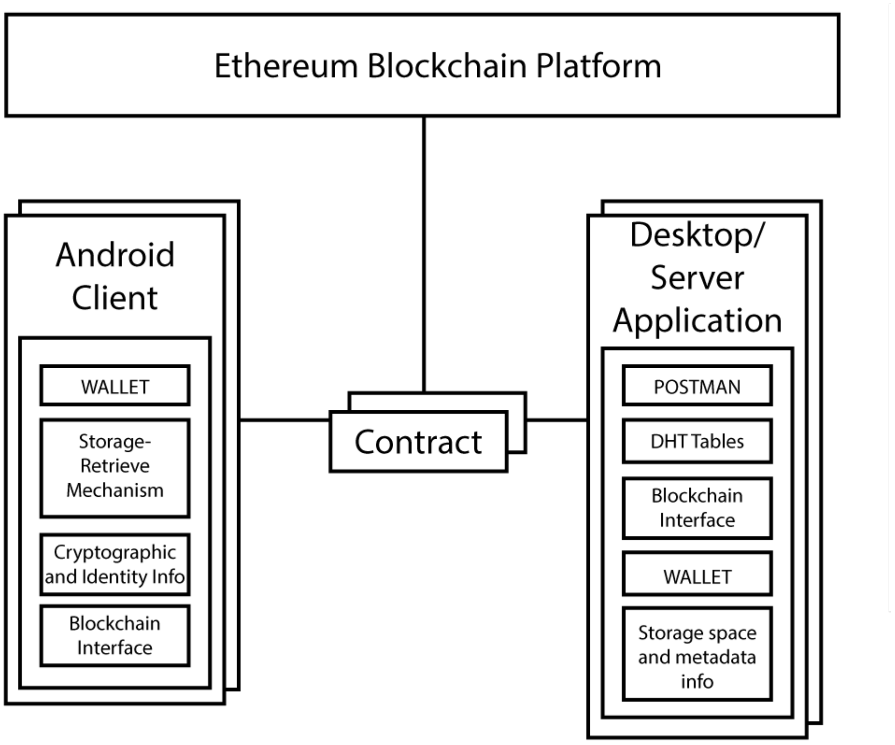

Ethereum Blockchain Platform
Android Client
WALLET
Storage-Retrieve Mechanism
Cryptographic and Identity Info
Blockchain Interface
Contract
Desktop/ Server Application
POSTMAN
DHT Tables
Blockchain Interface
WALLET
Storage space and metadata info

Figure 4.2: User Level Architecture

4.3.2 Core Architecture

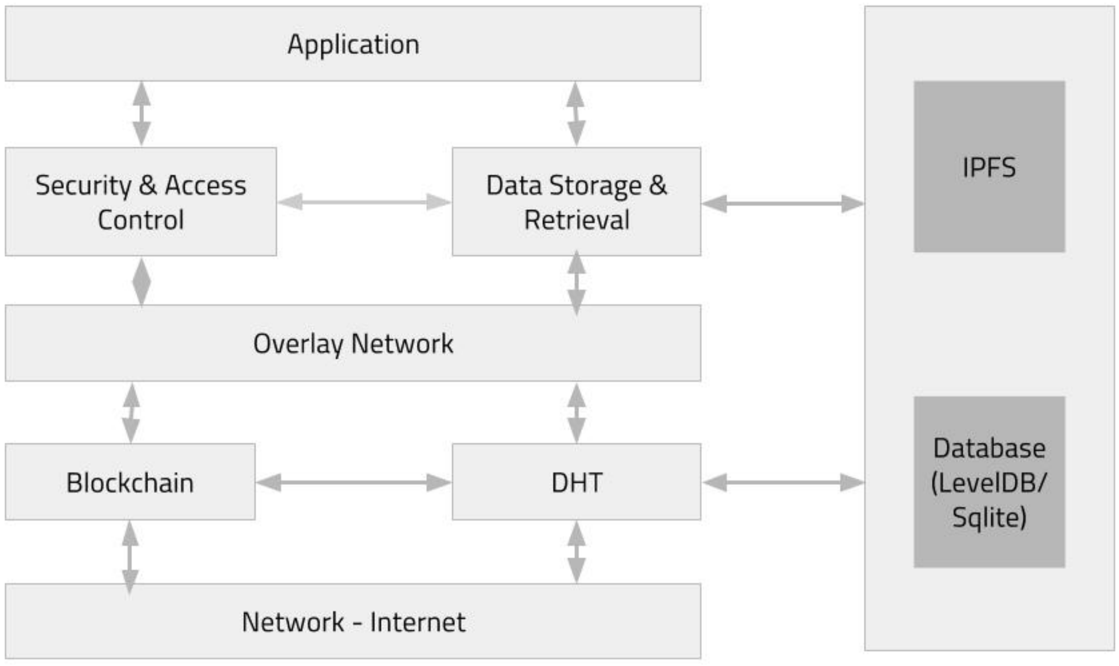


Figure 4.3: Core Architecture

4.4 Diagram

4.4.1 UseCase Diagram

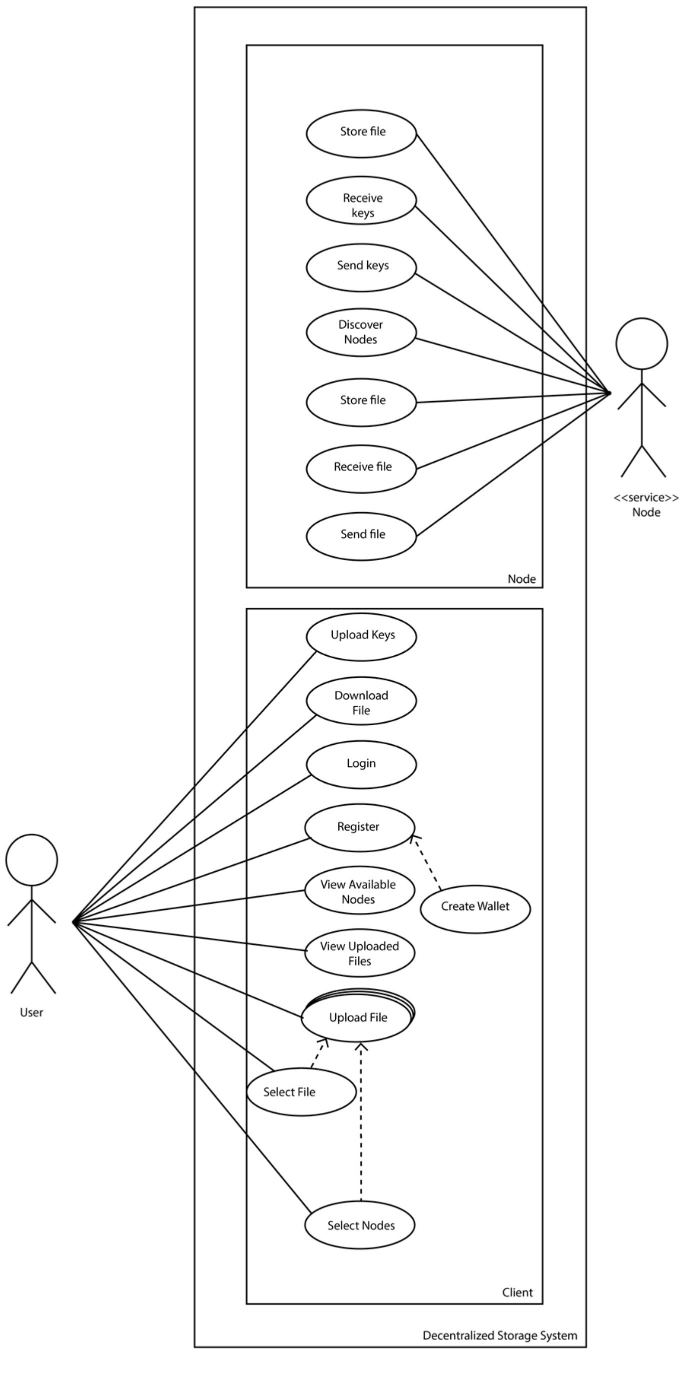


Figure 4.4: Usecase Diagaram

4.4.2 ERD Diagram

diagram.png

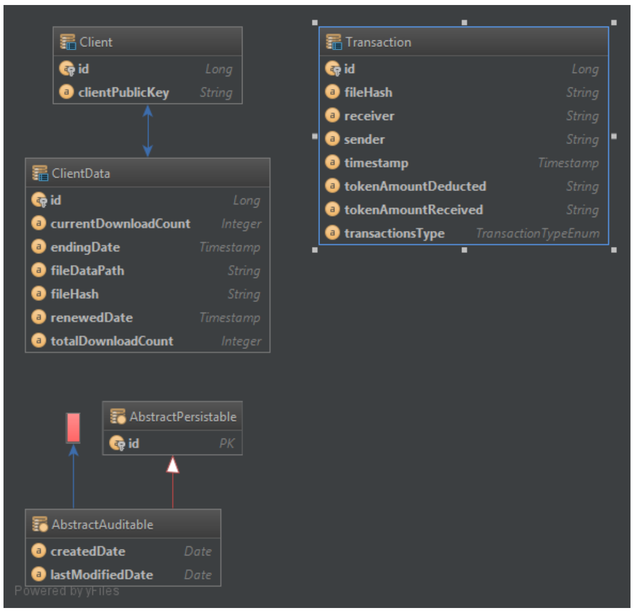


Figure 4.5: ERD Diagaram

4.5 Tools and Techniques

4.5.1 Java

Java is one to the popular general purpose computer language that is concurrent, class-based and Object Oriented Programming(OOP) which runs in Java Virtual Machine(JVM).It is intended to let application developers "write once, run anywhere" (WORA),] meaning that compiled Java code can run on all platforms that support Java without the need for recompilation. Java applications are typically compiled to bytecode that can run on any Java virtual machine (JVM) regardless of computer architecture. The Oracle implementation is packaged into two different distributions: The Java Runtime Environment (JRE) which contains the parts of the Java SE platform required to run Java programs and is intended for end users, and the Java Development Kit (JDK), which is intended for software developers and includes development tools such as the Java compiler, Javadoc, Jar, and a debugger.

4.5.2 Spring

It is application framework and inversion of control(IOC) for java platform.Any application can be built on top of the Java EE(Enterprise Edition) platform but it typically focuses on

Java applications.This framework is popular in the Java community and it is open source. It provides comprehensive programming and configuration model for java-based applications. It provides an IOC container which provides consistent means of configuring and managing Java objects using reflection. Container is responsible for managing object life cycles : creating,initializing and configuring these objects.Spring Framework has its own Aspect Oriented Programming(AOP) that modularizes cross-cutting concerns in aspects. Spring AOP is proxy-based pattern and configured at run time.

4.5.3 Android

Android is one of the popular mobile operating system developed by google based on a modified version of the Linux kernel and other open source software and designed primarily for touchscreen mobile devices such as smartphones and tablets.Android's default user interface is mainly based on direct manipulation, using touch inputs that loosely correspond to real-world actions, like swiping, tapping, pinching, and reverse pinching to manipulate on-screen objects, along with a virtual keyboard.

4.5.4 Web3j

web3j is a lightweight, highly modular, reactive, type safe Java and Android library for working with Smart Contracts and integrating with clients (nodes) on the Ethereum network.This allows you to work with the Ethereum blockchain, without the additional overhead of having to write your own integration code for the platform. It has complete implementation of JSON-RPC client (Application Programming Interface)API over Hypre Text Transfer Protocol(HTTP) and Inter Process Communication(IPC).It has android compatible version and supports Infura.It has complete support to ethereum wallet.

4.5.5 Solidity

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

4.5.6 ReactJS

It is a JavaScript library for building user interfaces. It is maintained by Facebook and a community of individual developers and companies. React can be used as a base in the development of single-page or mobile applications. Complex React applications usually require the use of additional libraries for state management, routing, and interaction with an API. It makes developer painless to create interactive UI. Properties commonly called props are passed from parent component to child. It also has feature of stateful components which can be passed to child components.React creates an in-memory data structure for virtual Document Object Model(DOM) and updates the browser DOM efficiently. Lifecycle in ReactJS are hooks which allows execution of code at set of points during component's lifetime.

4.5.7 SQLite

SQLite is an embedded SQL database engine. Unlike most other Structured Query Language(SQL) databases, SQLite does not have a separate server process. SQLite reads and writes directly to ordinary disk files. A complete SQL database with multiple tables, indices, triggers, and views, is contained in a single disk file. The database file format is cross-platform - we can freely copy a database between 32-bit and 64-bit systems or between big-endian and little-endian architectures. These features make SQLite a popular choice as an Application File Format. [8]

4.5.8 Rinkeby and Mist Browser

Rinkeby is a test network for deploying ethereum based smart contracts. As ethers hold real world value, this testnet is used as a test ether. Mist is a powerful Ethereum special-purpose browser. It offers like a overall view of the Ethereum blockchain and all needed tools to interact with the blockchain component like Ether, DAO, smart contracts. It is the indispensable tool for running or managing blockchain-specific DApps for the average user who doesn't need to understand technical aspects neither run Command Line Interface.

5. RESULTS AND ANALYSIS

5.1 Client Application

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

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

5.2 Node Application

5.3 Result of Node App

The node interface communicates with the :

- The interface shows how much storage space the files have taken, number of files the node has received and how many files in total clients have downloaded. List of all the chunks received as file can be watched in this panel. The list shows hash, creation and expiry date along with the number of download count for individual files. Events like receiving files are shown in real time using Websockets.
- This interface shows information about a deployed node with its public key, node id, private key. Other nodes connected or discovered by this node is shown with their node Identity, IP address and port.

6. CONCLUSION

The project consists of a P2P network where a node can join the network and provide the storage services for the client. The system uses several Cryptography and Network algorithms and provides two major services to client : Secure File Storage and Secret Sharing. The agreement between the parties are bound by Smart Contract and uses ERC20 token as a value for service. The two layers of the system can also be implemented separately as components for different use cases. The project was completed with a exciting exploration and research in Cryptography and Blockchain field. There is always room for the improvements in any projects. The project can be further enhanced including following features:

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

7. LIMITATIONS

The limitations of our system are listed below:

- There may be security concerns in this system. During development, pitfalls and loopholes in security have not been concerned in details.
- Currently there is size limit for file to store in network. This limit the user to store small files only.
- Cryptocurriencies are not legal all over the world where as our system needs those cryptocurrencies to work. This reduces our market size.

REFERNCES

[1] M. Crosby, P. Pattanayak, S. Verma, and V. Kalyanaraman, "Blockchain technology: Beyond bitcoin," *Applied Innovation*, vol. 2, pp. 6–10, 2016.

[2] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008.

[3] coldfustion, "Why blockchain matters more than you think!" Sep 2017. [Online]. Available: https://www.youtube.com/watch?v=sDNN0uH2Z3o

[4] "The great chain of being sure about things," Oct 2015. [Online]. Available: https://www.economist.com/news/briefing/21677228-technology-behind-bitcoin-lets-people-who-do-not-know-or-trust-each-other-build-dependable

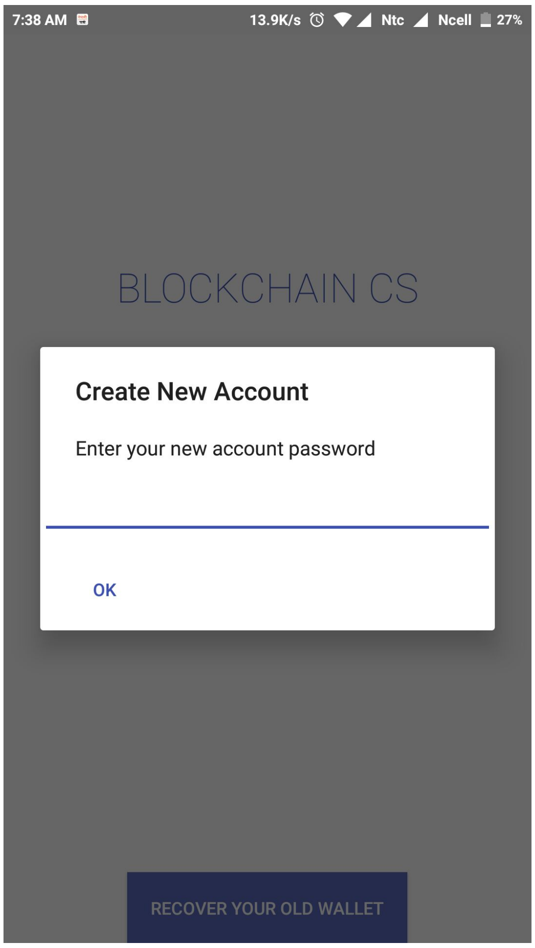
[5] T. McConaghy, R. Marques, A. Müller, D. De Jonghe, T. McConaghy, G. McMullen, R. Henderson, S. Bellemare, and A. Granzotto, "Bigchaindb: a scalable blockchain database," *white paper, BigChainDB*, 2016.

[6] S. Wilkinson, T. Boshevski, J. Brandoff, and V. Buterin, "Storj a peer-to-peer cloud storage network," 2014.

[7] S. Jung, "Filecoin v. sia, storj & maidsafe: The crowded push for decentralized storage," Aug 2017. [Online]. Available: https://medium.com/tokenreport/filecoin-v-sia-storj-maidsafe-the-crowded-push-for-decentralized-storage-7157eb5060c9

[8] [Online]. Available: https://www.sqlite.org/about.html

APPENDIX A: CLIENT APPLICATION SCREENSHOTS

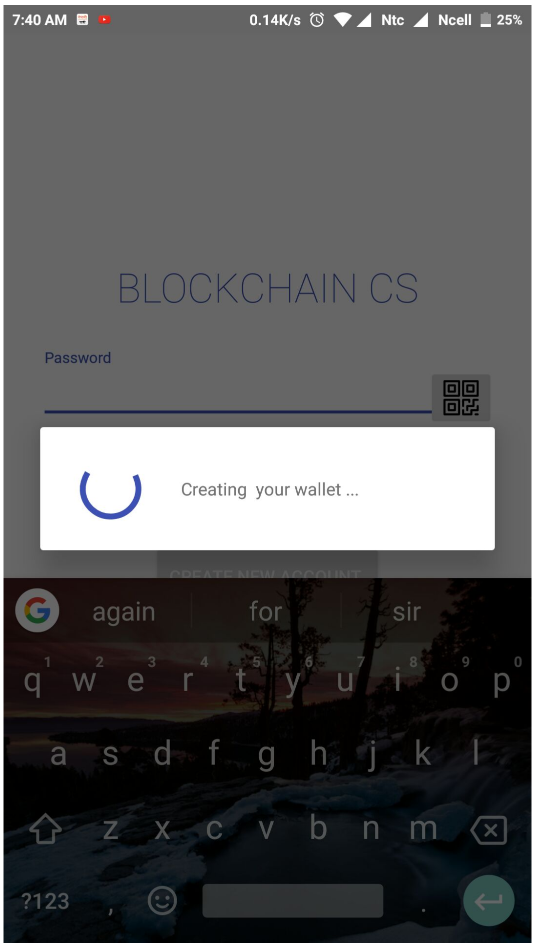


Create New Account

Enter your new account password

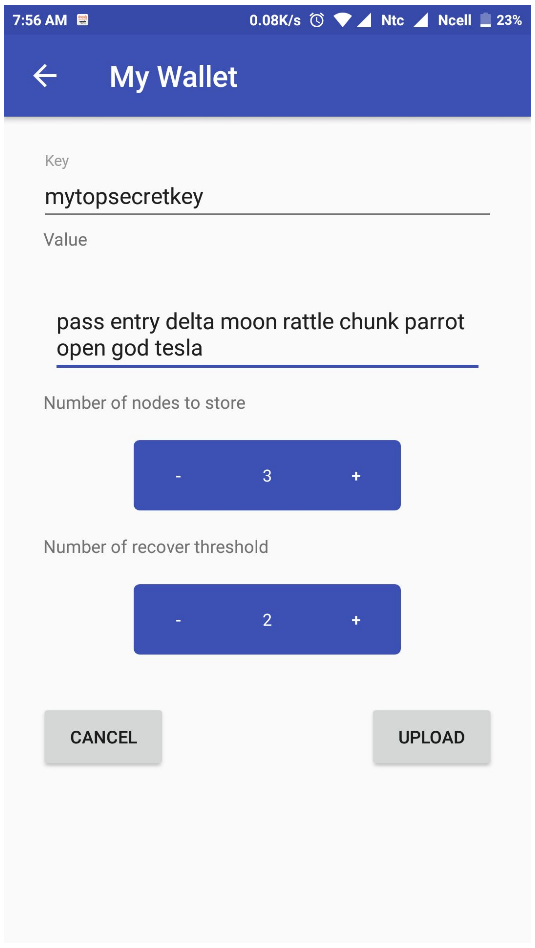
OK

Fig: Account Creation



Creating your wallet ...

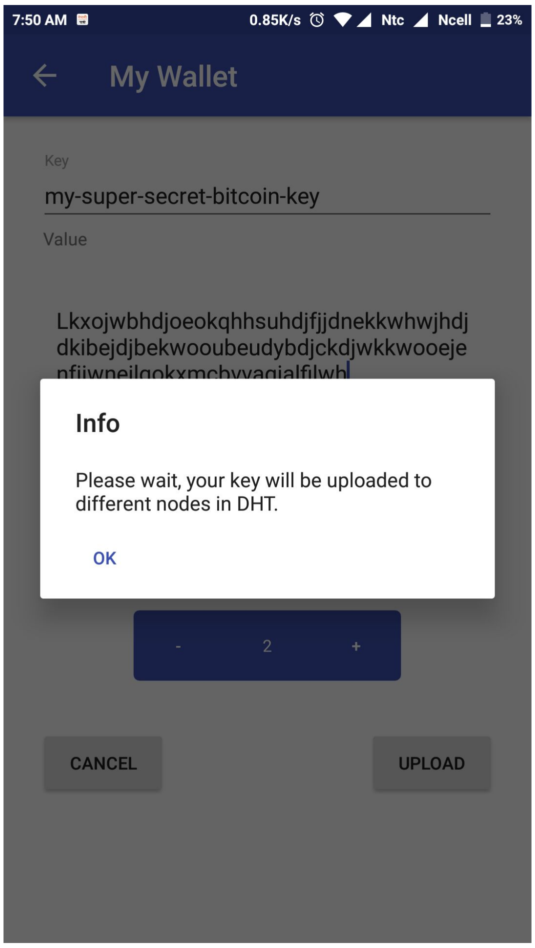
Fig: Wallet Generation



mytopsecretkey

pass entry delta moon rattle chunk parrot open god tesla

Fig: Samir Secret Sharing Process



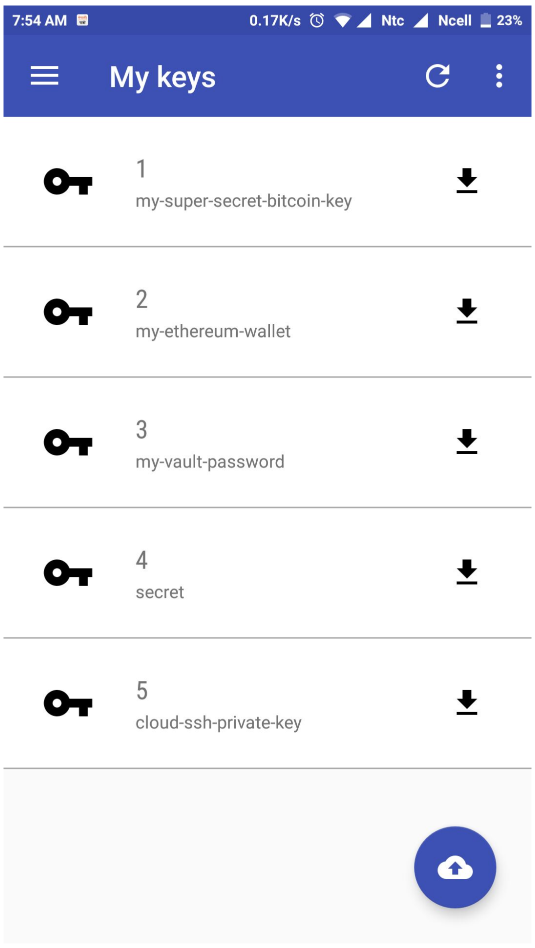
my-super-secret-bitcoin-key

Lkxojwbhdjoeokqhhsuhdjfjjdnekkwhwjhdj dkibejdjbekwooubeudybdjckdjwkkwooeje nfiiwneilqokxmchvvaqialfilwh

Info

Please wait, your key will be uploaded to different nodes in DHT.

Fig: Uploading Samir Secret


7:54 AM 0.17K/s Ntc Ncell 23%

My keys

1
my-super-secret-bitcoin-key

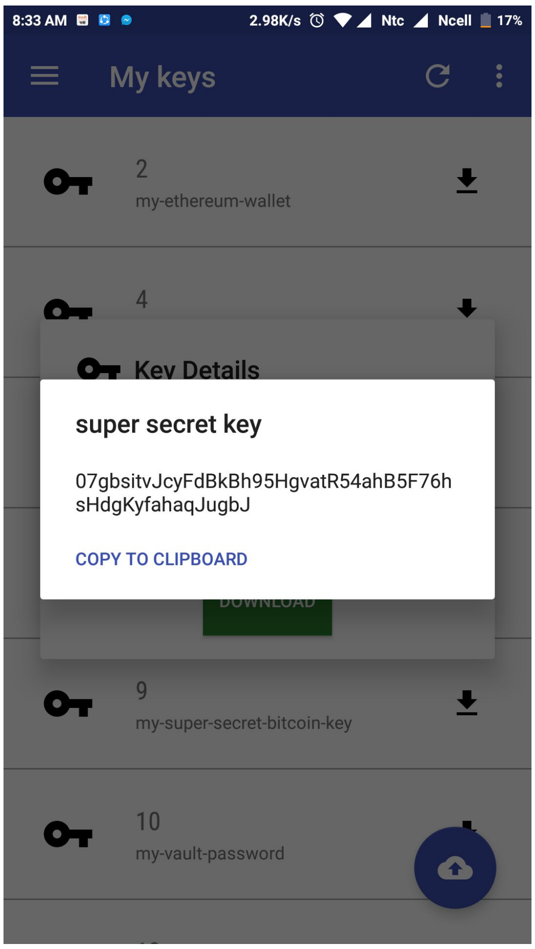
2
my-ethereum-wallet

3
my-vault-password

4
secret

5
cloud-ssh-private-key

Fig: Samir Secret’s List


8:33 AM 2.98K/s Ntc Ncell 17%

My keys

2
my-ethereum-wallet

4

Key Details

super secret key

07gbsitvJcyFdBkBh95HgvatR54ahB5F76h
sHdgKyfahaqJugbJ

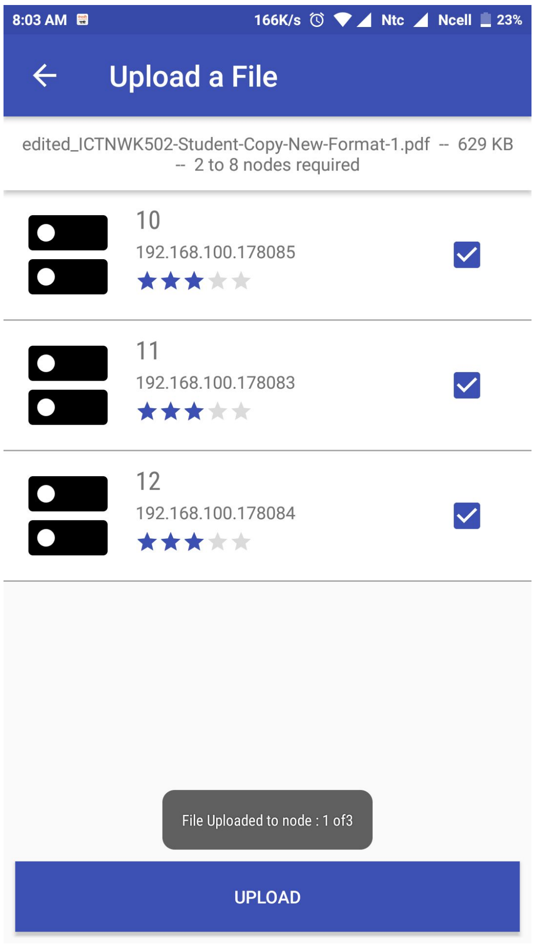
COPY TO CLIPBOARD

DOWNLOAD

9
my-super-secret-bitcoin-key

10
my-vault-password

Fig: Secret Detail



8:03 AM 166K/s Ntc Ncell 23%

Upload a File

edited_ICTNWK502-Student-Copy-New-Format-1.pdf -- 629 KB -- 2 to 8 nodes required

10
192.168.100.178085

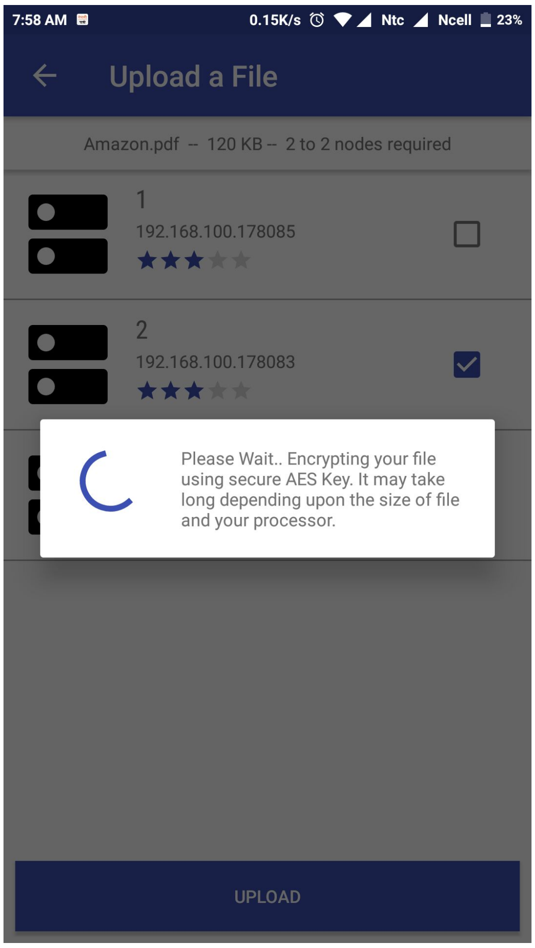
11
192.168.100.178083

12
192.168.100.178084

File Uploaded to node : 1 of3

UPLOAD

Fig: Uploading File



7:58 AM 0.15K/s Ntc Ncell 23%

Upload a File

Amazon.pdf -- 120 KB -- 2 to 2 nodes required

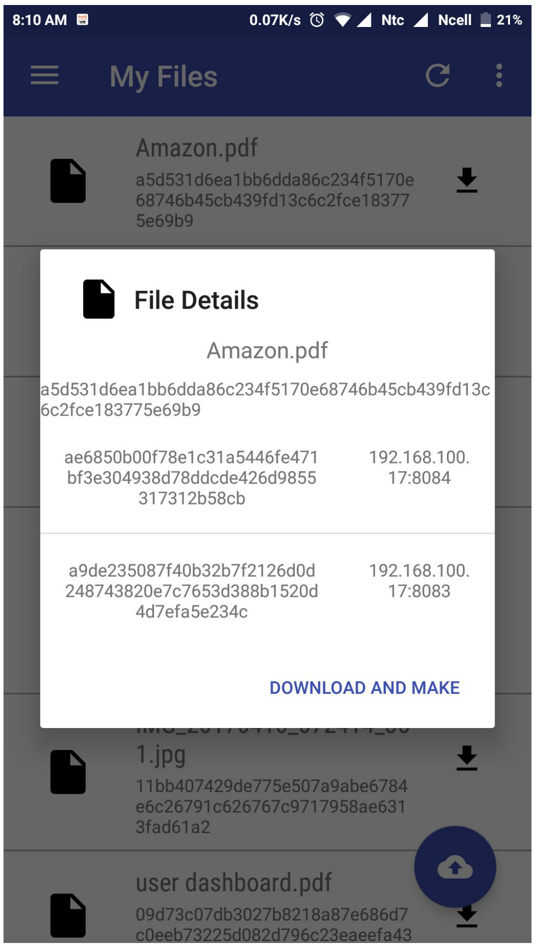
1
192.168.100.178085

2
192.168.100.178083

Please Wait.. Encrypting your file using secure AES Key. It may take long depending upon the size of file and your processor.

UPLOAD

Fig: Encrypting File


8:10 AM 0.07K/s Ntc Ncell 21%

My Files

Amazon.pdf
a5d531d6ea1bb6dda86c234f5170e68746b45cb439fd13c6c2fce183775e69b9

File Details

Amazon.pdf

a5d531d6ea1bb6dda86c234f5170e68746b45cb439fd13c6c2fce183775e69b9

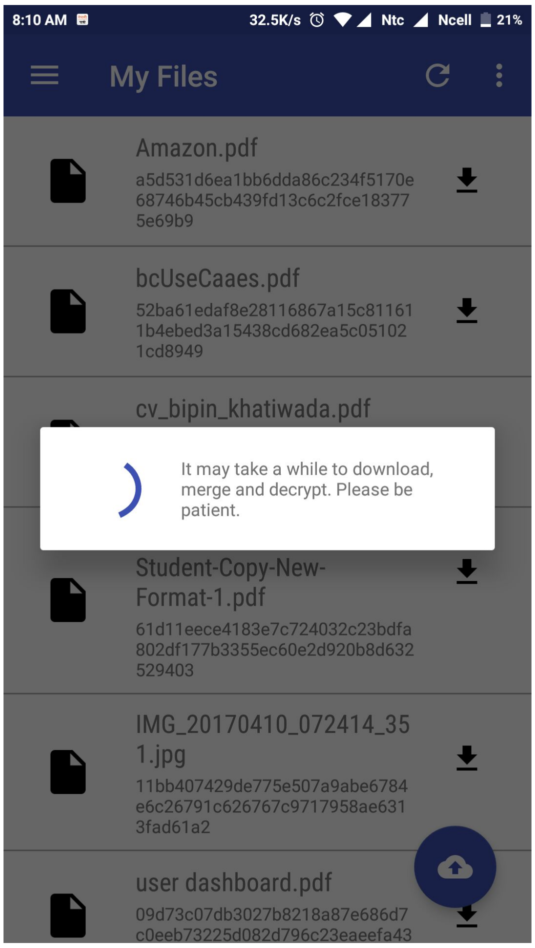
ae6850b00f78e1c31a5446fe471bf3e304938d78ddcde426d9855317312b58cb	192.168.100.17:8084
a9de235087f40b32b7f2126d0d248743820e7c7653d388b1520d4d7efa5e234c	192.168.100.17:8083

DOWNLOAD AND MAKE

1.jpg
11bb407429de775e507a9abe6784e6c26791c626767c9717958ae6313fad61a2

user dashboard.pdf
09d73c07db3027b8218a87e686d7c0eeb73225d082d796c23eaeefa43

Fig: File Details


8:10 AM 32.5K/s Ntc Ncell 21%

My Files

Amazon.pdf
a5d531d6ea1bb6dda86c234f5170e68746b45cb439fd13c6c2fce183775e69b9

bcUseCaaes.pdf
52ba61edaf8e28116867a15c811611b4ebed3a15438cd682ea5c051021cd8949

cv_bipin_khatiwada.pdf

It may take a while to download, merge and decrypt. Please be patient.

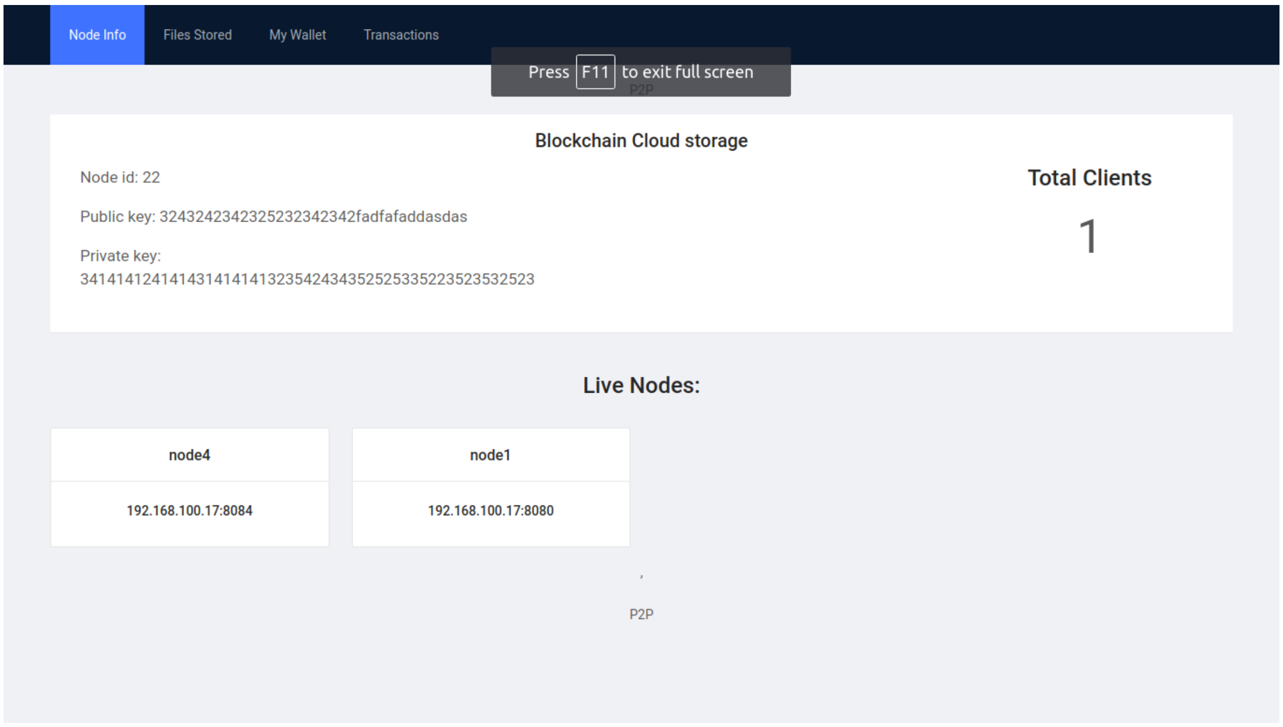
Student-Copy-New-Format-1.pdf
61d11eece4183e7c724032c23bdfa802df177b3355ec60e2d920b8d632529403

IMG_20170410_072414_351.jpg
11bb407429de775e507a9abe6784e6c26791c626767c9717958ae6313fad61a2

user dashboard.pdf
09d73c07db3027b8218a87e686d7c0eeb73225d082d796c23eaeefa43

Fig: File Download

APPENDIX B: NODE APPLICATION SCREENSHOTS



Blockchain Cloud storage

Node id: 22

Public key: 32432423423252323423 42fadfafaddasdas

Private key:
3414141241414314141413235424343525253352235235 32523

Total Clients

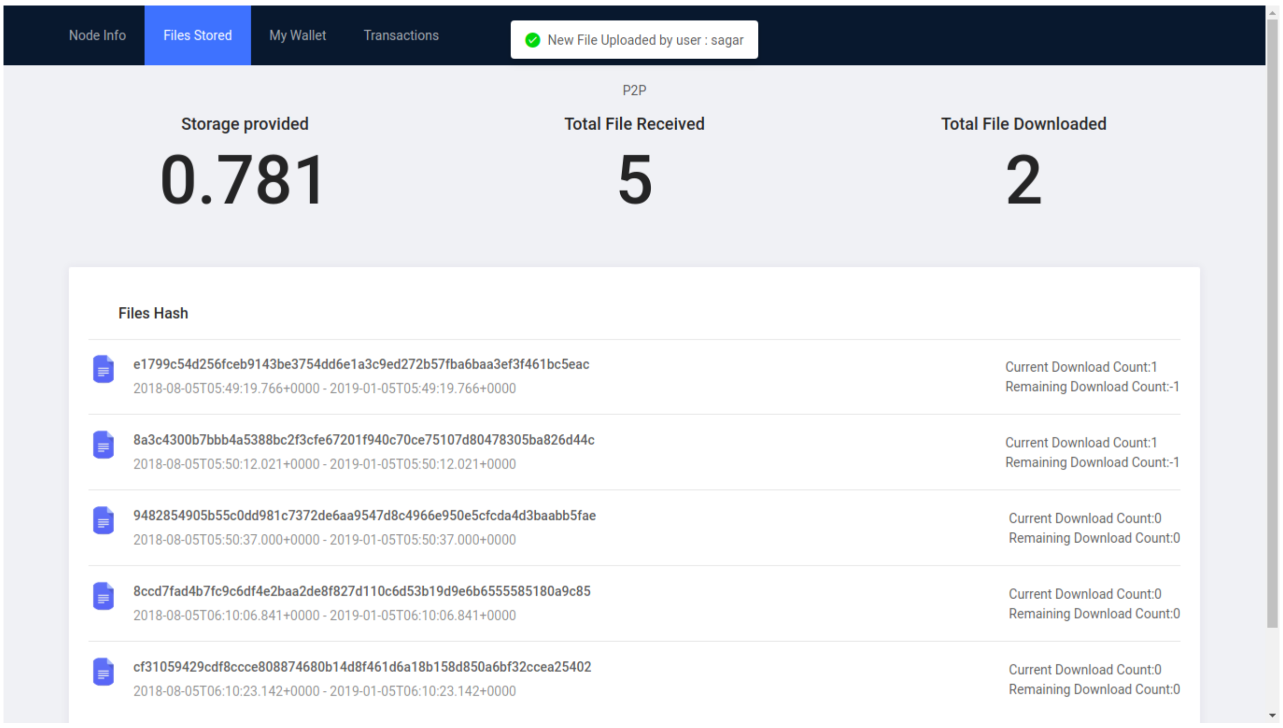
1

Live Nodes:

node4	node1
192.168.100.17:8084	192.168.100.17:8080

P2P

Fig: Node Connections



New File Uploaded by user : sagar

P2P

Storage provided
0.781

Total File Received
5

Total File Downloaded
2

Files Hash

e1799c54d256fceb9143be3754dd6e1a3c9ed272b57fba6baa3ef3f461bc5eac
2018-08-05T05:49:19.766+0000 - 2019-01-05T05:49:19.766+0000
Current Download Count:1
Remaining Download Count:-1

8a3c4300b7bbb4a5388bc2f3cfe67201f940c70ce75107d80478305ba826d44c
2018-08-05T05:50:12.021+0000 - 2019-01-05T05:50:12.021+0000
Current Download Count:1
Remaining Download Count:-1

9482854905b55c0dd981c7372de6aa9547d8c4966e950e5cfcda4d3baabb5fae
2018-08-05T05:50:37.000+0000 - 2019-01-05T05:50:37.000+0000
Current Download Count:0
Remaining Download Count:0

8ccd7fad4b7fc9c6df4e2baa2de8f827d110c6d53b19d9e6b6555585180a9c85
2018-08-05T06:10:06.841+0000 - 2019-01-05T06:10:06.841+0000
Current Download Count:0
Remaining Download Count:0

cf31059429cdf8ccce808874680b14d8f461d6a18b158d850a6bf32ccea25402
2018-08-05T06:10:23.142+0000 - 2019-01-05T06:10:23.142+0000
Current Download Count:0
Remaining Download Count:0

Fig: Node’s Data Info