Decentralized Cloud Storage using Blockchain

Progress Report

COMP4801 Final Year Project

Name: Shivam Tiwari, 3035239624
Supervisor: Dr. John Yuen

14th April, 2019
Abstract

The project aims to create a decentralized cloud storage application based on a peer-to-peer networking system. It aims to remove any reliance of users on third-party storage providers, which will in turn increase privacy and security of their data. The application aims to provide the functionality of searchable encryption over the cloud storage. The data will be stored on the basis of a renter-farmer situation wherein the renter’s will upload the user’s data to the client-side UI and the data will then be broken down into multiple fragments which will then be stored by the farmers on their systems. The project is on the development phase, having built the client-UI and is currently working on searchable encryption.
Acknowledgements

I would like to thank Professor Patrick Leung for his constant and excellent guidance in writing this report, without his efforts this report would not have had been possible. I would also like to thank Dr. John Yuen, for his guidance in designing and developing this project. Lastly, I would like to thank my group mates Mr. Tejasvi Mehra and Mr. Aman Johar for the constant hours they have put in with me to make this project a reality.
# Table of Contents

List of Figures 6

List of Tables 7

Abbreviations 8

1. Introduction 9
   1.2 Previous works 12
   1.3 Objective 13
   1.4 Motivation 13
   1.5 Status of Project 15
   1.6 Outline of Report 16

2. Methodology 17
   2.1 Technologies used 17
      2.1.1 Ethereum Blockchain: 17
      2.1.2 Storj.io: 18
      2.1.3 Ganache-cli: 18
      2.1.4 Truffle: 19
      2.1.5 Solidity: 19
      2.1.6 Node.js: 19
2.2 Framework

2.2.1 Storage of data
2.2.2 Retrieval of data
2.2.3 Encryption of the data
2.2.4 Pay for Storage

2.3 Workflow

2.4 Features

2.4.1 Searchable Encryption

2.4.1.1 Workflow of Searchable Encryption

2.4.2 Secure Multiparty Computation

3. Results

4. Challenges faced

5. Limitations

6. Future Work

7. Conclusion

References
List of Figures

1. **Figure 1:** The process of a transaction being added to the blockchain

2. **Figure 2:** The following figure shows the differences between traditional and a decentralized downloading of files

3. **Figure 3:** Homepage of App

4. **Figure 4:** Login Page of App

5. **Figure 5:** Register page of App

6. **Figure 6:** Showcasing how to select a file to encrypt

7. **Figure 7:** Showcasing how to enter tags and a passphrase

8. **Figure 8:** Showcasing the whole of the upload and encryption page

9. **Figure 9:** File retrieval page of App

10. **Figure 10:** How to input passphrase for decryption
List of Tables

1. **Table 1**: Index table for all documents to be mapped with their keywords. Pg. 33

2. **Table 2**: Encrypted File with File ID
   Pg. 33

3. **Table 3**: Encrypted keyword mapped with file ID. Pg. 34
Abbreviations

API: Application Programming Interface
ASP: Active Server Pages
GB: Giga Byte
HTTP: Hypertext Transfer Protocol
IPFS: InterPlanetary File System
kB: Kilobyte
P2P: Peer-to-peer
PHP: Hypertext Preprocessor, originally Personal Home Page
RSA Algorithm: Rivest-Shamir-Adleman Algorithm
1. Introduction

1.1 Background

Blockchain technology represents a shift in the ergonomics of storing data and building applications. As time is progressing, people are finding new and better ways of implementing the technology. One such application of blockchain is a decentralized cloud storage. The main reason blockchain technology is so popular, especially with financial markets, is that it provides a safe and secure alternative[1] for storing information, exchanging currency and validating transactions without the need of a third-party vendor or centralized organization.

Figure 1. The process of a transaction being added to the blockchain [1]
Blockchain is defined as a distributed ledger that records transactions between two entities and stores it permanently[1]. Thus it is a continuous growing list of records. Figure 1 shows the process of a transaction being added to the blockchain. This process is mostly used in exchanging cryptocurrency using blockchain. The reason for its popularity is that all the transactions stored in the ledger are immutable and the details of transaction are stored with cryptographic hashing so only the owner of the information can access said information[1]. On the other hand in conventional ledgers and storage systems all the information is stored with a central authority which has access to all the information stored by an individual on their systems and is theoretically open to more attacks [6] by hackers. Another reason for the popularity of blockchain is that a transaction is only validated when majority of the users on the chain agree that the transaction is true, thus taking away the authority from a central figure which can take decisions based on personal agendas and biases.

Figure 2. The following figure shows the differences between traditional and a decentralized downloading of files[2].
A decentralized cloud storage works in a similar fashion. Although the concept of decentralized systems has been around for quite a while, the application of decentralization specifically for cloud storage has recently been focused on more with the advent of blockchain technology[2]. A very easy way to understand decentralized storage would be to take the example of torrents. BitTorrent works on the principle of decentralized storage where every downloader (node) of a particular file (torrent) will eventually re-upload the data to a server as shown in Figure 2. This allows people to download fragments of a file from multiple sources thus reducing the load on a particular server and increasing efficiency in terms of time and speed[2] (see Figure 2 which shows the benefits of a Peer-to-Peer downloading system over a centralized downloading system). A decentralized cloud storage is quite similar to BitTorrent but with a much more secure system[2]. The nodes are incentivized to mine (download) files, which are then stored as fragments of the whole file on their systems. The files stored are encrypted and only the owner of the whole file knows the location of individual fragments of the file.

Cryptography, another major component in the project, is the practice and study of techniques for securing data for storage and communication in the presence of hackers, crackers and other adversaries. The use of secure hash algorithms like SHA-1, SHA-256 and other modern cryptographic algorithms like Advanced Encryption Standard or AES and Data Encryption Standard or DES help the project to protect data of the users from malicious users by encrypting the data. The report talks about two main algorithms used in the project SHA - 1 and AES and their usage in the project in later sections.
1.2 Previous works

The blockchain technology and decentralized cloud storage has been used by various firms like IPFS and Storj.io to add an competitive edge to the provision of storage by companies. There has been some research on searchable encryption but it continues to be a source of data leaks in some cases. The aim of this project is to build a searchable encryption that will in the eradicate data leaks and provide a safe and secure method of searching encrypted documents.

1.2.1 IPFS

InterPlanetary File System or IPFS is a peer-to-peer distributed file system that works aims to replace the current transfer protocol, HTTP, with itself. IPFS lets users download information from multiple sources at the same time, instead of downloading from a single source. This makes it enormously efficient in terms of and downloading speed and cutting costs for bandwidth. IPFS uses cryptocurrency to incentivize cloud storage using a payment channel called FileCoin.

1.2.1 Storj

Storj is a framework for decentralized cloud storage. It is a robust system that encrypts, shards and distributes the data across the all nodes on the network for storage. It provides a secure data storage system, which has a high performance guarantee while offering storage space at a fraction of the costs of traditional data storage providers.
1.3 Objective

This project aims to design and develop a system based on the concept of decentralized cloud storage using blockchain and incentivizes the nodes to store information and uses searchable encryption to provide a major feature over existing technologies. The project will build upon APIs from Storj.io and IPFS to build a decentralized cloud storage. To accomplish the objective for this project the major programming will go into building a highly secure and compatible searchable encryption, explained in later sections. As compared to traditional cloud storages like Google Drive, iCloud, Dropbox and data storage centers of data-centric companies like Facebook, which are privy to substantial amounts of data and have been known to have multiple data leaks over the years[6], such a system would provide a much safer and faster alternative. The functionality of searchable encryption will give this project and edge over other decentralized cloud storage systems.

1.4 Motivation

The main reason for a decentralized cloud storage system is the fact that it allows users to share and store data without relying on a third party provider. The project aims to mitigate data losses and increase protection of data by implementing client side encryption. Over the past few years cloud storage prices have skyrocketed. Traditional cloud storage companies like Dropbox charge users around $1 per GB of storage, whereas Storj gives 100 times[3] more storage at the same price. Moreover, given the frequent data leaks from data storage centers in recent times from data storage giants like Facebook and Dropbox [6], trust from third party storage providers is
waning. This gives rise to a need of cheap and trustworthy source of cloud storage. This is the primary reason for the development of our project.

1.4.1 Security and Data Redundancy

Data stored in a blockchain network is secured using cryptographic techniques. After the encryption, data is broken down and distributed amongst various nodes in the network, creating various redundancies. Even if one node is compromised, data lost is of no significant use to the hacker as it is highly encrypted and contains a very miniscule portion of the original data. Redundancies allow the data to be stored in multiple locations, hence even if data is lost from a single node, it can be recreated from some other node.

1.4.2 Cost efficiency

Development of a distributed cloud storage system will drive down infrastructure costs. Single party data storage providers need to acquire land and build data centers, maintenance and security personnel, buy expensive machinery and yet should be able to scale to a larger scale every other month. A distributed cloud storage system will completely remove much of these infrastructure costs and alleviate the burden of maintenance costs as it stores the encrypted data on a personal device of a person on the network. This also reduces the ability of traditional companies to completely monopolize the market of storing data and allows the user to be the provider as well.
1.5 Status of Project

The project has been completed by the team. Starting from sourcing the APIs from the Storj.io and IPFS and studying them the team has build a platform that is much better and is a faster alternative to already existing technologies. In the future, the application hopes to have added functionalities of multiparty computation and cryptocurrency based incentives for farmers giving their drives for storage. The application now has the added functionalities of searchable encryption to augment the decentralized cloud storage system that the team has built. The team uses AES and SHA-1 to implement searchable encryption, which will be explained in later parts of the report. Added functionality of a cryptocurrency would be developed by the team to provide unique payment system for the cloud storage and until then Ether (a cryptocurrency developed on Ethereum) would be used for payments to the farmers. A unique graphical user interface has been developed for the upholders or the renters to upload their files and pay for the space used per byte. Future developments will include a cli (command line version) for programmers to directly access the network and upload and download files from their terminal or command prompt. The final deliverable is a streamlined version of the very heavy systems like Storj.io and IPFS and as an added bonus provides a few added functionalities.
1.6 Outline of Report

This report aims to familiarize the reader with technologies that are going to be used in the development of StoreBlock. It outlines a brief procedure for the reader to understand how development will proceed for StoreBlock and the difficulties that are being faced and will be faced in the future during this development. The report is organized into 5 main sections including introduction. Section 2 will look at the methodology and the development procedure of the project and will discuss the main functionality, searchable encryption, section 3 will talk about the results of the final deliverable followed by the challenges faced in the development. Section 5 will talk about the limitations of the current deliverable, and how the team will work to mitigate these problems and future work will be detailed in section 6 before finally concluding the report in section 7.
2. Methodology

This section discusses the technologies used in this project in detail. It lays out the approach to the project and the application framework as well.

2.1 Technologies used

2.1.1 Ethereum Blockchain:

Ethereum is an open source, public, blockchain based distributed computing technology developed by Vitalik Buterin based on a modified version of the Nakamoto consensus. [7]

**Justification:** The reason for using Ethereum Blockchain over other blockchains like Bitcoin is that Ethereum provides a facility called smart contracts, which are programs on the basis of which transactions take place. [7]The application looks to achieve its main functionalities using these smart contracts. One might argue that Hyperledger too provides such a functionality to code functionalities, but due to the fact that blockchain is a relatively new concept there is limited documentation on how to use different technologies. [7]Since Ethereum is the most popular technology in the market right now, the community has tried its best to document the use cases of smart contracts.
2.1.2 Storj.io:

Storj is an open source, decentralized file storage application. It offers cloud storage service to its customers using a blockchain based hash table to store on the node of a P2P network.

**Justification:** Storj is a very popular decentralized cloud storage system. The main advantage in using Storj over IPFS, FileCoin and other such systems is that Storj is open source. That means that using Storj’s APIs is much easier. Furthermore, since Storj is open source there have already been lot of modifications to Storj over the years by individual developer teams which would give this project an advantage on skipping over the failed areas of development.

2.1.3 Ganache-cli:

Ganache is a personal blockchain developed by truffle framework which can be used to deploy smart contracts, develop dApps (decentralized applications) and is also used for testing[8]. Ganache-cli is the command line version of the testing environment. It is used to imitate Ethereum Blockchain and provide an environment for development which is faster, easier and safer [8].

**Justification:** Ganache provides a unique feature of deploying private chains on the developer’s personal computer which can be used for testing[8]. Deploying an application on the Ethereum blockchain costs money and every operation requires gas (based on the complexity of an application, carrying out functions on the chain requires gas which in turn requires money)[7]. Since a decentralized cloud storage would require very complex functions it would be counter
intuitive to deploy an application that is still in its starting stages as every time something needs to be changed, the application would have to be deployed again which would turn out to be expensive. Using Ganache is free and gives the same functionality with more privacy while in the development and testing phases of an application.

2.1.4 Truffle:
Truffle is a development environment which helps in integrating the web/client side application with the smart contracts. Truffle is used to compile smart contracts, automate contract testing on a private chain, migrate contract information to and from the web, enables network management for deploying to any number of networks.

2.1.5 Solidity:
Solidity is a high level programming language derived from Python and Javascript and specifically developed to write smart contracts. It offers the best and most stable development environment to the developer. The StoreBlock would use Solidity v0.4.25 for its development.

2.1.6 Node.js:
Node.js is a Javascript runtime environment. It runs on a dedicated HTTP server and employs a single thread to run a single process. This makes the application asynchronous and event-based.

Justification: Node.js as compared to PHP, ASP.net and other server-side scripting languages is more lightweight, eliminates waiting time between multiple requests and executes them simultaneously and runs non-blocking programming which is memory efficient. As the
StoreBlock will in itself be very heavyweight, it is a good idea to use a server-side scripting language which is memory efficient and lightweight. The two benefits of Node.js in the proposed project trump all positives of PHP and other languages over Nodejs. Ethereum also supports Node.js and its modules, called node modules, which will help in accelerating the pace of the project.

2.1.7 IPFS:

As discussed in earlier sections the InterPlanetary File System or IPFS is a peer-to-peer distributed file system that aims to replace the current transfer protocol, HTTP, with itself.

**Justification:** IPFS is an open source API which can be utilized to develop and build a P2P system. It serves as a basic foundation for the application and enables the team to shard, split files, and store the shards on different nodes across the network. This mitigates the major problem of building a P2P network from scratch. IPFS was chosen over Storj and Filecoin because IPFS is stable and its documentation is complete, thus making it easy to use.
2.2 Framework

All the design implementations within our application do the following 4 things.

(i) Store the data
(ii) Retrieve the data
(iii) Encrypt the data
(iv) Pay for storage

The following subsection will be explain how the data is stored, retrieved, maintained and paid for.

2.2.1 Storage of data

The storage of data in the application is done with the use of storage nodes. These storage nodes is run by individuals and entities called node operators. These storage operators download the encrypted fragment of data on their local systems. As this data is a fragment of the whole data and is highly encrypted, there would be no security threat of the data getting leaked. There are many redundant fragments to ensure no data loss. Thus, when a data is uploaded, multiple storage nodes will be storing the same data fragments. On successfully downloading the data and keeping it in their file space, the node operators will earn income. On every successful retrieval, these node operators will be rewarded with money. The information on where the individual data fragments are stored will be with the owner of the data.
2.2.2 Retrieval of data

Retrieval of data is done by the owner of the data. Due to redundancy of data, there are multiple nodes from which a single fragment of data can be uploaded and downloaded from. Storage nodes are selected on the basis of their availability at the time, their bandwidth and upload speed, and their distance to the retriever. This ensures high quality and high speed retrieval of data for the user.

2.2.3 Encryption of the data

It is ensured by the application that as soon as the data is uploaded by the user, it is encrypted using the secure hashing algorithm AES. After the data is encrypted it is “sharded” into multiple fragments of size 1MB or less. These data shards will then be again encrypted using the Advanced Encryption Standard to further secure the encrypted data. Once the data is fragmented and encrypted it will be uploaded to the servers for node operators, selected on random by an algorithm, to download. The data encrypted fragments will have a public key available to be seen by the public but will only be opened by a private key which only the uploader of the data has. Once the data is retrieved by the owner, the fragments will be decrypted to the original encryption and then linked together to form the original document, which will then be decrypted.
2.2.4 Pay for Storage

The owner of a data will be required to pay a minimal amount, as compared to universal standards, for every byte of storage he is using. The payments of this system will rely on blockchain. Once a data is stored, an encrypted hash verifying that the data was uploaded by user A and was stored by users $u_1, u_2, u_3, \ldots, u_n$ will be stored as a transaction on the blockchain. Smart contracts on the blockchain will then be used to distribute the earnings between the downloaders. Thus, blockchain will pay a very vital role in separating the earnings and ensuring each node operator will get their earnings.
2.3 Workflow

1. Homepage

The homepage of the web application welcomes the user to a screen where the user has the option of creating a new account or logging in directly.

Figure 3. Homepage
2. *The user can either choose to login or register with application*

*I. Login*

If the chooses to login directly, he will have to specify his email and the password he used to register on our app.

*Figure 4. Login*
II. Register

If the user has not registered with the application before he can choose to register using his email and a password.

![Register Form]

*Figure 5. Register*
3. **Upload the file**

The user can either upload the file using the drag and drop method or can upload the file by choosing from his folder.

![Figure 6. Select file to encrypt](image)

4. **Specifying tags and providing a personal key as a passphrase for encryption**

The user will then need to provide a personal key to encrypt the file and tags to identify the file uniquely.

![Figure 7. Entering tags and a passphrase to encrypt the file.](image)
The upload and encryption page looks like the following

![Figure 8. Upload and Encryption page](image)

*Figure 8. Upload and Encryption page*
5. Retrieval of files

There are two steps to retrieve the file.

I. The user first has to search for using the tags that will uniquely identify the file or using the exact name of the file. The file name is stored with the original username of the data owner and thus no one else can use tags or filename to retrieve the file.

Figure 9. File retrieval page
The second step involves the user inputting the passphrase (the private key) to decrypt the file. The decrypted file is automatically downloaded by the server onto the user’s machine.

*Figure 10. Inputting passphrase for decryption*
2.4 Features

There are two major features in this application:

(i) Searchable Encryption (has been developed)

(ii) Multiparty Computation (is part of a future phase of development)

The following two subsections will discuss the two functionalities in brief.

2.4.1 Searchable Encryption

Searchable Encryption refers to a technique in cryptography which enables a user to search for a word or a sentence in a document while it is still encrypted. Originally, any document that is stored on the web would first have to be decrypted by the central server to allow searching in the document. Searchable encryption, on the other hand, takes into consideration that a word or a sentence must be contributing to the encryption made by an user. Thus, the application will encrypt the word on the user’s computer and search through every encrypted file of the user. This will,

(i) enhance security: As the server providing authority would not have to carry out decryption on the server side, it will be impossible for anyone except the owner to know the contents of the file.

(ii) save money: Decrypting and then searching thousand’s of document for a single word on the user’s side would require huge amounts bandwidth. Searchable encryption skips this step.

(iii) save time: Searching through all the files after decryption takes would require a lot of time and will potentially slow down the processor, whereas searching through just the encrypted file would reduce the time needed to decrypt a file.
2.4.1.1 Workflow of Searchable Encryption

The workflow consists of five stages: key generation, document pre-processing, encryption, search using the tokens and decryption. The details for each stage are given below. [7]

1. **Generation of keys**: Using symmetric key generation the encryption and decryption keys are developed. The owner of the data saves the key in a secure location as it is used to both encrypt the data when uploading and decrypt the data when downloading.

2. **Document Preprocessing**: To start encrypting the document preprocessing must be done, here the keywords are extracted from the document either from a data analysis algorithm or from standard inputs from the data owner. After taking all the keywords, an index of is built where the all the files matching a particular keyword are put in the row pertaining to the keyword. For example, let $F$ be the set of files to be uploaded. This step involves taking each file in $F$, assigning it a label and then creating an index table with each row having a keyword and all associated files. For example, consider 5 files in $F = F_1, F_2, F_3, F_4, F_5$. An index table is built as shown in Table 1.

<table>
<thead>
<tr>
<th>Keyword</th>
<th>File ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>$K_1$</td>
<td>$F_1, F_3, F_5$</td>
</tr>
<tr>
<td>$K_2$</td>
<td>$F_1, F_2$</td>
</tr>
</tbody>
</table>
3. **Encryption:** To encrypt the files using the encryption key used in the first step, the documents and the keys are encrypted using various encryption standards like AES and SHA-1. For a key \( k \), the encrypted keyword is stored as \( e(K) \) and the file is stored as \( e(F) \). For the above example, the encrypted document list is shown in Table 3 and encrypted keyword list in Table 4.

*Table 2 – Encrypted File with File ID*

<table>
<thead>
<tr>
<th>File</th>
<th>File ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>ENC(F_1)</td>
<td>1</td>
</tr>
<tr>
<td>ENC(F_2)</td>
<td>2</td>
</tr>
<tr>
<td>ENC(F_3)</td>
<td>3</td>
</tr>
<tr>
<td>Keyword</td>
<td>File ID</td>
</tr>
<tr>
<td>----------</td>
<td>------------</td>
</tr>
<tr>
<td>ENC(F₄)</td>
<td>4</td>
</tr>
<tr>
<td>ENC(F₅)</td>
<td>5</td>
</tr>
</tbody>
</table>

Table 3 – Encrypted keyword mapped with file ID

<table>
<thead>
<tr>
<th>Keyword</th>
<th>File ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>ENC(K₁)</td>
<td>D₁, D₃, D₅</td>
</tr>
<tr>
<td>ENC(K₂)</td>
<td>D₁, D₂</td>
</tr>
<tr>
<td>ENC(K₃)</td>
<td>D₁, D₂, D₃</td>
</tr>
<tr>
<td>ENC(K₄)</td>
<td>D₃</td>
</tr>
<tr>
<td>ENC(K₅)</td>
<td>D₃, D₅</td>
</tr>
<tr>
<td>ENC(K₆)</td>
<td>D₄</td>
</tr>
<tr>
<td>ENC(K₇)</td>
<td>D₄</td>
</tr>
<tr>
<td>ENC(K₈)</td>
<td>D₄, D₅</td>
</tr>
<tr>
<td>ENC(K₉)</td>
<td>D₉</td>
</tr>
</tbody>
</table>
4. **Search**: The backend server takes in the multiple search tokens provided by the user and creates a counter for each document matching a specified keyword. The document matching the most number of keywords is returned to the user.

5. **Decryption**: To decrypt the returned file, the data owner must use the symmetric key used to encrypt the file and the identifier keys in the first stage.
2.4.2 Secure Multiparty Computation

Secure multiparty computation refers to the ability of a program to compute complex cryptographic functions by multiple nodes instead of just one, thus increasing the efficiency.

In the application, when a user puts in a request for retrieval of a document, the server will start downloading the file from multiple nodes instead of just one. As there are many copies of a fragment spread across the network, the network will download different pieces of the same fragment from multiple nodes. This will increase the efficiency of the bandwidth and lessen the time taken to download a document.
3. Results

The development for the product has finished. The team set up a robust environment for the basic foundation on Node.js. This consists of a user registration and login service which is connected to a file upload system. The user can choose to upload a file with maximum size of 1MB onto our server. The information is stored in a MongoDB collection and the database is itself location on the cloud hosted by mlabs.com. After the file is uploaded, the file is encrypted by a script present on the client’s machine based on a “passphrase” entered by the user. This ensures that the sensitive information is not revealed to the server. There is also a way to retrieve the file based on its unique file id generated during the time the file is being uploaded. Once the file has been retrieved the user can use the same “passphrase” to decrypt and download the file. The script for this would be available only on the client side to ensure data security. With this, work on the UI/UX is near completion for the renters. Figure 5 shows the UI/UX currently developed. The UI/UX has been made using Bootstrap in collaboration with HTML and CSS.
4. Challenges faced

As there are very limited resources to blockchain and decentralized cloud storage, it was difficult to find a starting point to migrate the development made for the project. As the technology of blockchain is relatively new, it was very difficult to implement functionalities that would otherwise be relatively easy to code. Some anticipated difficulties were the development of P2P system and its integration with the final dApp. This was mitigated using the APIs and software made available by IPFS. Use of an existing API of the application Storj’s for development of a base system also proved to be difficult due to lack of documentation on how to use the APIs and thus it was not used but rather IPFS was used to implement the base.

The difficulties related to lack of resources continued to be a problem throughout the project but as the proficiency of the team in the subject grew over time, this difficulty was considerably alleviated. Once the team formed its own documentation on usage of APIs, the difficulties related to their usage were minimal. The development of a P2P system, in general, is very difficult and needs high computation power along with multiple networks for testing, and thus a decentralized system was achieved using the APIs of IPFS. The provision of special computers with high computation by the University of Hong Kong coupled with multiple networks for testing contributed a lot in reducing the difficulty of building such a system.
5. Limitations

There are some limitations to the present application, which will be taken care of in the next iteration of the project as detailed in the next section of future work. They are:

1. **Upload file size:** Even though the project implements file sharding, the file size being uploaded is limited due to front end code. This will be taken care of in future updates when retrieving files from IPFS is faster.

2. **Retrieval Speed:** The retrieval speed for IPFS depends on the node which is being created. If the file is not retrievable from the node, it creates another node to retrieve file. It may take upto 3 minutes to retrieve a file.

3. **Regular Support:** As IPFS and other technologies are being updated extremely fast the project also needs regular maintenance to keep up with the changes.

4. **Only Text Files:** Only text files may be uploaded to the system. In the future updates this feature will be extended to graphic and media files, till then such uploads give errors.
6. Future Work

The team plans to work on the project for the next few months to make it a market worthy product. The decentralized cloud storage is already pretty efficient but to make it compete against an already existing market, the team will add added functionalities of multi-party computation, to increase speed and natural language processing to identify what the text is about and suggest the optimum tags to the user. There is also the issue of IPFS taking a long time while to retrieve a file. The team is currently working on a solution to this and will launch the product to the market when ready. More modern UI/UX and support for a mobile application is also in the next phase of development for this project. The upload support will also be extended to graphic and media files in the future updates and upload of file size would also be increased. Regular support would be hired to keep the project up to date with the changing technologies.
7. Conclusion

Even though initially the progress of the project was slow, the team completed the project as per the schedule. The main requirements of the project were completed and the final project exceeds the team’s expectations. Although there are still a few challenges and limitations the team faces, the team is committed to making a product of the future that can be launched into the market space and lead innovation in the decentralized cloud storage arena.
References


8. [8] Truffle framework, Ganache: Read Me [Online], Available: 