Decentralized Cloud Storage using Blockchain

1/20/2019

Dr. John Yeun
Aman Johar - 3035253888
Shivam Tiwari - 3035239624
Tejasvi Mehra - 3035239600

The University of Hong Kong
Department of Computer Science
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 renters 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.
Acknowledgments

We 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. We would also like to thank Dr. John Yuen, for his guidance in designing and developing this project.
# Table of Contents

Abstract 2  
Acknowledgments 3  
List of Figures 5  
List of Tables 6  
Abbreviations 7  
1. Introduction 8  
  1.2 Previous works 10  
  1.3 Objective 10  
  1.4 Motivation 11  
  1.5 Status of Project 12  
  1.6 Outline of Report 12  
2. Methodology 13  
  2.1 Technologies used 13  
    2.1.1 Ethereum Blockchain: 13  
    2.1.2 Storj.io: 13  
    2.1.3 Ganache-cli: 14  
    2.1.4 Truffle: 14  
    2.1.5 Solidity: 14  
    2.1.6 Node.js: 15  
  2.2 Framework 15  
    2.2.1 Storage of data 15  
    2.2.2 Retrieval of data 16  
    2.2.3 Encryption of the data 16  
    2.2.4 Pay for Storage 16  
  2.3 Features 17  
    2.3.1 Searchable Encryption 17  
      2.3.1.1 Workflow of Searchable Encryption 18  
3. Progress and Future Planning 22  
4. Anticipated Difficulties and Potential 24  
5. Conclusion 25  
References 26
List of Figures

Figure 1 - The process of a transaction being added to the blockchain [1]........................... 8
Figure 2 - Differences between traditional and a decentralized downloading of files [2] .......... 9
Figure 3 - Working of Searchable Encryption ..........................................................................18
Figure 4 - Figure showcasing the application’s renters UI. ......................................................22
List of Tables

Table 1 - Table for mapping all keywords in a document .................................................................19
Table 2 - Inverted index table mapping keyword to all documents .........................................................19
Table 3 – Encrypted Document with Document ID ..............................................................................20
Table 4 – Encrypted keyword with Document ID ................................................................................21
Abbreviations

API: Application Programming Interface
ASP: Active Server Pages
GB: GigaByte
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 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.

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 the transaction are stored with cryptographic hashing so only the owner of the information can access said information [1]. On
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). 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.
1.2 Previous works

The blockchain technology and decentralized cloud storage have been used by various firms like IPFS and Storj.io to add a 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 searchable encryption that will 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 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 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 an advantage 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 an 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 for a 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 stolen is of no significant use to the hacker as it is highly encrypted and contains a very minuscule 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 APIs from the Storj.io and IPFS have already been studied and the team has started to make changes to the code to provide a much better and faster alternative to already existing technologies. In the future, the application hopes to have added functionalities of searchable encryption, multiparty computation and cryptocurrency-based incentives for farmers giving their drives for storage. The team has finished developing encryption and decryption part of uploaded documents using Advanced Encryption Standard (AES). Once the searchable encryption is completed it will be made compatible with the APIs from Storj.io to develop a decentralized cloud storage with searchable encryption. The cryptocurrency would be developed by the team to provide a 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 would be available to the uploaders or the renters to upload their files and pay for the space used per byte. The final deliverable hopes to be a streamlined version of the very heavy systems like Storj.io and IPFS but still provide added functionality.

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 the 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 current progress in the project and the next steps that the team will take in the development. Section 4 will talk about the limitations faced in the project and potential solutions before finally concluding the report in section 5.

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 based on 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, but since 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 a 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) [8]. 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.

2.2 Framework

All the design implementations within our application will 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 explained how the data will be stored, retrieved, maintained and paid for.

2.2.1 Storage of data

The storage of data in the application will be done with the use of storage nodes. These storage nodes will be run by individuals and entities called node operators. These storage operators will download the encrypted fragment of data on their local systems. As this data will be a fragment of the whole data and will be highly encrypted, there would be no security threat of the data getting leaked. There will be many redundant fragments to ensure no data loss. Thus, 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 will be done by the owner of the data. Due to the redundancy of data, there will be multiple nodes from which a single fragment of data can be uploaded and downloaded from. Storage nodes will be selected based on their availability at the time, their bandwidth and upload speed, and their distance to the retriever. This will ensure 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 will be “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 the blockchain. Once a data is stored, an encrypted hash verifying that the data was uploaded by user A and was stored by users u1, u2, u3, ...,un 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 Features

The major part of this application is searchable encryption. The following two subsections will explain the Searchable Encryption and discuss its working in detail.

2.3.1 Searchable Encryption

Searchable encryption is a technique in cryptography that enables searching for information in an encrypted document present in cloud storage which satisfying a prerequisite without decrypting the document to search. This involves three participants:

- **Data Owner**: The data owner is the original owner of information which is to be stored. The data owner is responsible for encrypting the data using an encryption key before uploading it to the cloud server.

- **Storage Provide**: This entity provides the cloud storage services to its clients. It sets up a cloud server which stores the encrypted data given by the data owner. It is also responsible for managing and searching for the data for a given search query by a user. The encrypted data which fulfills the search query is returned back to the user thus, it is also responsible for data retrieval. The provider must not be aware of the type of information it stores during its upload or retrieval.

- **Data User**: This entity uses the service provided by the storage provider to retrieve specific encrypted data through a search query. The data user and data owner may be same or different, however, the key used by data owner is needed to decrypt the retrieved information.

The data owner has n documents labeled as D1, D2,..., Dn that are to be stored on the cloud. The data owner encrypts these documents using his encryption key k into \( C = (C_1, C_2, ..., C_n) \) along with an Index I for the documents. The data owner sends \((C, I)\) to the server. The Data User, to search for a keyword \( w \), needs to generate a search token \( t_w \) and send it to the server as a search query. The server uses C, I and \( t_w \) to search for the encrypted text \( C_x \) containing the keywords.
given in the search token and returns it to the user. The data user then decrypts the text locally using the key k obtained from the data owner [9].

![Diagram of Searchable Encryption Workflow]

**2.3.1.1 Workflow of Searchable Encryption**

The workflow consists of six stages: key generation, document pre-processing, encryption, search token generation, search and decryption. The details for each stage are given below. [10]

1. **Key Generation:** It involves generating the encryption/decryption key using symmetric key generation. The same key is used by the data owner to encrypt the document and data user to decrypt.

2. **Document Pre-processing:** This stage involves all the functions to pre-processing the document. This is done to start the encryption process. Before encryption, the pre-processing is done to get the keywords present in each document and build an index. Let $D$ be the set of documents to be uploaded. This step involves taking each document in $D$, assigning it a label and then creating an index table with each row having a document and its keywords. For example, consider 5 documents in $D = D_1, D_2, D_3, D_4, D_5$. An index table is built as shown in Table 1.
Table 1 - Table for mapping all keywords in a document

<table>
<thead>
<tr>
<th>Document</th>
<th>Document ID</th>
<th>Keywords</th>
</tr>
</thead>
<tbody>
<tr>
<td>D₁</td>
<td>1</td>
<td>(W_1, W_2, W_3)</td>
</tr>
<tr>
<td>D₂</td>
<td>2</td>
<td>(W_2, W_3)</td>
</tr>
<tr>
<td>D₃</td>
<td>3</td>
<td>(W_1, W_3, W_4, W_5)</td>
</tr>
<tr>
<td>D₄</td>
<td>4</td>
<td>(W_6, W_7, W_8)</td>
</tr>
<tr>
<td>D₅</td>
<td>5</td>
<td>(W_1, W_5, W_8, W_9)</td>
</tr>
</tbody>
</table>

After creating an index table, an inverted index table is created mapping each keyword to all the documents it occurs in.

For the above example, an inverted index table is shown in Table 2

Table 2 - Inverted index table mapping keyword to all documents

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Document ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>(W_1)</td>
<td>1,3,5</td>
</tr>
<tr>
<td>(W_2)</td>
<td>1,2</td>
</tr>
<tr>
<td>(W_3)</td>
<td>1,2,3</td>
</tr>
<tr>
<td>(W_4)</td>
<td>3</td>
</tr>
<tr>
<td>(W_5)</td>
<td>3,5</td>
</tr>
<tr>
<td>(W_6)</td>
<td>4</td>
</tr>
<tr>
<td>(W_7)</td>
<td>4</td>
</tr>
<tr>
<td>(W_8)</td>
<td>4,5</td>
</tr>
<tr>
<td>(W_9)</td>
<td>9</td>
</tr>
</tbody>
</table>
3. **Encryption:** This stage is used to encrypt the documents and the index table. The encryption is done using the key generated during key generation stage. For a key $k$, the encrypted keyword is stored as $ENC(W_i)$ and the document is stored as $ENC(D_i)$. For the above example, the encrypted document list is shown in Table 3 and encrypted keyword list in Table 4.

*Table 3 – Encrypted Document with Document ID*

<table>
<thead>
<tr>
<th>Document</th>
<th>Document ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>ENC(D₁)</td>
<td>1</td>
</tr>
<tr>
<td>ENC(D₂)</td>
<td>2</td>
</tr>
<tr>
<td>ENC(D₃)</td>
<td>3</td>
</tr>
<tr>
<td>ENC(D₄)</td>
<td>4</td>
</tr>
<tr>
<td>ENC(D₅)</td>
<td>5</td>
</tr>
</tbody>
</table>
Table 4 – Encrypted keyword with Document ID

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Document ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>ENC(W₁)</td>
<td>1,3,5</td>
</tr>
<tr>
<td>ENC(W₂)</td>
<td>1,2</td>
</tr>
<tr>
<td>ENC(W₃)</td>
<td>1,2,3</td>
</tr>
<tr>
<td>ENC(W₄)</td>
<td>3</td>
</tr>
<tr>
<td>ENC(W₅)</td>
<td>3,5</td>
</tr>
<tr>
<td>ENC(W₆)</td>
<td>4</td>
</tr>
<tr>
<td>ENC(W₇)</td>
<td>4</td>
</tr>
<tr>
<td>ENC(W₈)</td>
<td>4,5</td>
</tr>
<tr>
<td>ENC(W₉)</td>
<td>9</td>
</tr>
</tbody>
</table>

4. **Search Token Generation**: This involves searching for a keyword in the encrypted documents stored. It contains the functions to create a search token from the data user to the server. The user inputs all the keywords it wants to search and then computes the search token \( t = (t₁, t₂, ..., tₙ) \) for \( n \) keywords. Once computed, the token is sent to the server in a search query.

5. **Search**: The server uses the search token and the encrypted keyword index table and searches for the encrypted keyword. When \( ENC(W_i) \) matches any keyword sent in search token \( t \), the document ID is returned. The document containing the most matches is returned to the data user.

6. **Decryption**: The data user now decrypts the document obtained by using the symmetric key generated in the key generation stage which is obtained from the data owner.
3. Progress and Future Planning

The development process is on track with the internal schedule of the team. Currently, the team has set up a basic environment on Nodejs. This consists of user registration and login service which is connected to a file upload system. The user can choose to upload a file with a maximum size of 1MB onto our server. The information is stored in a MongoDB collection and the database is itself located 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.

File fragmentation is the next step which would allow a user to upload a file of size greater than 1MB and break it into chunks of smaller size. The team has a solid idea on the process and
implementation of this would be done on the client size by making use of FileReader objects present in JQuery.

After file fragmentation, the team would develop Searchable Encryption onto the application as discussed above. Moreover, the team has familiarized itself with Solidity, the main development language to be used to develop the smart contracts used in this application. A lot of research has gone into finding the right blockchain for our application and to understand how to use cryptocurrencies as our payment option. With the completion of the Searchable Encryption, the team will migrate the functionality onto Ethereum Blockchain to ease the process of integrating it with the payment function to retrieve files by using searchable encryption.

Development of additional functionalities such as multi-party computation and an independent p2p network will take place in the final phase of development before the final deliverable is completed.
4. Anticipated Difficulties and Potential

As there are very limited resources to blockchain and decentralized cloud storage, it is difficult to find a starting point to migrate the development made for the project. As the technology of blockchain is relatively new, it is very difficult to implement functionalities that would otherwise be relatively easy to code. Some anticipated difficulties are the development of P2P system and its integration with the final dApp. Use of an existing API of Storj for development of a base system has also proven to be difficult due to lack of documentation on how to use the APIs.

The difficulties related to lack of resources will continue to be a problem but as the proficiency of the team in the subject will continue to grow over the next couple of months, this difficulty will be considerably alleviated. Once the team forms its own documentation on usage of APIs, the difficulties related to their use will also be made minimal. The development of a P2P system, in general, is very difficult and needs high computation power along with multiple networks for testing. The provision of special computers with high computation by the University of Hong Kong coupled with multiple networks for testing has contributed a lot in reducing the difficulty of building such a system.
5. Conclusion

Blockchain and cloud storage might still both be in the stages of infancy, but they are growing at a rapid pace. This project aims to build a decentralized cloud storage application using blockchain. The application is being built to act as a proof of concept which would be able to build a strong case for the use of blockchain, decentralized networks and cloud storage together. The use case of decentralized networks and cloud storage hopes to revolutionize the cloud storage industry towards a safer and cheaper alternative to the traditional cloud storage systems. The progress of the project was slow with the development only starting in late October, but it has gained momentum and the development of some major components such as file upload, encryption and searchable encryption are almost over. In the coming few months, the team hopes to have completed most of the development and moved on to add major features like multi party computation before showcasing the final deliverable. There are a few hurdles that the project might face during its development, but due to the well laid out mitigation strategies, these would not lead to any setbacks. The team believes that they will deliver a product that will have the potential to change the cloud storage industry for the better.
References


