Decentralized Storage Network with Smart Contract Incentivisation

PROJECT REPORT
submitted towards the partial fulfillment of
the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

Submitted by

Anshul Shah
Nikhil Sheoran
Suraj Gupta

Under the guidance of :

Dr. Manoj Mishra and Dr. Sugata Gangopadhyay

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY, ROORKEE
ROORKEE- 247667 (INDIA)

April, 2018

Candidate's Declaration

We declare that the work presented in this project report with the title "**Decentralized Storage Network with Smart Contract Incentivisation**" towards the fulfillment of the requirement for the award of the degree of **Bachelor of Technology** in **Computer Science & Engineering** submitted in the **Dept. of Computer Science & Engineering**, **Indian Institute of Technology, Roorkee, India** is an authentic record of our own work carried out during the period **from August 2017 to April 2018** under the supervision of **Dr. Manoj Mishra**, Professor, Dept. of CSE, IIT Roorkee and **Dr. Sugata Gangopadhyay**, Associate Professor, Dept. of CSE, IIT Roorkee.

The content of this report has not been submitted by us for the award of any other degree of this or any other institute.

DATE :	SIGNED : ....
PLACE :	ANSHUL SHAH (EN. NO: 14114013)
DATE :	SIGNED : ....
PLACE :	NIKHIL SHEORAN (EN. NO: 14114031)
DATE :	SIGNED : ....
PLACE :	SURAJ GUPTA (EN. NO: 14114065)

Certificate

This is to certify that the statement made by the candidate is correct to the best of my knowledge and belief.

DATE : .... SIGNED : ....

Manoj Mishra
Professor
Dept. of CSE, IIT Roorkee

Certificate

This is to certify that the statement made by the candidate is correct to the best of my knowledge and belief.

DATE : ....

SIGNATURE : ....
(DR. MANOJ MISHRA)
PROFESSOR
DEPT. OF CSE, IIT ROORKEE

DATE : ....

SIGNATURE : ....
(DR. SUGATA GANGOPADHYAY)
ASSOCIATE PROFESSOR
DEPT. OF CSE, IIT ROORKEE

Acknowledgements

First and foremost, we would like to express our sincere gratitude towards our guides **Dr. Manoj Mishra**, Professor, Dept. of Computer Science and Engineering, IIT Roorkee and **Dr. Sugata Gangopadhyay**, Associate Professor, Dept. of Computer Science and Engineering, IIT Roorkee for their ideal guidance throughout the period. Their advices, insightful discussions and constructive criticisms certainly enhanced our knowledge and improved our skills. Their constant encouragement, support and motivation have always been key sources of strength for us to overcome all the difficult and struggling phases.
We would also like to thank **Department of Computer Science and Engineering, IIT Roorkee** for providing lab and other resources for this project, as well as for being a perfect catalyst to our growth and education over these past four years.
We also extend our gratitude to all our friends, for keeping us motivated and providing us with valuable insights through various interesting discussions.

Abstract

In our project, we present our work upon peer to peer data replication system. This system is set apart from conventional cloud storage solutions by effectively utilizing the public contract systems readily available through the use of blockchain technology. Centralized Storage networks suffer from a major drawback of the factor of trust on a central authority and the concerns of privacy. We propose a novel architecture which utilizes latest developments in the cryptographic primitives and the blockchain technology to structure a Decentralized Storage Networks. In this case, the nodes provide their active free storage in exchange of the incentive in form of cryptographic currency, thus creating a self-sustaining storage network that can be used by others. We will discuss various protocols for creating this system and analyze various scenarios possible on the system.

Contents

List of Figures

We dedicate our work to Computer Science and Engineering Department, our families and friends.

Chapter 1

Introduction

Recent technological developments in the field of blockchain technology have forced us to challenge our view of thinking internet as a bunch of concentrated centralized service providers. Various blockchain networks have proven the importance of the decentralized transaction ledgers. These decentralized networks are providing platform for building useful services without any central management or trusted parties. The applications of blockchain are bringing significant efficiency to financial transactions, supply chain systems and social networking among others [4].

In our work, we try to leverage the blockchain technology in the field of storage networks. We provide a technology which enables any computer system with free storage to participate in a decentralized storage network. The provider will get the incentive in the form of crypto currency on lending this storage. On the other hand, any client requiring free storage can get it from the system. All the information of available free storage with each provider and the information about each storage contract created between a client and a provider is stored on a decentralized ledger. In this way, we are able to create a self-sustaining storage network with minimal central authority.

1.1 Elementary Components

- **Client** - These are the nodes that come up on the system to get storage capacity. They store their data and pay for the services provided in the form of cryptocurrency.
- **Provider** - These are the nodes that have free storage capacity. They lend their storage capacity and earn incentive for this in form of cryptocurrency.
- **Oracle** - A system that provides the storage tokens to new providers on verification. Verification is done using 'Proof of Space' challenges.

- **Storage Token** - Protocol tokens that maintain the information about the current free storage of any provider. They are issued to a new provider after verification of the storage space by the oracle.
- **Storage Agreement** - When a client stores it data, the selected Provider node's storage tokens are bound in a smart contract.
- **Virtual Currency** - The cryptocurrency maintained on the blockchain. This used to transfer funds from client to provider for providing storage.

1.2 Architecture Overview

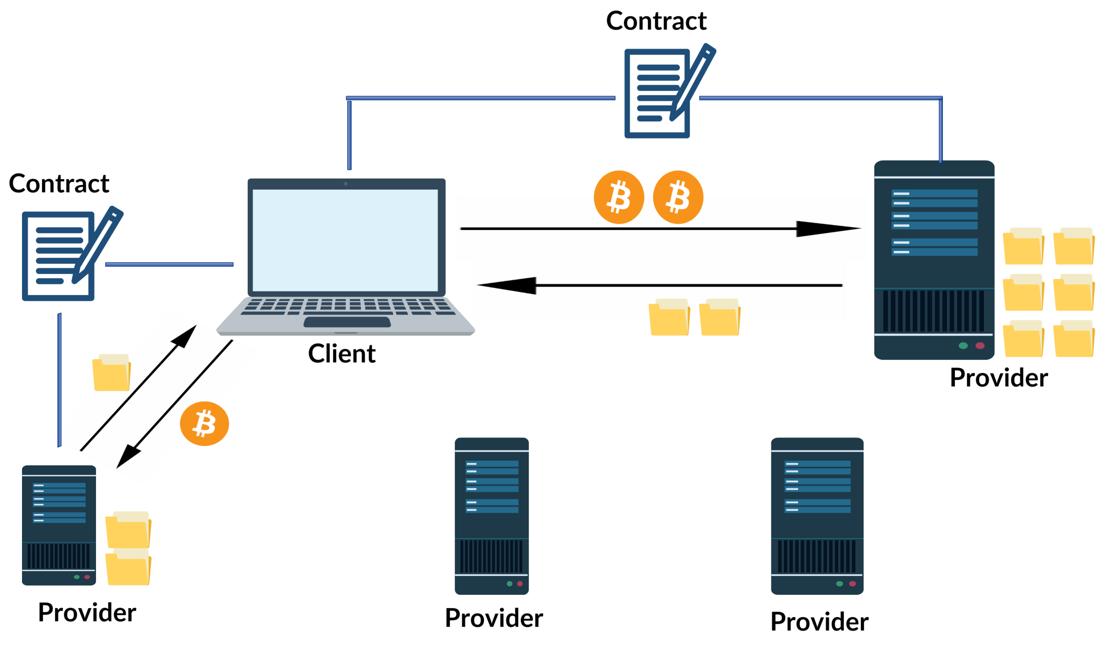

FIGURE 1.1: Decentralized Storage Network Architecture

- Whenever any new provider comes up on the system, it registers itself to the oracle by solving the relevant challenges proving the availability of free storage. On registration, the provider is given the storage tokens.
- When a client joins the system with some data to be stored, a storage request is created and on acceptance by the provider, storage agreement is created between the provider and client and the data is transferred to the provider.
- Now, the client node can repeatedly verify the data and reinforce the agreement after a fixed duration of time. On successful verification, the payment is made to the provider node.

- The termination of storage agreement can either be client initiated or provider initiated. In case it is initiated by client, the data is sent to the client and storage agreement gets terminated. In case it is initiated by the provider, the agreement is updated to reflect a new available provider, automatically identified by the system. In both the cases, the storage tokens are returned to the provider.

1.3 Desired Properties

- The data transaction happens on a **peer to peer basis** between the clients and the providers. This is unlike the concept used in Permacoin, which replicates the desired file in the blockchain of all the participant nodes [2]. The individual providers may have a dynamic pricing depending on their storage supply and client demand. The client software then selects the most suitable provider based on their pricing.
- **Node Induction:** For any node to join the network, it needs an initial peer. Using various peer discovery methods, the new node could communicate with other peers on the network.
- **Oracle**: To prevent overload, a central Oracle needs to operate at minimum cost in terms computation cost and blockchain transaction costs. The oracle plays no role in the P2P transactions which occurs between any two nodes. The oracle publishes the smart contract on the blockchain and being the owner has the sole authority to issue the storage tokens to providers.
- For the system to be self-sustaining, there needs to be a trust system which needs to be developed between the clients and providers. This system can be produced with the help of the blockchain technology. A **smart contract** deployed on the blockchain maintains an incentive layer by ensuring the accounting tasks are performed without any threat from attackers. The blockchain provides an append only ledger with consensus among nodes to uphold the trust and mitigate adversarial threats.

Chapter 2

Related Work

Data storage has always been a radically evolving field related to Computer Science starting from the magnetic discs to the today's Cloud storage networks. Today, there are various **SaaS(Storage as a Service)** platforms like Google Cloud, Microsoft Azure, Dropbox, etc. are present. They provide the advantage of reduced infrastructure costs, flexibility of usage, negligible maintenance and ease of scalability. However, they have some of their issues in the form of lack of control, and privacy concerns. [1] provides a comprehensive study of the advantages and drawbacks of the various cloud storage techniques used conventionally. In our work, we try to use the existing concepts of the cloud storage but trying to make it decentralized utilising the blockchain technology.

2.1 Peer to Peer Systems

Decentralisation of storage network is pivoted in the concept of peer to peer interactions of autonomous nodes in a network. The basic requirements for creating this nature of system are the underlying communication protocols which handles information exchange between the nodes. For efficient communication, our system needs a virtual overlay network as implemented in a DHT. It ensures that the node can easily query for a piece of information/resource in the network, even if the resource is rare [6]. Use of DHT enables the nodes to store a set (key, value) pairs on the network, and any node can later retrieve the value associated with the key. The properties of fault tolerance, decentralization and scalability can be achieved by implementations of DHT algorithms like Kademlia, Chord and Pastry.

2.2 Blockchain Systems

The Bitcoin white paper of 2008 has led to the conceptualization of the blockchain. This introduction of the concept and technical challenges is attributed to anonymous person or group known as Satoshi Nakamoto. The biggest engineering problem which was solved by the currency was the double spending problem outdoing the necessity of a central authority or institution [9]. The solution was a publicly visible append-only ledger comprising of financial transactions. The ledger was enforced security by cryptographic computational proofs which guarantee safety until the majority of the participating nodes go rogue.

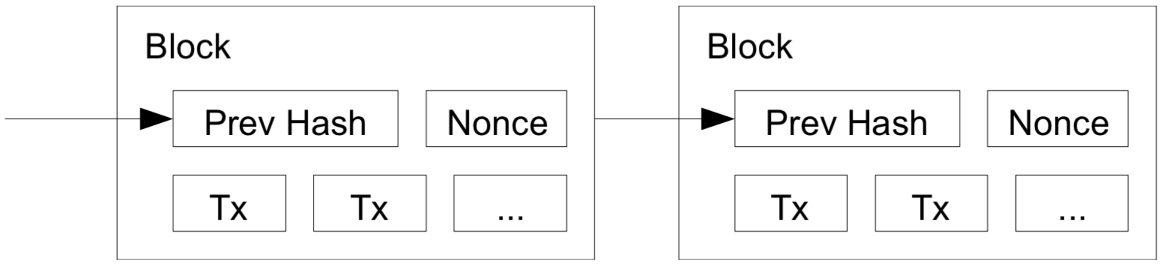

FIGURE 2.1: Blockchain with Blocks and Transactions

In the next phase of evolution, blockchain emerged as a platform for developing applications beyond the straightforward currency transactions. This was first made technically possible by the development of the Ethereum platform. It provides a turing complete set of instructions which are executed on a global network of nodes making it possible to create smart contracts - a piece of code which is run and verified on all the network nodes [3].It can be used for all kind of applications from legal processes to insurance premiums to crowdfunding agreements to financial derivatives. They can also be used for identity management and digital asset watermarking.

2.3 Decentralization of Storage

Apart from the other applications of blockchain, recent advancements in the field of decentralized storage network have become visible as well. There are ongoing efforts in the industry to create a decentralized storage network using the concepts of blockchain such as *storj.io* and *filecoin.io*. One of the major efforts, Filecoin [7], tries to create blockchain based on a novel concept of 'Proof of Spacetime', where blocks are created by miners that are storing data. This 'Proof of Spacetime' replaces the conventional 'Proof of Work' used in the blockchain system. It means that the miners need not spend wasteful computation for mining the new blocks, but they mine the blocks by storing data in the network. The native tokens of this blockchain is used for the transaction between clients and providers.

Working on similar line, we are trying to build an incentive layer based on any existing blockchain system and using their native tokens for the transactions. Also, we leverage the smart contracts to store the information about storage capacity and various storage agreements between the clients and the providers.

Chapter 3

Methodology

For our approach of creating a decentralized storage network, we divide our procedure into three major phases. The first phase deals with authorization of a new provider in the network. The second phase deals with the schemes associated with the interactions concerning storage agreements and data storage. The last phase describes the process of verification of remote data possession and payment mechanism.

3.1 Registering New Providers

3.1.1 Procedure

- A new provider requests to enter the system and claims availability of an amount S (say in bytes) of storage.
- The registration requires provider to give a Proof of Space[10], the availability of a non-trivial amount of disk space by solving a challenge given by the service provider. There are two phases : Initialization and Verification.
- In the initialization phase, the provider is given an initialization seed, say I, that lets him fill in the disk space with some computed data. This step will require availability of at-least S amount of storage. This data could be in the form a Merkle Hash-tree (upto a level T depending on size S)
- Once initialization has been completed, in the verification phase, the oracle starts issuing challenges where the user engages in the proofs using very small computation, given that he has the precomputed data stored with him.
- On successful solution of these challenges, the verifier issues storage tokens corresponding to an amount S of storage.

3.1.2 Desired Properties

- The challenge should be such that it requires availability of at-least S amount of simultaneous storage to be efficiently solved.
- The verification should have a low space and time footprint for the verifier.
- The storage requirement, computation complexity and communication complexity of the verifier during both the initialization and verification phases should be small particularly polylogarithmic or constant in storage requirement S.

3.1.3 Memory-Hard Puzzles

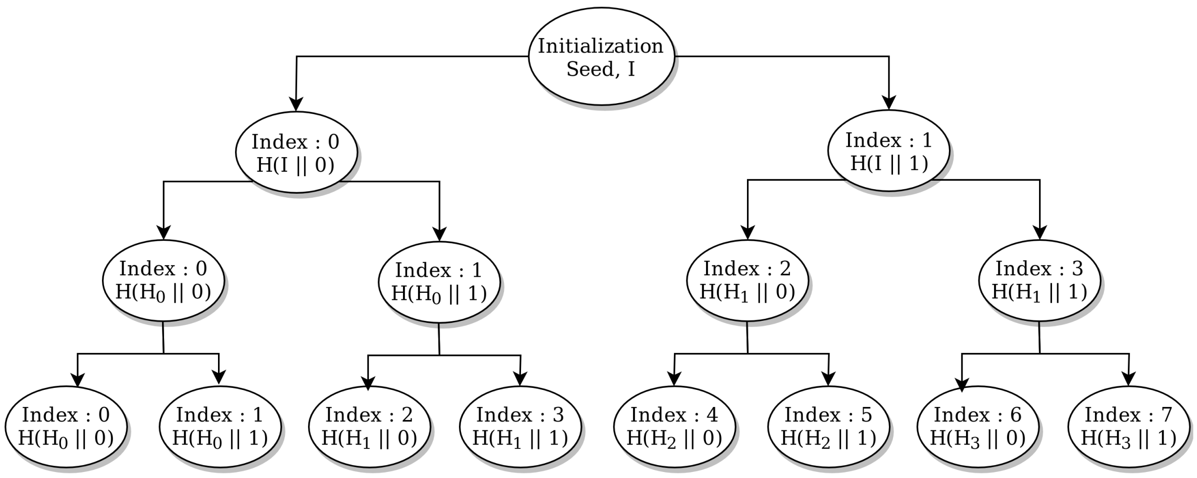

FIGURE 3.1: GGM Hash Tree for Hash values computation

- An example of a memory hard puzzle is GGM Hash tree. Starting with the random seed I obtained in the initialization phase, a binary tree of depth T is built by using the following scheme. If a node has label x, the label of its left child is $H(x||0)$ and label of its right child is $H(x||1)$, where H is a cryptographic function.
- So, basically each leaf is represented by a pair (i, x) where $i \in 0, 1^T$ is the index of the leaf and x is the label on that leaf. These are then sorted based on the x values.
- In the verification phase, the oracle challenges the provider by picking a random index i, computing the label x on that node and sending it to client, asking the client to send back the corresponding index. If the provider had stored all the 2^T records, it can quickly find the corresponding (i, x) and solve the challenge successfully. If it hasn't, it won't be able to solve the challenge efficiently.

3.2 Data Storage

Whenever a new client comes up on the system with some data to be stored, the list of prospective storage providers is discovered using the peer to peer node discovery mechanism. Once the client side gets the list of the possible provider candidates, a suitable candidate is selected and a storage request is sent to that particular provider.

3.2.1 Node Discovery

Our application allows peers to connect or disconnect from a network at any time. As a result, our application needs sophisticated discovery mechanisms to enable nodes to find, identify and communicate with other nodes. In our system, we are employing the oracle to function as a lightweight automated tracker system [8]. The tracker provides an initial seed list of peers to the nodes, which then allows any further discovery of peers through discovery mechanisms.

3.2.2 Creation of Storage Agreement

Algorithm 1 Data Storage Scheme

```
1: Client requests provider for creation of a Storage Agreement.
2: if $Provider.StorageTokens > RequestedTokens$ then
3:     $LastVerificationTime \leftarrow CurrentTime$
4:     $ExpiryTime \leftarrow CurrentTime$
5:     $Provider.Tokens \leftarrow Provider.Tokens - RequestedTokens$
6: Client uploads the data on the provider, and issues the initial challenge.
7: if $Provider.Solution == CorrectAnswer$ then
8:     $LastVerificationTime \leftarrow CurrentTime$
9:     $ExpiryTime \leftarrow CurrentTime + 24hrs$
10: if $CurrentTime > ExpiryTime$ then
11:     $Provider.Tokens \leftarrow Provider.Tokens + RequestedTokens$
12:     Storage Agreement is revoked.
```

The storage agreement is implemented using smart contracts. So, this whole information is stored on a distributed ledger and therefore not requiring any central information repository. This agreement acts as a binding between the client and the provider and discourages any malicious practice as shown in 3.2.

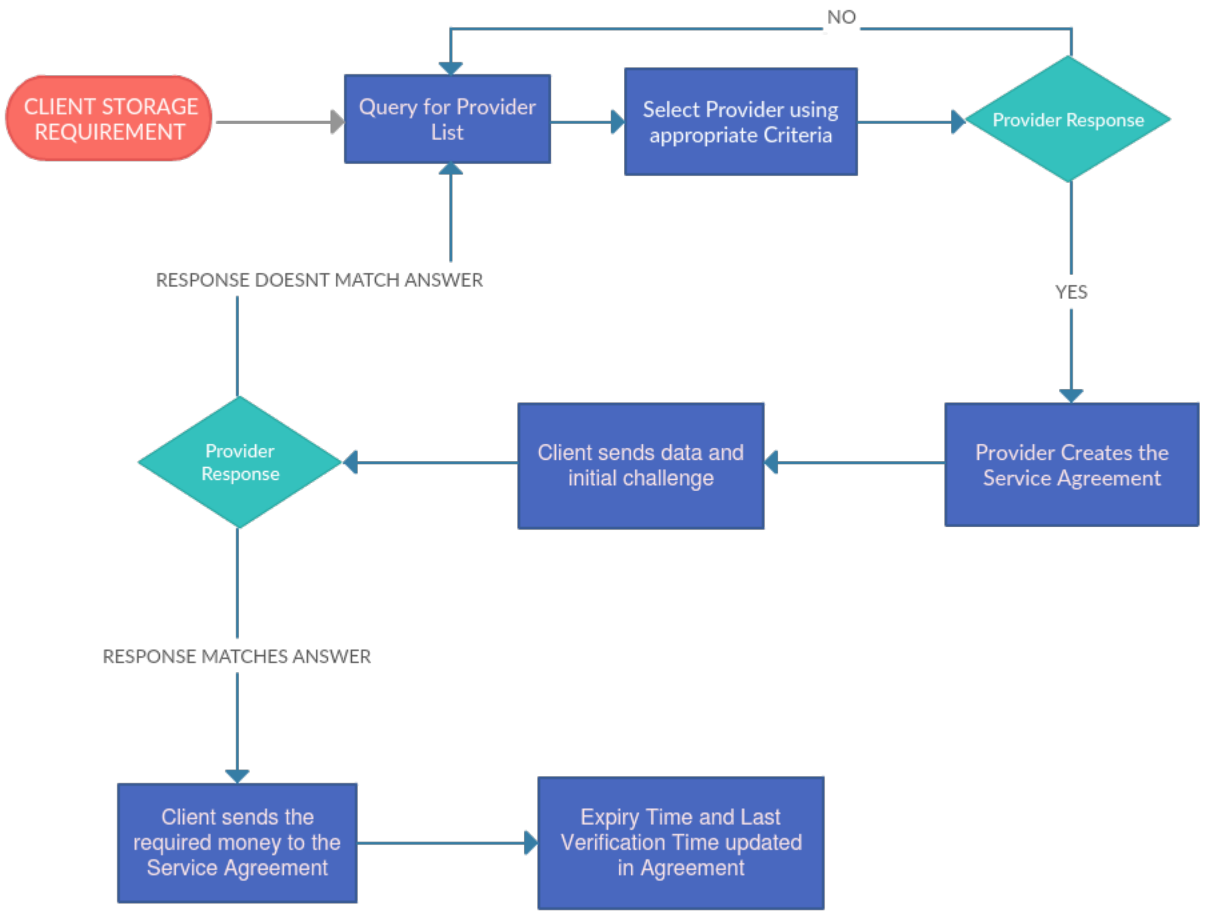
NO
CLIENT STORAGE REQUIREMENT
Query for Provider List
Select Provider using appropriate Criteria
Provider Response
RESPONSE DOESNT MATCH ANSWER
YES
Provider Response
Client sends data and initial challenge
Provider Creates the Service Agreement
RESPONSE MATCHES ANSWER
Client sends the required money to the Service Agreement
Expiry Time and Last Verification Time updated in Agreement

FIGURE 3.2: Flow Chart for the Storage Agreement

3.3 Data Verification and Retrieval

Once a storage agreement is in place between the client and the provider, the client can verify the presence of data with the provider at regular intervals. Data verification helps in ensuring that the provider doesn't drop the data mid way and enhances the reliability of the system. Whenever, the data verification process occurs, the storage agreement gets reinforced and the appropriate funds are sent from the client to the provider.

Data verification is enabled through 'Proof of Storage' mechanisms where the client sends a challenge to the provider which can be solved by the provider only if it has the actual data sent by the client.

The client can retrieve his data whenever required. But, in order to prevent excessive network congestion, a base amount would be levied after a minimum number of retrievals (say 1) per day.

3.3.1 Procedure

Algorithm 2 Data Verification

```
1: Client creates a $PoSt$ challenge.
2: Provider solves the challenge and sends the solution to the Client.
3: if $Success$ then
4:     Client sends desired money to contract.
5:     $LastVerificationTime \leftarrow CurrentTime$
6:     $ExpiryTime \leftarrow CurrentTime + 24hrs$
7:     Contract sends money to the provider.
8: else
9:     No updation of agreement occurs.
10:    Contract expires when $ExpiryTime == CurrentTime$
```

3.3.2 Desired Properties

- The challenge should be such that it can be solved only if the provider has the data.
- The verification should have a low space and time complexity for the verifier(the client).
- The challenge should be such that for a particular pair of client and provider, the verification can be done multiple times with minimal information sent each time over the network.

3.3.3 Proof of Storage Mechanism

Proof of storage challenges are such challenges which allow a client to frequently and efficiently verify that any system, that was to store the client's large amount of data is not cheating it. In the methodology we use for $ProofofStorage$ [5], we have two phases - $Setup$ and $Verification$ phase. In the setup phase, client initially sends some tokens along with the data. For each $verification$ phase, the $challenge$ is sent. Now, based on the $challenge$ and the data, the provider sends the answer to the challenge.

Notation

- D - the outsourced data
- OWN - the client who owns the data
- $SERV$ - the provider, i.e. the system that stores the data

- f - a pseudo-random function
- g - a pseudo-random permutation

Setup Phase

Assume we have a data D divided into d blocks. We want to be able to create total t challenges. Two keys W and Z are used to generate session permutation keys and challenge nonces respectively.

For the *Setup* phase, OWN creates t random challenges and the corresponding answers, known as **tokens**. For the i^{th} challenge, OWN generates a random permutation of r indices as follows:

Algorithm 3 Token Generation

```
1: Generate permutation key $k_i = f_W(i)$ and challenge nonce $c_i = f_Z(i)$
2: Compute the set of indices $\{ I_j | 1 \leq j \leq r \}$ where $I_j = g_{k_i}(j)$
3: Computer token $v_i = H\ ( c_i , D[I_1] , .. , D[I_r] )$
```

The token for each of the t challenges, v_i represents the solution of the challenge. The nonce c_i is needed to prevent any pre-computation by SRV.

Verification phase

- For the i^{th} verification, OWN regenerates k_i and c_i and sends it to SRV.
- SRV computes the permutation with the help of k_i and then generates v_i with the help of the data as $v_i = H\ (c_i , D[g_{k_i}(1)] , .. , D[g_{k_i}(r)])$
- SRV sends it to OWN as the answer to the given challenge.
- OWN verifies it and on success, OWN assumes that SRV still has the data D with a quite high probability.

Chapter 4

Implementation

There are numerous tools and technologies available for developing a decentralized application. We have identified and employed the most relevant tools for developing the infrastructure of our application. We have a web application with two interfaces, one for each provider and client, with options given specifically for each user. A Provider will have the options for setting the storage rates and selecting disk space. Whereas, a Client will select the upload data and the maximum cost he will spend recurringly. The Oracle is a background service which is responsible for issuing storage tokens to the providers.

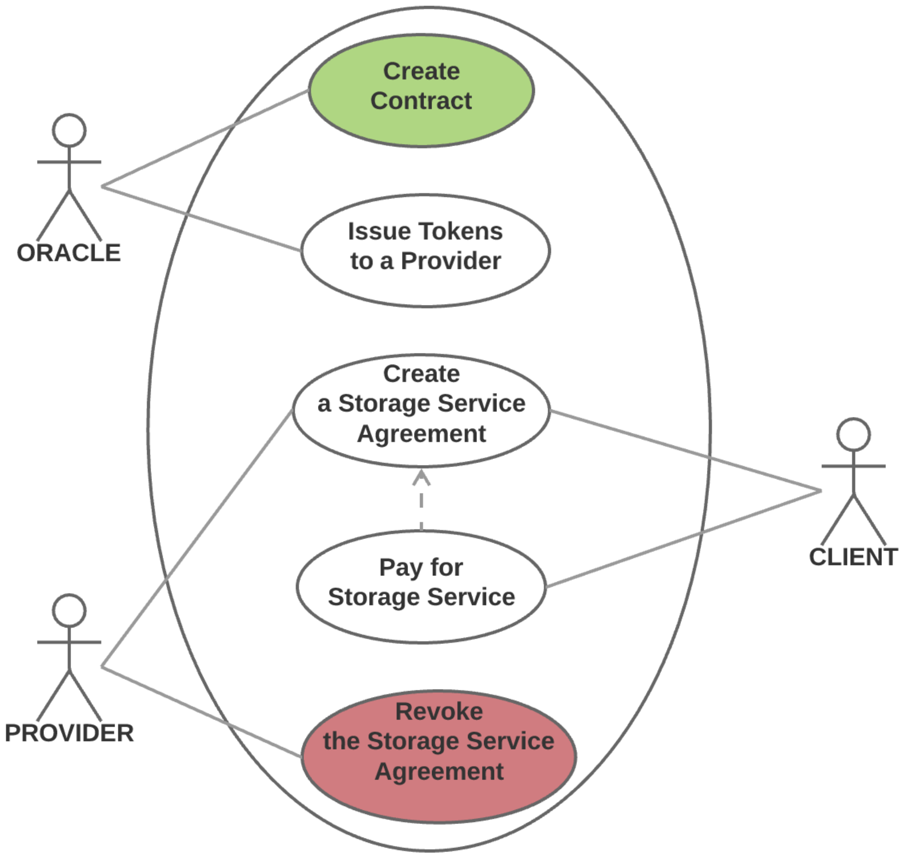
Create Contract
ORACLE
Issue Tokens to a Provider
Create a Storage Service Agreement
CLIENT
Pay for Storage Service
PROVIDER
Revoke the Storage Service Agreement

FIGURE 4.1: Use Case Diagram for the Storage Agreement

4.1 Tools and Technologies Used

- **Solidity** : Language used for writing the smart contracts.
- **Truffle** : Framework based on node which is used for testing and deploying the smart contracts.
- **Ganache (testrpc)** : Ethereum Blockchain Simulator
- **Flask (python)** : Used to create a lightweight oracle service
- **Web3 (python)** : Python library used to communicate with the Ethereum blockchain
- **SQLite** : In memory lightweight database. It is used in the oracle server for persisting data.

4.2 Implementation of Contracts

We have used the Ethereum blockchain as the incentive layer for our application. The smart contracts deployed on the blockchain will handle all the transactional use cases of our application. The class diagram for the smart contract is shown in 4.2. The Contract will be deployed only once by the oracle and its blockchain contract address as well as contract interface ABI (application binary interface) will be broadcasted publicly. The users which trust the contract code and the oracle will join the network using the information broadcasted. Depending on the role of each node, it will perform one of the several functions as shown in 4.1.

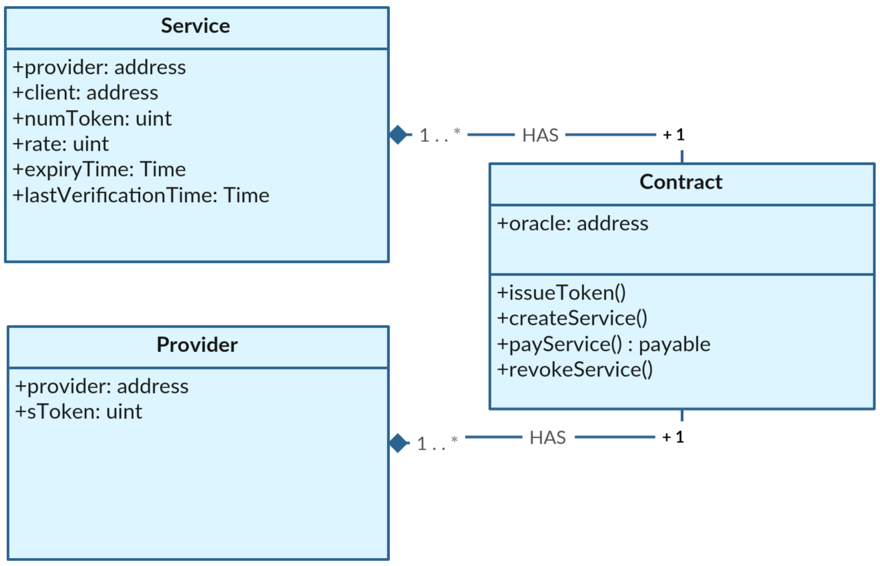

FIGURE 4.2: Class Diagram for the Contract

Chapter 5

Analysis

We analyzed and studied the robustness of our architecture with respect to various scenarios involving malicious clients and providers.

5.1 Malicious Provider Scenarios

These are some scenarios where the provider acts maliciously with intentions to harm the system. The protocols of our system ensures that it is robust and no client is harmed due to these malicious intentions.

5.1.1 Provider drops the data

- The provider may abruptly drop a client's data which was already bounded by a storage agreement before the $ExpiryTime$.
- In this case, the system ensures minimal replication by creating copies from the valid data available with other storage providers.
- Also, since the agreement hasn't been terminated correctly, the provider doesn't get back his storage tokens thus leaving him with lesser storage capacity.

5.1.2 Sybil Attack

- The provider may drop data from his existing contracts and rejoin as by creating a new identity. This is also known as a Sybil Attack.
- The computationally and time expensive Proof of Space along with the initial joining deposit prevents provider from unnecessarily rejoining as a new provider.

5.2 Malicious Client Scenarios

These are some scenarios where the client acts maliciously aiming to harm the system. The protocols of our system ensures that it is robust and no provider is harmed due to these malicious intentions.

5.2.1 Client evades the system without verifications

- The client may store the data once and does not come back to verify/retrieve the data thus leading to blockage of providers' storage capacity.
- In order to overcome this, the agreement has an expiry time and a buffer amount equivalent to this charge is kept as security in the contract.
- Once the agreement has expired, the agreement is terminated and provider gets back his storage tokens along with the security amount deposited.

5.2.2 Client repeatedly requests the data

- The client can repeatedly request his data creating congestion in the network.
- In order to prevent this, a daily retrieval limit is set beyond which the client is charged with a minimal amount for each retrieval.
- This ensures that the provider gets paid for the repeated network transfers while ensuring the client doesn't overflow the network.

Chapter 6

Conclusion

6.1 Comparison with traditional storage systems

The proposed decentralized storage network can be observed to have some advantages over the conventional storage systems.

- **Privacy Concerns** : In a decentralized storage network, there is no single party having all the data. Instead, it is distributed among the various nodes of the system. So, only the client has the full ownership of its data.
- **Decentralized** : As the system doesn't have any central control entity, there is no single point of failure.
- **Peer to peer** : As all the data tranfers are peer to peer, if leveraged, it can lead to marginally fast transfers.
- **Incentivized**: Anybody having free storage capacity can earn incentives for it. Moreover, it will bring down the price of storage.

6.2 Future Work

The current architecture can be made much more robust and efficient by incorporating some changes in the protocols and the architecture.

- **Optimal Provider Selection** : Algorithms can be devised for finding the optimal provider for a client based on network latency and other factors.
- **Profiling of Providers** : Historic profiles of providers can be created based on their previous records to facilitate the clients in optimal provider selection.
- **Publicly verifiable Proof of Storage** : Verifier nodes can be introduced in the system which perform data verification on behalf of the client based on publicily verifiable PoSt.

- **Distribution of Data** : We can have a mechanism to distribute the data into chunks and store it with various providers over the system. This ensures that only the client have full access to client's own data.
- **Transferable Storage Agreement** : Providing option of transferring the storage agreement to other providers.

In our project, we have explored the possibility of creating a decentralized storage network with the help of blockchain technology. While creating such a system, we have identified various challenges mainly preventing frauds arising from the clients and providers while preserving their anonymous identities. We employed various decentralized technological tools for solving this problem. If scaled efficiently, the idea of decentralized storage network can act as a storage mechanism for future.

Bibliography

[1] S. M. Bansode Amol S. Choure. “A Comprehensive Survey on Storage Techniques in Cloud Computing”. In: *International Journal of Computer Applications* (2015).

[2] Elaine Shi Bryan Parno Andrew Miller Ari Juels and Jonathan Katz. “Permacoin: Repurposing Bitcoin Work for Data Preservation”. In: (2014). URL: `https://ieeexplore.ieee.org/document/6956582/`.

[3] Vitalik Buterin. “Ethereum: A Next-Generation Smart Contract and Decentralized Application Platform”. In: (2013). URL: `https://github.com/ethereum/wiki/wiki/White-Paper`.

[4] Institute for Development and Research in Banking Technology. “Applications of Blockchain Technology to Banking and Financial Sector in India”. In: (2017). URL: `http://www.idrbt.ac.in/assets/publications/Best%20Practices/BCT.pdf`.

[5] Luigi V. Mancini Giuseppe Ateniese Roberto Di Pietro and Gene Tsudik. “Scalable and Efficient Provable Data Possession”. In: (2008). URL: `https://eprint.iacr.org/2008/114.pdf`.

[6] Yonggang Wen Hao Zhang and Haiyong Xie. “A Survey on Distributed Hash Table : Theory, Platforms, and Applications”. In: (2013). URL: `https://pdfs.semanticscholar.org/8859/3bf30b9de1497ba22b1136ebc433f648bbc8.pdf`.

[7] Protocol Labs. “Filecoin: A Decentralized Storage Network”. In: (2013). URL: `https://filecoin.io/filecoin.pdf`.

[8] P. Mehra D. Paul M. Kelaskar V. Matossian and M. Parashar. “A Study of Discovery Mechanisms for Peer-to-Peer Applications”. In: (2008). URL: `https://pdfs.semanticscholar.org/2f7b/e383a03bd7584b46d9e6d9cdf39fbe92bc20.pdf`.

[9] Satoshi Nakamoto. “Bitcoin: A Peer-to-Peer Electronic Cash System”. In: (2008). URL: `https://bitcoin.org/bitcoin.pdf`.

[10] Vladimir Kolmogorov Stefan Dziembowski Sebastian Faust and Krzysztof Pietrzak. “Proofs of Space”. In: (2013). URL: `https://eprint.iacr.org/2013/796.pdf`.