Electronics and Computer Science
Faculty of Physical Sciences and Engineering
University of Southampton

Thomas Smith, tcs1g20
December 5, 2022

Using Blockchain for Video Game Distribution

Project Supervisor: Leonardo Aniello
Second Examiner: tbd

A project report submitted for the award of
BSc Computer Science

Abstract

Video game developers will often have to rely on third party platforms for the distribution of their games; this comes at a large monetary cost to the developer and leaves users at a greater risk of censorship and with weak digital ownership that is reliant on the platform staying active. This project uses the Ethereum blockchain to facilitate the large-scale distribution and continuous updating of video games that allows developers to directly interact with their users, who will now have true digital ownership.

Statement of Originality

- I have read and understood the ECS Academic Integrity information and the University's Academic Integrity Guidance for Students.
- I am aware that failure to act in accordance with the Regulations Governing Academic Integrity may lead to the imposition of penalties which, for the most serious cases, may include termination of programme.
- I consent to the University copying and distributing any or all of my work in any form and using third parties (who may be based outside the EU/EEA) to verify whether my work contains plagiarised material, and for quality assurance purposes.

You must change the statements in the boxes if you do not agree with them.

We expect you to acknowledge all sources of information (e.g. ideas, algorithms, data) using citations. You must also put quotation marks around any sections of text that you have copied without paraphrasing. If any figures or tables have been taken or modified from another source, you must explain this in the caption and cite the original source.

I have acknowledged all sources, and identified any content taken from elsewhere.

If you have used any code (e.g. open-source code), reference designs, or similar resources that have been produced by anyone else, you must list them in the box below. In the report, you must explain what was used and how it relates to the work you have done.

I have not used any resources produced by anyone else.

You can consult with module teaching staff/demonstrators, but you should not show anyone else your work (this includes uploading your work to publicly-accessible repositories e.g. Github, unless expressly permitted by the module leader), or help them to do theirs. For individual assignments, we expect you to work on your own. For group assignments, we expect that you work only with your allocated group. You must get permission in writing from the module teaching staff before you seek outside assistance, e.g. a proofreading service, and declare it here.

I did all the work myself, or with my allocated group, and have not helped anyone else.

We expect that you have not fabricated, modified or distorted any data, evidence, references, experimental results, or other material used or presented in the report. You must clearly describe your experiments and how the results were obtained, and include all data, source code and/or designs (either in the report, or submitted as a separate file) so that your results could be reproduced.

The material in the report is genuine, and I have included all my data/-code/designs.

We expect that you have not previously submitted any part of this work for another assessment. You must get permission in writing from the module teaching staff before re-using any of your previously submitted work for this assessment.

I have not submitted any part of this work for another assessment.

If your work involved research/studies (including surveys) on human participants, their cells or data, or on animals, you must have been granted ethical approval before the work was carried out, and any experiments must have followed these requirements. You must give details of this in the report, and list the ethical approval reference number(s) in the box below.

My work did not involve human participants, their cells or data, or animals.

Acknowledgements

I would like to thank my supervisor, Leonardo Aniello, for his support throughout this project.

Contents

Chapter 1

Problem Statement

Video games are often large and highly popular pieces of software that are typically distributed for developers by a third party platform like Steam or Epic Games. Whilst these platforms provide benefits such as availability, and some social features they have some major downsides that includes:

(a) taking a large cut of all revenue,
Steam take a 30% cut
(b) being vulnerable to censorship from governments, and
The Chinese version of Steam is heavily censored
(c) the user's access to their games is linked to the platform.
If the platform shuts down, the user loses all their games

A blockchain-based platform will provide greater profits to the developer, eliminate the need for trust in a third party platform, and allow users greater control over the games they own as their access is not directly linked to one service.

1.1 Goals

The goal of this project is to implement a large-scale distribution platform that will allow game developers to release and continuously update their games on a public network by directly interacting with their users. This aims to boost revenue for the developer, reduce the risk of censorship, and improve the rights of the user in terms of digital ownership. The design should include:

- how data is shared between nodes in the network,
- how downloaded data can be verified using the network,
- how users can be incentivised by developers to help distributed their games,
- how users can prove their contribution,
- how users can prove they have purchased a game.

1.2 Scope

This project will look at how the Ethereum blockchain and smart contracts can be used to create a large-scale distribution platform for video games. This be deployed to a 'testnet', where Ether has no value and applications can be tested in a live environment. Whilst the focus on this application is on video games, this should be a valid solution for software of all types.

The application itself will consist of a set of smart contracts, written in Solidity, with tests and a basic UI to be written using TypeScript. Tools like Ganache and MetaMask may also be useful for development.

Chapter 2

Background Research

2.1 BitTorrent

BitTorrent [5, 10] is the most popular p2p file-sharing platform, in which users will barter for chunks of files by downloading and uploading them in a tit-for-tat fashion, such that peers with a high upload rate will typically also have a high download rate. For a user to download data from BitTorrent they would:

Download Protocol

1. Find the corresponding torrent file that contains metadata about the torrent such as the location of a tracker, file information such as name, size and path in the directory.
2. The user will find peers also interested in that torrent through a tracker and will establish connections with them.
3. The data is split into constant-sized blocks and are downloaded individually. BitTorrent uses a tit-for-tat mechanism that incentivises users to contribute by providing preferable treatment to nodes who upload data as well.
4. The user will download blocks based upon the following priority:
   (a) **Strict Priority** Data is split into pieces and sub-pieces with the aim that once a given sub-piece is requested then all of the other sub-pieces in the same piece are requested
   (b) **Rarest First** Aims to download the piece that the fewest peers have to increase supply.
   (c) **Random First Piece** When a peer has no pieces, it will try to get one as soon as possible to be able to contribute.
5. The node will continuously upload blocks it has while active.

Availability

It is commonly suggested that availability of torrents is the biggest issue surrounding BitTorrent as *'38% of torrents become unavailable in the first month'* [5] and that *'the majority of users disconnect from the network within a few hours after the download has finished'* [10]. This paper [9] looks at how the use of multiple trackers for the same content and DHTs can be used to boost availability.

2.2 Ethereum

Ethereum is a Turing-complete, distributed, transaction-based blockchain that allows the deployment of decentralized applications through the use of smart contracts. Ether is the currency used on Ethereum and can be traded between accounts and is used to execute smart contract code on the network.

Smart Contracts

A smart contract is an executable piece of code that is used to automate processes and enforce agreements between two or more parties. This code is then executed by every node on the ethereum network using the EVM.

Gas is a unit of measurement that is used to specify the computational effort required to execute operations on the Ethereum network. Each transaction must set a limit on the amount of gas can be used during code executing, and this is paid using ether. However, this can lead to smart contracts failing to execute due to *running out of gas*. By tying the computational effort of a smart contract to ether, the Ethereum network reduces the risk of DoS attacks as an attacker will likely not have the funds to perform such an attack.

Chapter 3

Literature Review

3.1 Blockchain-Based Cloud Storage

Blockchain technology can be leverage for large-scale, distributed cloud storage to allow data to be stored across the network and provide public and private storage. In table 3.1, I detail some examples of how blockchain has been used to create cloud storage platforms or supplement existing ones:

One gap found when researching these solutions was that

Paper	Description of Solution
Built with Blockchain	
Blockchain Based Data Integrity Verification in P2P Cloud Storage [15]	This paper uses Merkle trees to help verify the integrity of data within a P2P blockchain cloud storage network as well as looking at how different structures of Merkle trees effect the performance of the system.
Deduplication with Blockchain for Secure Cloud Storage [7]	This paper describes a deduplication scheme that uses the blockchain to record storage information and distribute files to multiple servers. This is implemented as a set of smart contracts.
Block-secure: Blockchain based scheme for secure P2P cloud storage [6]	A distributed cloud system in which users divide their own data into encrypted chunks and upload those chunks randomly into the blockchain, P2P network.
Blockchain-Based Medical Records Secure Storage and Medical Service Framework [2]	Describes a secure and immutable storage scheme to manage personal medical records as well as a service framework to allow for the sharing of these records.
Blockchain to Supplement a Cloud Platform	
A Blockchain-Based Access Control System for Cloud Storage [13]	This paper describes a method for using blockchain to facilitate the access control over a cloud storage system. The blockchain stores an immutable record of all '*meaningful security events*', such as key generation, access policy, assignment, etc.

Cloud Data Provenance using IPFS and Blockchain Technology [3]	Uses blockchain technology and IPFS to provide an efficient way to securely store provenance [1] data such that it is out of reach of adversaries, but can be used to verify the integrity of data on a cloud storage system.

Table 3.1: *Examples of blockchain cloud storage systems [12]*

3.2 P2P File Sharing

These applications involve a distributed network of computers that share data with each other without the need for a central party to facilitate. Table 3.2 shows some example p2p file-sharing networks.

One of the main issues with these networks come from their anonymity property in that you can never fully trust that what you're downloading isn't malicious. Using blockchain can add a layer of trust by allowing users to identify the author of an upload and match that to a real world entity, such as a company, or to their history of uploads within the network,

System	**Description of Solution**
IPFS [1]	IPFS is a content-addressable, block storage system and forms a Merkle DAG, which is a data structure that allows the construction of versioned file systems, blockchains and a Permanent Web. IPFS
BitTorrent [10]	BitTorrent is a p2p file-sharing system that has user bartering for chunks of data in a tit-for-tat fashion, which provides incentive for users to contribute to the network. Information about data is stored in .torrent files that can be found online and these help a user find other users interested in the same content they are. It is estimated that tens of millions of users use BitTorrent every day [14].
AFS [8, 4]	The Andrew File System was a prototype distributed system by IBM and Carnegie-Mellon University in the 1980s that allowed users to access their files from any computer in the network.
Napster [11]	Napster uses a cluster of centralized servers to maintain an index of every file currently available and which peers have access to it. A node will maintain a connection to this central server and will query it to find files; the server responds with a list of peers and their bandwidth and the node will form a connection with one or many of them and download the data.

[1] Provenance data are access logs of stored data that can trace the integrity of data and will contain private user information.

Gnutella [11]	Gnutella nodes form an overlay network by sending *ping-pong* messages. When a node sends a *ping* message to their peers, each of them replies with a *pong* message and the *ping* is forwarded to their peers. To download a file, a node will flood a message to its neighbors, who will check if they have and return a message saying so; regardless, the node will continue to flood their request till they find a suitable node to download off of.

Table 3.2: *Various global distributed file systems.*

Chapter 4

Design

4.1 Stakeholders & Requirements

4.1.1 Stakeholders

Game Developers These primary stakeholders will use the application to release their game and its subsequent updates to their users. These developers could be publishing individual projects or be a part of a major games studio.

Players These primary stakeholders will use the application to download and play games published to it. They will also help distribute the game for incentives provided by developers.

4.1.2 Functional Requirements

ID	Description
	Must
F_M1	Store software metadata on a blockchain, including a reference to a previous block where appropriate
F_M2	A node must request individual shards from its peers
F_M3	A node must be able to discover peers relevant to the software it wants
F_M4	Software must be updatable through the blockchain
F_M5	A node must be able to upload software
F_M6	A node must be able to download software in its entirety from nodes in the same network.
F_M7	A node must be able to verify the integrity of each block it downloads
F_M8	The application should run on the Ethereum network
	Should
F_S1	Allow users to restrict their software to only a specific set of nodes
F_S3	Allow a node to prove they have helped distribute software
	Could
F_C1	Allow users to request specific software versions

F_C2	Allow nodes to join groups for automatic identity verification

4.1.3 Non-Functional Requirements

ID	Description
	Must
NF_M1	The application is decentralized and cannot be controlled by any one party
NF_M2	Any user must be able to join and contribute to the network
NF_M3	Users should remain anonymous to other nodes in the network. However, to provide incentives and proof-of-purchase nodes will need to be identified by an uploader.
	Should
NF_S1	This application must be scalable, such that many users can upload and download the same software at the same time.
	Could

4.2 Design Considerations

Type of Blockchain

It is clear by **NF_M2** that the blockchain must be public. Some of the benefits we would gain from a public blockchain are that it:

- will result in a larger set of users, which will boost the availability of software,
- will reduce the risk of censorship as no one set of nodes will have authority over the network, and
- results in greater data integrity.

The Ethereum blockchain provides an extensive platform for development

Verifying Integrity

As per **F_M7**, a node must be able to verify the integrity of each shard of data. To do this, the application will leverage blockchain's inherent immutability property and store the hash data of all files in the body of each block.

Figure 4.1 shows how shard hashes are stored. Hashes are stored in a tree, which mimics the folder layout of the software such that the leaves are hashes and all other nodes represents directories or files.

Downloading Content

To achieve **F_M1**, **F_M2**, and **F_M6**, software in the application will be content addressable and nodes will communicate with each other to collect shards. In Section 2.1,

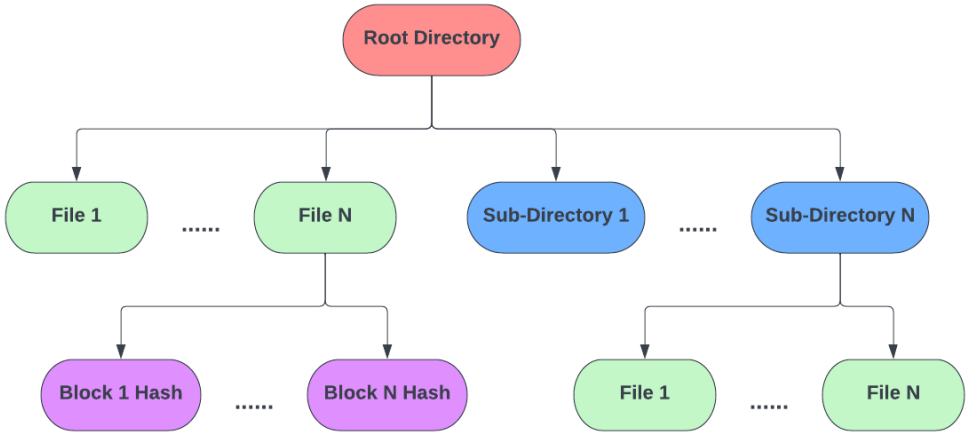

Figure 4.1: *How shard data is stored.*

I mentioned how BitTorrent nodes choose, which shards of data to request first and this application will use a similar model. The general steps for downloading data are:

1. A node will find the block containing information about the software they want,
2. They send out a discover request using the root hash to find peers that are also interested in the same software and then will form connections,
3. The node will then barter for shards of data by downloading and uploading with peers,
4. The node will download the entirety of the software and will continue to upload it.

The blockchain will be used to store useful metadata about each game similar to a torrent file. Blockchain's inherent immutability property combined with the use of public key infrastructure for identifying developers means that software is safe and verifiable.

Updating Software

As per **F_M4**, software must be updatable through the network. To do this, when a block represents an update to a piece of software it will include the hash of the block containing the previous version of the software.

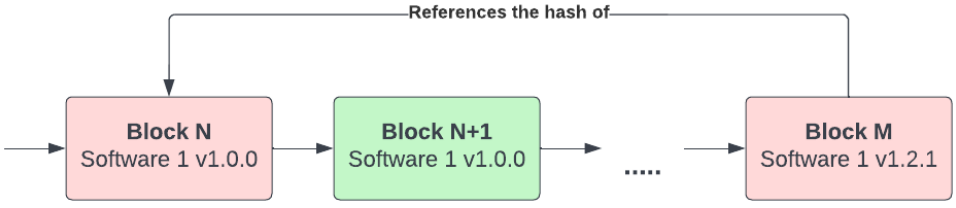

Figure 4.2: *How blocks can relate to older blocks.*

Each update to the software should contain its own complete directory tree, similar to Figure 4.1, but will likely contain duplicate hashes. When an update is released, a node will look for changes in the directory tree and request the corresponding shards. According to **F_C1**, shards from old versions of software could be persisted but this should be down to the choice of the user.

Digital Ownership

Using public key infrastructure, developers can identify themselves publicly to users by including their digital certificate within every upload they make to the blockchain. Users will purchase access to the software through the Ethereum blockchain, using ether, and will be provided with a purchase certificate that's encrypted with the developers private key.

This purchase certificate will need to be sent by users to seeders in the network before they can start downloading the software. This means that only users who have purchased the software can download it.

Proving Contribution

When a user purchases a piece of software they will be granted a unique seeder token. When a user successfully downloads a shard of data off of a peer they will reply with a confirmation message, containing this seeder token, that is encrypted using the developers public key. When a user wants to prove that they have contributed to the distribution of the game, they will send a collection of these messages to the developer, who will judge their validity.

Sequence Diagram

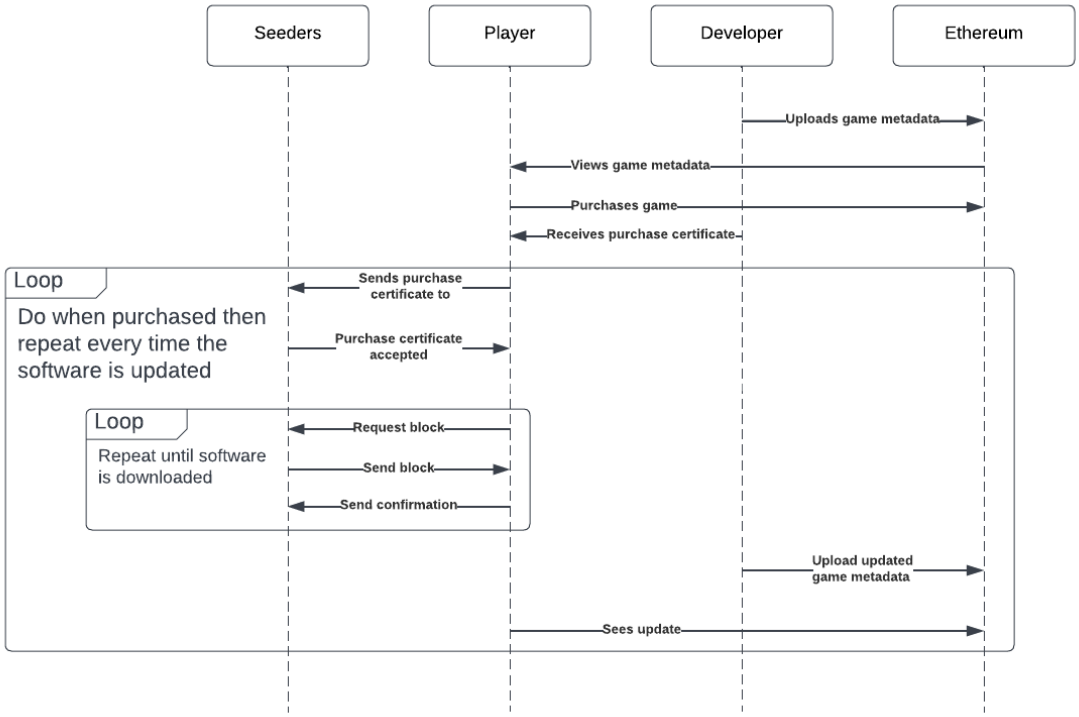

Figure 4.3: *The main interactions within the application.*

4.3 Limitations

This project will not attempt to mimic any of the social features (friends, achievements, message boards, etc.) provided by platforms like Steam. This may leave users with an inconsistent social experience for games as they will have to rely on the developers own implementations, other social platforms or just not have the features at all.

Like mentioned in Section 2.1, the main issue with p2p sharing is about the availability of content. The long-term distribution of a video game will often be reliant on the community, and the incentives provided by the developers, which may leave less popular titles unavailable.

Chapter 5

Project Management

5.1 Risk Assessment

Risk	Loss	Prob	Risk	Mitigation
Laptop damaged or lost	3	1	5	All work is stored using version control and periodic backups will be made and stored locally and in cloud storage. I have other devices that could be used to continue development.
Difficulty with blockchain development	2	3	6	I will seek advice from my supervisor about how to tackle certain problems and if necessary, what aspects of my project I should change.
The application is not finished	1	3	3	Using agile development will ensure that I will at least have a minimal working application. If I feel that I am running out of time, I will focus on expanding test cases and improving the write-up.
No suitable large scale test environment	2	5	10	I do not have the infrastructure to test this project on a large network, however small scale tests will be possible.
Personal illness	3	2	6	Depending on the amount of lost time, I may have to not complete some of the SHOULD or COULD requirements.

Table 5.1: *The risk assessment of this project.*

5.2 Work to Date

My work has primarily been on research, looking at how blockchain has been used to build and supplement cloud storage systems as well as how various peer-to-peer functioned and

performed. I have proposed a design for the application to be built on the EVM.

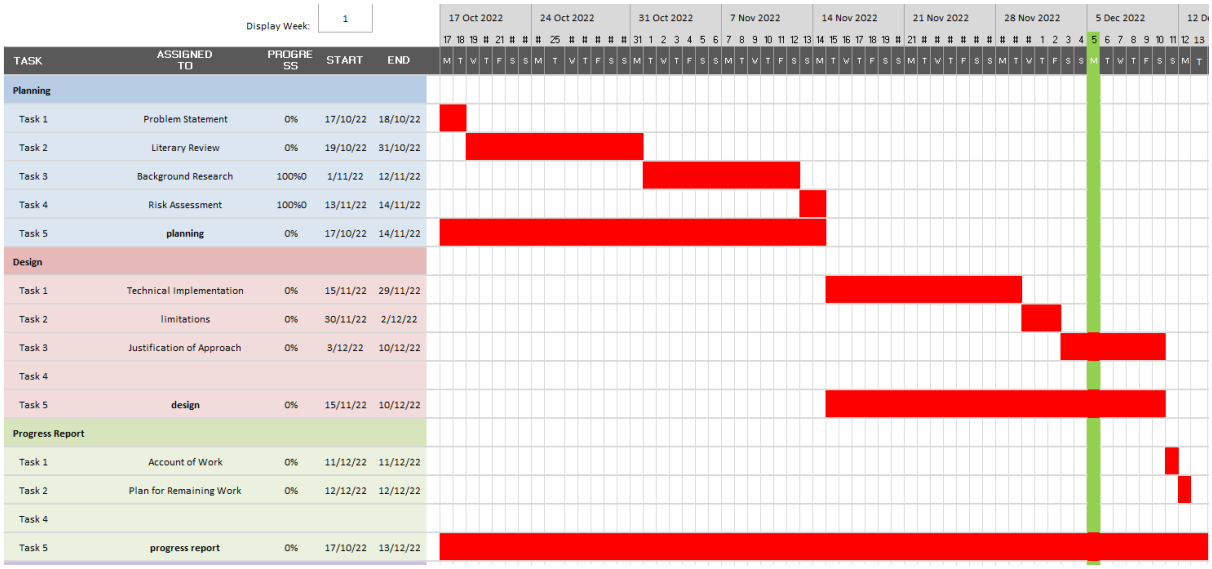

Figure 5.1: *A Gantt chart for my work up until the progress report.*

5.3 Plan of Future Work

Implementation & Testing This phase I will use the agile development methodology to build my application. My sprints will all be structured into three phases:

1. **Preparation** Deciding on the set of requirements to complete and making any initial design decisions and diagrams,
2. **Implementation** using test-driven development, I will work on requirements based on their prioritization, and
3. **Review** I will discuss the completed work in that sprint including design choices, what was completed, and any issues.

Testing Strategy and Results This phase will be used to discuss my strategy for testing and how it affected the overall success of my project. This will also be supplemented by a series of test results that show how my application fared against the tests I ran, including a discussion on any noteworthy test results.

Evaluation This phase will focus on me critically reflecting aspects of my project and will be used to discuss questions such as:

- How does my application fare as a solution to the initial problem?
- What changes to the application would make it more successful?
- What are the limitations of the application?
- What issues did I have during this project?
- If I were to do this project again, what would I change?

Bibliography

[1] Benet, J. IPFS - content addressed, versioned, p2p file system.

[2] Chen, Y., Ding, S., Xu, Z., Zheng, H., and Yang, S. Blockchain-based medical records secure storage and medical service framework. 5.

[3] Hasan, S. S., Sultan, N. H., and Barbhuiya, F. A. Cloud data provenance using IPFS and blockchain technology. In *Proceedings of the Seventh International Workshop on Security in Cloud Computing*, SCC '19, Association for Computing Machinery, pp. 5–12.

[4] Howard, J. H., Kazar, M. L., Menees, S. G., Nichols, D. A., Satyanarayanan, M., Sidebotham, R. N., and West, M. J. Scale and performance in a distributed file system. 51–81.

[5] Kaune, S., Rumín, R. C., Tyson, G., Mauthe, A., Guerrero, C., and Steinmetz, R. Unraveling BitTorrent's file unavailability: Measurements and analysis. In *2010 IEEE Tenth International Conference on Peer-to-Peer Computing (P2P)*, pp. 1–9. ISSN: 2161-3567.

[6] Li, J., Wu, J., and Chen, L. Block-secure: Blockchain based scheme for secure p2p cloud storage. 219–231.

[7] Li, J., Wu, J., Chen, L., and Li, J. Deduplication with blockchain for secure cloud storage. In *Big Data*, Z. Xu, X. Gao, Q. Miao, Y. Zhang, and J. Bu, Eds., Communications in Computer and Information Science, Springer, pp. 558–570.

[8] Morris, J. H., Satyanarayanan, M., Conner, M. H., Howard, J. H., Rosenthal, D. S., and Smith, F. D. Andrew: a distributed personal computing environment. 184–201.

[9] Neglia, G., Reina, G., Zhang, H., Towsley, D., Venkataramani, A., and Danaher, J. Availability in BitTorrent systems. In *IEEE INFOCOM 2007 - 26th IEEE International Conference on Computer Communications*, pp. 2216–2224. ISSN: 0743-166X.

[10] Pouwelse, J., Garbacki, P., Epema, D., and Sips, H. The bittorrent p2p file-sharing system: Measurements and analysis. In *Peer-to-Peer Systems IV*, M. Castro and R. van Renesse, Eds., Lecture Notes in Computer Science, Springer, pp. 205–216.

[11] Saroiu, S., Gummadi, P. K., and Gribble, S. D. Measurement study of peer-to-peer file sharing systems. In *Multimedia Computing and Networking 2002*, vol. 4673, SPIE, pp. 156–170.

[12] Sharma, P., Jindal, R., and Borah, M. D. Blockchain technology for cloud storage: A systematic literature review. 1–32.

[13] Sukhodolskiy, I., and Zapechnikov, S. A blockchain-based access control system for cloud storage. In *2018 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus)*, pp. 1575–1578.

[14] Wang, L., and Kangasharju, J. Measuring large-scale distributed systems: case of BitTorrent mainline DHT. In *IEEE P2P 2013 Proceedings*, pp. 1–10. ISSN: 2161-3567.

[15] Yue, D., Li, R., Zhang, Y., Tian, W., and Peng, C. Blockchain based data integrity verification in p2p cloud storage. In *2018 IEEE 24th International Conference on Parallel and Distributed Systems (ICPADS)*, pp. 561–568. ISSN: 1521-9097.