BlockTorrent: A Privacy-Preserving Data Availability Protocol for Multiple Stakeholder Scenarios

Ambrose William Hill

A thesis in fulfilment of the requirements for the degree of

Master of Engineering

School of Computer Science and Engineering
Faculty of Engineering The University of New South Wales

Prof. Salil Kanhere
Prof. Raja Jurdak
Dr. Volkan Dedeoglu

March 2023

ORIGINALITY STATEMENT

☑ I hereby declare that this submission is my own work and to the best of my knowledge it contains no materials previously published or written by another person, or substantial proportions of material which have been accepted for the award of any other degree or diploma at UNSW or any other educational institution, except where due acknowledgement is made in the thesis. Any contribution made to the research by others, with whom I have worked at UNSW or elsewhere, is explicitly acknowledged in the thesis. I also declare that the intellectual content of this thesis is the product of my own work, except to the extent that assistance from others in the project's design and conception or in style, presentation and linguistic expression is acknowledged.

COPYRIGHT STATEMENT

AUTHENTICITY STATEMENT

☑ I certify that the Library deposit digital copy is a direct equivalent of the final officially approved version of my thesis.

☑ The candidate has declared that **their thesis has publications - either published or submitted for publication - incorporated into it in lieu of a Chapter/s. Details of these publications are provided below.**.

Publication Details #1

Full Title:	BlockTorrent: A privacy-preserving data availability protocol for multiple stakeholder scenarios
Authors:	A. Hill, S. Mishra, A. Dorri, V. Dedeoglu, R. Jurdak and S. Kanhere
Journal or Book Name:	IEEE International Conference on Blockchain and Cryptocurrency (ICBC) 2021
Volume/Page Numbers:	103-112
Date Accepted/Published:	12/6/2021
Status:	published
The Candidate's Contribution to the Work:	I was the first and main author of the published works. The published work was going to be the first chapter of my PhD research, however, after deferring for one year as a job opportunity came up, I decided to transfer my PhD research into a master's degree. The master's thesis revolves around this published paper and expands on the work. I was responsible for the design, implementation, evaluation and completion of the paper and getting it submitted and accepted. I had supervision from my professors as well as an undergraduate student who helped with implementation. However, the core idea, the design of the system and its use in applications were all contributions of mine.
Location of the work in the thesis and/or how the work is incorporated in the thesis:	The published paperwork is sprinkled throughout the thesis however it is predominantly used in Chapter 3 (BlockTorrent) where the system is explained in detail.

Candidate's Declaration

I confirm that where I have used a publication in lieu of a chapter, the listed publication(s) above meet(s) the requirements to be included in the thesis. I also declare that I have complied with the Thesis Examination Procedure.

Abstract

As industries across the globe continue to digitize their processes, the need for a mechanism to share private data between multiple stakeholders is becoming increasingly apparent. However, sharing data poses challenges around privacy and accessibility, particularly in disputes between stakeholders with a shared interest, such as a supply chain. Auditors currently rely on stakeholders' compliance in order to verify data. Malicious parties may falsify the data before passing it on to the auditor. Using supply chains as a case study we present BlockTorrent, a protocol to address these challenges and help facilitate data sharing between supply chain participants and named after the integration of Blockchain technology and the BitTorrent protocol. BlockTorrent allows participants to securely share their data in near real-time with other participants without the risk of information leakage or allowing data falsification, whilst guaranteeing data availability for auditors. This is achieved using a novel combination of distributed storage and on-chain secret sharing. This thesis provides an implementation and evaluation of BlockTorrent, highlighting its performance and a security discussion, specifically that a system like BlockTorrent can reach large transaction throughput as high as 500 tps and be viable in a real world environment. Lastly, the thesis provides a discussion on the privacy challenges that were considered when designing BlockTorrent.

Acknowledgements

I would like to express my sincere gratitude to my supervisors, Salil Kanhere, Raja Jurdak, and Volkan Dedeoglu, whom without this work would have never been started or completed. They provided me with guidance that allowed me to grow both academically and personally and I will be forever grateful for their time and patience during my time completing this degree.

Contents

Abstract i

Acknowledgements ii

List of Figures v

List of Tables vi

Abbreviations vii

Publications viii

1 Introduction 1

1.1 Supply Chain on Blockchain 2

1.2 The Data Availability Problem 3

1.3 Research Questions 5

1.4 Contributions 6

2 Background 8

2.1 Technology Background 8

2.1.1 Blockchain 8

2.1.2 BitTorrent 11

2.1.3 Distributed Storage 12

2.2 Related Work 14

2.2.1 Research Gap 17

3 BlockTorrent 18

3.1 System Architecture 19

3.2 System Functionality 22
3.2.1 Hashed Sharding 23
3.2.2 Key Management 23
3.2.3 Storing Data 26
3.2.4 File Retrieval 28

4 Evaluations **33**
4.1 Implementation 33
4.2 Evaluation 35
4.2.1 Performance Evaluation 35
4.2.2 Security 40
4.2.3 Potential Attacks 43

5 Challenges and Conclusion **48**
5.1 Challenges and Discussions 48
5.1.1 Future Work 51
5.2 Conclusion 52

Bibliography **55**

List of Figures

2.1 How blocks are linked inside a blockchain. Source: intellipaat.com . . 9
2.2 Size of the bitcoin blockchain over the years. 13

3.1 A high-level overview of the proposed architecture. 20
3.2 The layers and interactions of BlockTorrent architecture 21
3.3 Transaction flow for the storing of data. 26
3.4 Transaction flow for file retrieving. 30

4.1 Time taken to split files into chunks 36
4.2 Time taken to distribute chunks across the network 37
4.3 Time taken to retrieve chunks and recreate the file 37
4.4 Transaction throughput and latency of storing and splitting the private key on the blockchain. 39
4.5 Transaction throughput and latency for querying a specific number of shards from a particular private key. 40

List of Tables

2.1 Summary of BlockTorrent's comparison to related work. 14
2.2 Summary of BlockTorrent's comparison to related work. 15

3.1 How the design decisions of BlockTorrent align with the requirements of the system 32

4.1 Identified security attacks and proposed countermeasures using BlockTorrent. 44

Abbreviations

BC Blockchain.

BFT Byzantine Fault Tolerance.

DFS Distributed File Systems.

IoT Internet of Things.

IoTD Internet of Things Devices.

NFT Non-Fungible Token.

P2P Peer-To-Peer.

TPS Transactions Per Second.

Publications

The main contributions of the thesis are based on the following publication:

A. Hill, S. Mishra, A. Dorri, V. Dedeoglu, R. Jurdak and S. Kanhere, *"BlockTorrent: A privacy-preserving data availability protocol for multiple stakeholder scenarios"*, In the IEEE International Conference on Blockchain and Cryptocurrency (ICBC), Sydney, Australia, 2021.

Chapter 1

Introduction

Blockchain technology has gained significant attention in recent years due to its potential to revolutionise various industries, including supply chain management[1]. Blockchain is a decentralised, digital ledger technology that allows for secure and transparent transactions without the need for intermediaries such as banks or government institutions to maintain its stability and security. It is a distributed database that stores data in blocks that are linked together in a chronological chain, creating a permanent and unalterable record of transactions. Each block contains a unique cryptographic code, or hash, that ensures the integrity and authenticity of the data. This technology has gained prominence as the underlying technology behind cryptocurrencies such as Bitcoin [2]. However, its potential applications go far beyond simple digital currencies. It has the potential to revolutionise various industries, including finance, supply chain management, healthcare and law. This is due to one of the most significant features of blockchain, the fact that it exists exclusively in decentralised environments. Since it is a distributed ledger, no single entity controls the data, making it in certain situations, more secure and resistant to hacking and fraud. Additionally, transactions on the blockchain are transparent and traceable, enabling greater accountability and trust in the data accumulated in the chain. The process of duplicating the chains and distributing them to multiple users on the network leads to a large amount of confidence that the data is not being manipulated, in fact, blockchains are considered immutable because of this property. There is, however, one problem that arises because of this property which is commonly referred to as the "Rubbish In, Rubbish Out" dilemma. Since the data cannot be changed or altered once it is submitted to the blockchain, you have to be very careful about how the information is being sent and to somehow ensure

that the information is both valid and accurate. For cryptocurrencies where the validity of a transaction is based on a user's balance alone, this is less of a concern, but when you consider a blockchain application on a supply chain, where the data being stored is non-uniform and complex this becomes a problem. Alongside this, in a supply chain, most entities would not be willing to share private information so there is a need for a solution that incentivises untrusted parties to share and provide potentially sensitive information with each other.

This thesis aims to support blockchain technology in its endeavour to solve problems plaguing some of the biggest industries in the world. Specifically, the problem of data availability within supply chains. Blockchain has the potential to solve many problems but only if the technology is understood and applied correctly. This thesis is laid out as follows: First, background information on blockchain technology, supply chains and the data availability problem, this is followed by the introduction of the BlockTorrent Protocol, the main contribution of this thesis, before finally providing an evaluation and discussion of the proposed architecture with a discussion around the security and privacy implications of such a solution.

1.1 Supply Chain on Blockchain

A supply chain, in essence, is a network of businesses, individuals, and activities involved in creating and delivering a product or service to the end consumer. There are many moving parts to a supply chain including, logistics, administration, financing and auditing that are required to work in unison in order to manage a supply chain. Managing a supply chain effectively is crucial to ensure that the right product is delivered to the right place at the right time. Supply chains exist in almost every industry across the globe and they are used to create efficiencies in logistics in all areas, from raw resources to manufactured products. Supply chains are a vital cog in the global ecosystem and as a result of their importance, it would be easy to assume that they are already optimised and heading towards perfection. Whilst this may be true for specific trades, it is almost certainly not the case for the vast majority of supply chains operating around the world. Recently, the digital age has provided mechanisms for transferring information at rapid speeds regardless of physical distance however there is still one major problem plaguing all supply chains

in existence, how do you trust that the data you are receiving is accurate? The short answer is you can't, but there are steps you can take to help eliminate untrusted data.

Traditional supply chain management systems face several challenges, including limited visibility, high costs, and inefficiencies [3]. Blockchain technology offers a solution to these challenges by providing a transparent, secure, and decentralised platform for supply chain management. In a blockchain-based supply chain, transactions are recorded on a distributed ledger that is accessible to all participants in the network. This enables real-time tracking of products and transactions, enhancing transparency and accountability throughout the supply chain. Additionally, the use of smart contracts, which are self-executing agreements with the terms of the contract directly written into code, can automate various processes in the supply chain, reducing costs and increasing efficiency [4].

Overall, supply chain on blockchain has the potential to transform the way businesses manage their supply chains, leading to increased transparency, efficiency, and trust among all stakeholders but there are still large problems facing the technology and its applications within supply chains [3].

1.2 The Data Availability Problem

Simply put, blockchain provides a technology that can maintain information and protect it against hacks and data corruption, but there is an inherent need to trust the information that was initially put on the blockchain and going one step further, to trust the source of the information. A malicious participant on a blockchain could have a facade of being honest and transparent with its information but behind closed doors, is manipulating the data they are entering on to the blockchain. The malicious entity could also just refuse to share accurate information when requested, leading to a situation where blockchain is used but the data stored on the blockchain is effectively useless. As mentioned, this is commonly referred to as "Rubbish in, Rubbish out" in the blockchain industry.

To illustrate this problem a little clearer, let us use a specific example such as the construction supply chain. The construction industry is one of the longest-running industries of human development to date. From turning mud into bricks

and metal into skyscrapers, it has allowed human creativity and innovation to thrive. There is a clear improvement when it comes to construction, it can be seen as one drives through a city and notices the subtle differences in buildings that were constructed in different eras, and although it is easy to see the improvements in the physical design and construction of buildings, there is a significant element of the construction industry that has not received the same amount of innovation and that is the administration of construction projects.

The construction administration system is one of many systems in the world today that heavily rely on participant compliance when it comes to data sharing and accessibility. The most common form of these systems is the supply chain [5]. A construction supply chain involves a group of organisations that are responsible for facilitating the transfer of resources and services from suppliers to construction sites. Recently, Internet of Things (IoT) sensor devices have been integrated with some supply chains allowing for the automatic tracking of items during their journey from supplier to end user. Furthermore, the sensorisation of supply chains has given the stakeholders access to real-time data allowing for optimisation and improving efficiency.

Participants willing to share real-time data amongst themselves will facilitate faster trades, enjoy lower operational costs and have the ability to detect and competently rectify delays. However, data sharing among supply chain participants poses privacy and security challenges. Sharing data can lead to information leaking, such as contextual details of quality or quantity of supplies or company secrets like the temperature used to store resources. These challenges make it highly critical for the supply chain entities to keep all data secure and private, particularly data captured automatically i.e. IoT sensor data. In response to this, many supply chain entities prefer centralised solutions for managing their digital data. Although this solves privacy concerns, it causes an issue when an audit is required. Each entity can manipulate or misrepresent the data that is supplied solely by them. Some existing solutions require hashes of the data to be shared at the point of capture so that the data cannot be changed at a later date. However, storing only the hash cannot solve this problem as the participant that provided the hash can claim that the data has been lost or manipulate the data in advance so that the hash reflects the corrupted information.

Blockchain is one potential technology to help facilitate a solution to this problem.

Blockchain supports multi-stakeholder applications, such as a supply chain, with data immutability, audibility and access control [6]. Blockchain-based supply chains have seen increased interest due to their improved scalability as seen in [7]–[10]. Although blockchain is a promising solution to address supply chain challenges [1], the overarching problem is that anyone on the network can access data stored on the blockchain.

To solve this issue, a few avenues are available to ensure data privacy on a blockchain, such as using a permissioned chain or a mechanism like channels that exist on Hyperledger's blockchain solutions[11]. Even then, these solutions come with challenges when trying to share data that only exists on a private channel or permissioned chain.

Participants could collude to store incorrect or inaccurate information, choosing to hide information or selectively storing partial information in the first place.

Furthermore, supply chain entities could use the blockchain as an indexed database, only submit hashes of the information to the blockchain, and store the bulk of their data off the chain. This would help to prevent the leakage of privacy-sensitive information.

There are alternate methods of storing data on a blockchain, such as requiring the shared information to be encrypted. However, a malicious party could claim they lost the decryption key and assuming the encryption standards are high this still results in missing or corrupted information.

While this approach ensures data privacy, it impacts the accessibility of the data. This leads to the main concern that data can still be made intentionally unavailable. Whether this information was inaccurate or misleading or intentionally corrupted so that it could never be accessed again, there is a critical need for mechanisms that ensure data availability and integrity to authorised parties while preserving the privacy of data contributors.

1.3 Research Questions

As with all areas of research, there are countless questions to be answered. For the purpose and scope of this thesis, there will be a specific focus on the transfer and trust of information within supply chain.

This thesis aims to answer the following research questions around blockchains integration with supply chain and the Internet of Things.

- What supply chain processes can blockchain help advance?
- How can blockchain improve the trust of information inside supply chains?
- Can blockchain or other distributed storage methods be used to guarantee access to data in multi-stakeholder scenarios, like supply chains?

1.4 Contributions

This thesis's main contribution is presenting a solution to the data availability problem described above. This thesis proposes BlockTorrent, a novel blockchain-based data management protocol that enhances data availability while protecting participants' privacy. BlockTorrent is an integration of BitTorrent [12], a Peer-To-Peer (P2P) file-sharing protocol and Blockchain.

Although BlockTorrent can be used by any application that requires data availability among multiple untrusted stakeholders, we focus on the supply chain context as a representative case study. To ensure data accessibility, BlockTorrent distributes data among a set of peers in the blockchain. The participating nodes should be able to read the data for audibility; however, this compromises user privacy. In BlockTorrent, data is split into multiple chunks, and each chunk is sent to a randomly chosen node, another user, for storage. The selection of nodes involves hashed sharding, which is a random process where each node in the network is assigned a range of values, each chunk is hashed and then distributed to the nodes based on the value of the digest [13]. Data splitting helps prevent unauthorised access, as even if an adversary can decrypt a chunk, they still need every other chunk to retrieve the complete file. It also reduces the load on the network through the transmission of small packets rather than the bulk transfer of IoT data logs, similar to how the BitTorrent [14] protocol distributes files.

We provide a solution for maintaining the data through its lifecycle. As the data is being dissected, it is necessary to store how the data is being split up and where each piece is located so that it can be recreated later on. BlockTorrent uses a permissioned blockchain to store the metadata of each distributed chunk. Our

protocol allows each participant in a supply chain to continually share their data with other participants without leaking private information or impacting the supply chain performance.

As explained above there is a real world need for systems and protocols to maintain data availability through the data's entire lifecycle. Leveraging two new technologies, in blockchain & bit torrent, in a way that solves an impact problem like data availability is significant research that should be continued in any and all directions possible. This thesis and its contributions are only a small part of the blockchain research industry and could lead to significant innovations in the near future.

The major contributions of this thesis are:

- A data-sharing protocol that can be used by participants to share their private data on the main blockchain securely. These participants can be generalised to any untrusting stakeholders The protocol enhances data availability while preserving data confidentiality, as it distributes chunks of data to random peers on the network while securely storing the encrypted details of file sharing on the blockchain. The secret sharing process is implemented using a smart contract such that no participant can manipulate the process.

- A generalised blockchain and IoT architecture that can be applied to most supply chains in existence today.

- An evaluation based on an implementation of the protocol that demonstrates the system's robustness and resistance to major security attacks and efficacious performance in terms of scalability.

- A discussion about performance and privacy trade-offs that came into consideration while designing the BlockTorrent protocol.

Chapter 2

Background

This chapter aims to provide background information on the overall supply chain industry and the technology that is being used to improve it. Following the background information there will also be an overview of the related work in the fields of distributed storage and data availability. As BlockTorrent is a solution based on blockchain technology the related work and background will have a focus on solutions that involve blockchain.

2.1 Technology Background

2.1.1 Blockchain

Blockchain technology is a distributed and decentralised digital ledger that is used to record transactions across many users in a way that is secure, transparent, and permanent. It was originally developed in 2008 [2] by an anonymous person or group using the pseudonym Satoshi Nakamoto as a way to facilitate transactions on a decentralized digital currency called Bitcoin. At its core, a blockchain has two main components, a data structure and a consensus algorithm. Usually, the data structure will consist of a chain of blocks, where each block contains a list of transactions. Each block in the chain is linked to the previous block through a cryptographic hash function, which ensures that once a block is added to the chain, it cannot be altered or deleted without invalidating the entire chain. A consensus algorithm is a protocol that can be used by a network of users to come to an agreement or make a decision without the need for consistent authority [15].

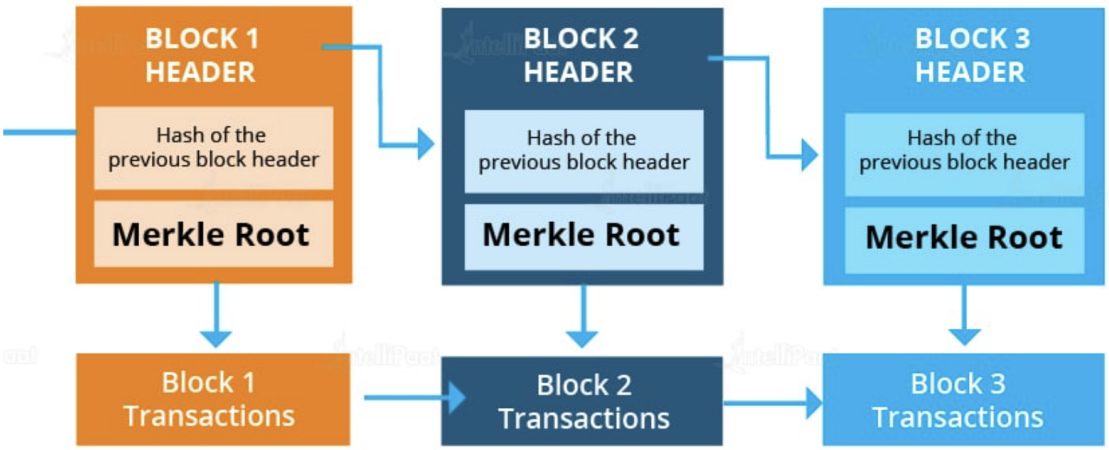

Figure 2.1: How blocks are linked inside a blockchain. Source: intellipaat.com

There are many different consensus algorithms in use today [15] and one way of classifying them in a helpful manner is to use Byzantine fault tolerance (BFT) algorithms. BFT is a concept in distributed computing that refers to the ability of a system to function correctly and reach a consensus even in the presence of faulty or malicious components. This is helpful as when you start operating in distributed computing environments it is hard to identify all participants. In fact, cryptocurrencies boast that they allow for secure anonymous entities, so when you need to be able to stop malicious actors from being able to compromise your network but you can't authenticate any entities. There is an added problem as well because you have all nodes executing the same identical protocol, there is no element of secrecy in the protocol itself. This means that the protocol has to be resilient when all of its processes and mechanisms are publicly known [16].

The term 'Byzantine' comes from the 'Byzantine Generals Problem', a theoretical problem in computer science that asks how a group of generals, each commanding a portion of an army, can come to an agreement on whether to attack or retreat, despite the possibility of some of the generals being traitors who may send conflicting messages. The problem is used as an analogy to describe the challenge of achieving consensus in distributed computing systems where some nodes may be unreliable or malicious [17].

In a BFT system, nodes in the network work together to reach a consensus on the state of the system. Each node has a copy of the current state of the system

and can propose updates to it. The system is designed to detect and mitigate any faults or attacks, such as messages that are intentionally altered or delayed by malicious nodes, to ensure that the system continues to operate correctly. An algorithm being BFT is important in blockchain technology, as it is used to ensure the integrity and security of the network [16]. In blockchains, nodes work together to validate transactions and add them to the ledger. The system must be able to reach a consensus on the state of the ledger, even in the presence of faulty or malicious nodes. Byzantine proof helps to ensure that the system remains secure and trustworthy, even when some nodes are compromised or colluding.

The decentralised nature of blockchain means that there is no need for a central authority or intermediary, such as a bank, to validate or process transactions. Instead, transactions are validated by a network of nodes, or computers, that are connected to the blockchain network [11]. The protocol that handles these validations is known as a consensus algorithm. Each node on the network has a copy of the entire blockchain, which means that transactions are verified and recorded by multiple parties in real-time and makes it increasingly difficult to lie, cheat or misled other users as they have a full copy of the information as well.

In addition to being decentralised and secure, blockchain technology has several other key features that make it a popular choice for a variety of applications. These include:

- Transparency: Because the blockchain ledger is distributed, all transactions that occur on the network are visible to anyone with access to the network. This provides a high degree of transparency and accountability, which is particularly useful in industries such as finance, where transparency is essential.

- Immutability: Once a block is added to the blockchain, it cannot be altered or deleted. This means that the data stored on the blockchain is permanent and tamper-proof, providing a high degree of security and trust.

- Efficiency: Blockchain technology enables transactions to be processed quickly and efficiently, with no need for intermediaries or lengthy settlement periods. This can help to reduce costs and increase efficiency in industries such as supply chain management and real estate.

- Smart contracts: Blockchain technology can be used to facilitate the creation

of self-executing smart contracts, which are computer programs that automatically execute the terms of a contract when certain conditions are met. This can help to reduce the need for intermediaries and increase the efficiency and transparency of contract execution.

The concept of this data structure has been around since the early 1990s in a data structure known as Hash chains, the integration of the data structure and consensus algorithm is where the true novelty and brilliance of bitcoin lies. This small integration of two technologies is what led to cryptocurrencies, Non-Fungible Tokens (NFTs) and Web 3.0 as a whole.

Overall, blockchain technology has the potential to transform many industries by providing a secure, decentralised, and efficient way to facilitate transactions and exchange value [18]. These new ways of exchanging information can also help improve trust in the information itself and in systems as a whole. While it is still a relatively new technology, it has already been adopted by a variety of industries and is likely to continue to grow in popularity in the coming years.

2.1.2 BitTorrent

BitTorrent is a peer-to-peer file-sharing protocol that was first developed by Bram Cohen in 2001 [19]. It is a decentralised method of sharing large files over the internet that has become one of the most popular ways to distribute files, such as movies, music, and software. BitTorrent works by breaking files into smaller pieces, which are then shared among a network of users. This allows users to download and upload files segments simultaneously, making the process faster and more efficient than traditional file-sharing methods.

One of the key benefits of BitTorrent is that it uses a distributed network of users to share files, rather than relying on a centralised server. This means that files can be downloaded and shared from multiple sources, making the process faster and more resilient to network failures. Additionally, BitTorrent's use of smaller file pieces means that users can download only the parts of a file that they need, rather than having to download the entire file all at once. This can help to reduce bandwidth usage and improve download speeds. This aspect of BitTorrent is similar to blockchain's decentralised trait, as both BitTorrent and blockchain exist exclusively in peer-to-peer networks.

However, BitTorrent has also been associated with piracy and copyright infringement, as it is often used to share copyrighted content without permission. This has led to legal challenges for BitTorrent and its users and has prompted efforts to develop more legal and legitimate uses for the technology. One of those uses is by game development giant Activision Blizzard. Blizzard uses a custom BitTorrent client to distribute updates for its games, including World of Warcraft, StarCraft II, and Diablo 3. This helps speed up downloads for everyone by allowing people to share their upload bandwidth with others, leveraging this previously unused bandwidth tends towards faster downloads for everyone. Of course, it also saves Blizzard money on their internet usage bills as well as bringing legitimacy to a notorious data exchange protocol.

Even with its association with piracy, BitTorrent has proven to be one of the fastest file-sharing protocols currently in use. Its efficient file size and leverage of every user's bandwidth make it a good fit with blockchain. If integrated correctly a system with both blockchain and BitTorrent could attempt to speed up information transfer. Both BitTorrent and Blockchain exist on P2P networks leading to the question of how are these two technologies best integrated.

2.1.3 Distributed Storage

As mentioned blockchains are immutable and generally you need all blocks in the chain to be able to correctly validate new transactions. This leads to a problem as blockchains become more popular and have increased traffic, where do we store all of the blocks? There is also the question of redundancy, if a blockchain requires every user to store every block, there is 100% data redundancy for each user. Whilst this brings security and legitimacy to certain blockchains (such as bitcoin), there are many applications where it is infeasible or improbable to expect every participant to store every block and/or transaction, a supply chain for instance is one of these applications.

The bitcoin blockchain network has grown consistently over the 14 years since its inception. As seen in Fig 2.2, the network is currently over 460 GB in size. A blockchain operating in a traditional fashion would expect all of its users and nodes to store a full copy of all that information. It would also require any new users wishing to get an up-to-date copy of the blockchain to wait until they had

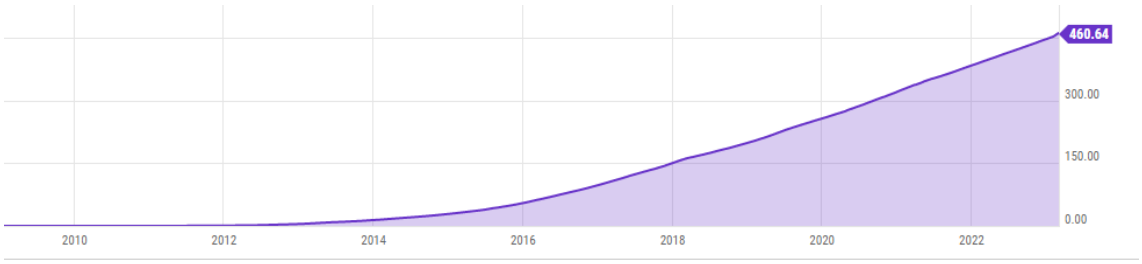

Figure 2.2: Size of the bitcoin blockchain over the years.

completed a 460 GB download. Even with the fast internet speeds of today that could take days to complete and it is unreasonable to expect that from an everyday user. A common approach to minimise this is by having different nodes for different purposes. There are many types of nodes but the main ones that can be seen on almost any blockchain are full node, light node and miner/validtor node.

1. Full Node: This is a complete copy of the blockchain and can validate any and all transactions being submitted to the blockchain. They maintain a complete and accurate copy of the chain. For bitcoin this would be all 460GBs.

2. Light Node: This is a lightweight version of a full node where a complete copy of the blockchain is not required. They generally store the most recent blocks and can validate new transactions, but cannot look back past the first block they stored. These are designed to run on lightweight devices with bandwidth or storage limitations. If a light node is missing any vital information it will communicate with a full node until it has all the information required.

3. Miner/validator Node: This node is responsible for adding new transactions to the blockchain. How this is done is dependent on the consensus algorithm employed by the blockchain. For bitcoin, this is Proof of Work (PoW) and for Ethereum, this is Proof of Stake (PoS). This is where complex calculations are being executed and generally, they do not store a copy of the blockchain, rather just the most recent blocks or any other information required to validate and add transactions.

The problem of large data storage affects all blockchains and has led to a large research area of computer science, namely distributed storage, and at the core this is one of the problems this thesis is attempting to solve.

There is also the concern of how do we get around storing data in a distributed fashion. We could force supply chain participants to use a blockchain but whose servers or computers are being used to store the data? Most large entities will want to use centralised storage solutions as they most likely will have sensitive information to protect and more than anything else, they do not trust the other participants to handle their data securely. This forces any solution that wants to be adopted into having separate storage solutions or not being able to store any data of importance on the blockchain.

2.2 Related Work

This section will outline the current work that has been done in the field of data availability in distributed systems, starting with two tables that summarise the largest pieces of work and their shortcomings.

Related Work	Decentralised	Availability	Privacy
Ethereum Swarm [20]	✓	✓	×
IPFS [21]	✓	×	×
BTT [19]	✓	×	×
BigChainDB [22]	✓	✓	×
IoTSmartContract [23]	×	×	✓
Controllable BC Data [24]	×	✓	✓
Secure IoT [25]	✓	×	✓
BlockTorrent [Our Work]	✓	✓	✓

Table 2.1: Summary of BlockTorrent's comparison to related work.

Table 2.1 gives a summary of the key features used in related literature and production systems. This section will provide a brief discussion of the related work in distributed data storage, data sharing, and data availability.

The authors in [26] present a survey of the current Distributed File Systems (DFS) and their integration with blockchain. They discuss two main solutions in Ethereum's Swarm and IPFS and compare the two via a wide range of metrics. They also provide a seven layer framework that any DFS should have, that consists of identity, data, data-swap, network, routing, consensus and incentive layers. In-

Related Work	Integrity	Generalised Data Store
Ethereum Swarm [20]	✓	✓
IPFS [21]	✓	✓
BTT [19]	✓	✓
BigChainDB [22]	✓	×
IoTSmartContract [23]	✓	✓
Controllable BC Data [24]	✓	×
Secure IoT [25]	×	×
BlockTorrent [Our Work]	✓	✓

Table 2.2: Summary of BlockTorrent's comparison to related work.

creasing scalability of the framework and the privacy of the users while sharing data are mentioned as the fundamental challenges and future research directions.

In [27], the authors discuss the major issues and challenges for data storage in blockchain. The proposed solution stores the data in an offline storage medium and the corresponding hash in the blockchain. It also mentions "BigchainDB", which enables blockchain-like trusted transactions on top of an existing modern distributed database system. The authors further discuss the issues with blockchain storage such as the impact on the "Right to Forget". The paper does not mention any methods to handle data distribution or splitting up data for efficient storage, access, and availability.

The authors in [23] discuss a system where data is encrypted and stored using a Trusted Computing Environment - Intel SGX, and the corresponding hash of the data is stored in the blockchain. A third-party who needs to access the data can request the data and check integrity via the hash. The system does not have a data splitting mechanism for splitting and reconstructing information which can lead to bottleneck congestion when the network is busy. There is only one host of the data, i.e. the data owner, hence, it can be very difficult for other parties to acquire that data. There is no distribution protocol mentioned as they attempt to solve the challenge of data management through trusted environments. The Intel SGX can be used to provide this environment but it forces users of the system to buy specific, potentially expensive hardware.

The authors in [24] propose a system that stores documents in a blockchain-based cloud server and keeps track of the changes being made to the documents. Any

user of the network can upload a document to the system. If the document is to be altered by another user, then he has to send the changes to the owner in the form of encrypted messages, which have to be validated. There is a Trusted Authority (TA) which has control over the network. All the requests are sent to the TA, which keeps track of the changes made to the document, also centralising the system. The TA has the veto power to cancel any changes to a document as well as monitor the user's identities and behaviors. Moreover, the documents are stored in a cloud server furthering the centralisation of the network. Therefore, if the TA acts maliciously or the cloud server is accessed by a malicious party, then the security can be easily breached. The system relies on standard networking protocols for data distribution and access and does not provide functionality for splitting up documents, which can lead to slower access times in the network. A system that relies on a TA could be improved by using a smart contract to manage the TA's actions on the network and depending on the complexity of the contract, act as an authority figure itself.

In [19], the authors propose BitTorrent Token (BTT), a crypto token that is attached to the BitTorrent network. It is used as a reward for users that seed content which incentives more users and creates a more active network, overall boosting download speeds. Users can then use these tokens to pay for a faster download speed from other users. The token will be integrated with the BitTorrent File System which is a proposed decentralised file storage system that will make use of the millions of BitTorrent nodes. The main purpose of BTT is to facilitate faster downloads for the torrent and file storage networks and not guarantee data accessibility.

In [25], the authors propose a system that splits the data into data chunks and then distributes the data chunks among the nodes using a proximity metric. The system is structured into two planes: control and data. The data plane uses a cloud-based service to store the data. The control plane is built on a blockchain and keeps the access control policies and metadata on the stored data. To reduce the storage and bandwidth requirements, data is compressed before encryption. The trade-offs of their proposed system, such as the data availability vs. the number of database replications are not discussed. Similarly, they have not considered how a party can reconstruct the whole data from the distributed data chunks.

In [28] the authors propose using Shamir's (k,n) key sharing technique in combination with a partitioned blockchain to improve data integrity and privacy. Blocktor-

rent also uses this key sharing technique and it will be explained in detail in Section 3.2.2. They formulate a cloud storage system that is able to distribute data storage amongst the peers on the network, however the system does not guarantee data availability. The authors do not provide an implementation and their evaluation is theory based.

The authors in [29] propose a secure online storage system that splits up the storage of information and metadata. They make use of the blockchain to store the metadata and distribute information over a P2P network similar to BitTorrent. The system relies on users to create their shared secrets independently which does not guarantee availability. The authors also do not provide an implementation or performance evaluation.

In [30], the authors propose an improvement to the BitTorrent protocol by introducing agents known as replicators. These replicators are used to optimise the file download speed in BitTorrent networks by alleviating the strain on the seeders of popular files. The authors provide proof for how these agents increase the download speed for other peers on the network. This solution focuses on popular files which change over time, so it is not suitable for private data that may never become popular. The paper also lacks a security and privacy discussion around replicating data and relies on the security present in BitTorrent.

2.2.1 Research Gap

So far, distributed storage approaches have been unable to strike a balance between protecting the sensitive information of data owners and ensuring data availability and integrity to authorised parties. The current state of the art does not offer solutions that distribute, route, find and reconstruct chunked data, rather they focus on using the immutability feature of blockchain to store the data hash. Related works also lack a thorough privacy and security evaluation, thus leaving room for future work to address these challenges by providing discussions and evaluations of potential solutions.

Chapter 3

BlockTorrent

This chapter outlines the details of *BlockTorrent* using supply chain applications as an example scenario. Accessing information in large supply chains is a challenging problem. This is largely due to the ownership of the information, as each individual organisation has sole ownership of their captured data and there is currently no process to guarantee access to that data. Ideally, supply chain agents would act in good faith and there would be no need for third parties to access the private information of a supply chain participant. However, due to imperfect processes, there are disputes and delays at many stages of the supply chain. Each of these disputes can cause an expensive and lengthy resolution process which incentivises companies to misconstrue the information they present to a mediator. Hiding, manipulating or claiming ignorance only furthers the delay, impacting more participants and increasing costs.

BlockTorrent is a privacy-preserving protocol that ensures data availability during audits. BlockTorrent combines blockchain technology as the underlying platform, to ensure immutability and availability through replication, and the BitTorrent protocol to share large amounts of data chunks with each participant, which enhances data privacy through data splitting. BitTorrent has been proven to be an effective file-sharing protocol in P2P networks, and it further improves BlockTorrent's transfer times, allowing for near real-time data sharing.

Each participant in the supply chain is incentivised to share supply chain-related data but in a way that it is secured from non-authorised parties. This incentive is twofold: *firstly*, BlockTorrent uses penalties for non-compliance, which can be financial or in extreme cases result in removal from the system, furthering the incentive for honest participant behaviour. A point of consideration here is that in a real world application it can be hard or even impossible at times to completely remove

a participant from a network, in these cases the fine would need to be large enough to incentivise honesty. *Secondly*, a participant honestly complying with BlockTorrent will be able to provide reliable evidence that shows they are adhering to legal industry regulations, if relevant. Since the data is being shared as it is captured, the chance of it being manipulated in favour of the participant is slim, which helps verify compliance.

First, we will introduce the network layers and their interactions within the system architecture. BlockTorrent relies on three key networks set up in parallel in order to reduce bottlenecks in network traffic. After, we will outline the functionality that the proposed system offers and describe how it solves the data-sharing challenges explained in the previous sections. While we use a supply chain as an example scenario, it should be noted that this solution is application-agnostic and can be used as a data-sharing mechanism within any information system.

3.1 System Architecture

In this section, we outline the system architecture for the data-sharing framework defined above. First, we define the key entities of the architecture:

- *Participant:* Any organisation that is a part of the supply chain employing BlockTorrent. This could be the supplier, transporter, retailer or an authority representing the local, state or federal governments.
- *Admin Nodes:* A group of nodes, one for each participant that is responsible for accessing all layers of the network. These nodes monitor and maintain the data-sharing mechanism for each organisation.
- *Auditor:* A unique participant that is responsible for auditing the supply chain data and is the entity that is responsible for mediating the dispute resolution process between participants.
- *User:* All other users interacting with the system such as a buyer.

The participants involved in the supply chain jointly form a consortium blockchain where they are able to communicate and exchange data. This blockchain will be referred to as the main chain. Each organisation also has its own private database

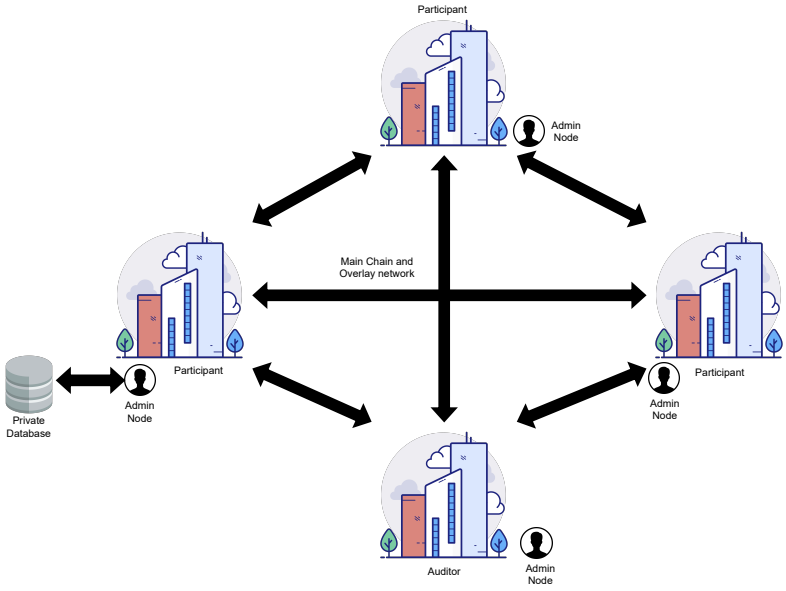

Figure 3.1: A high-level overview of the proposed architecture.

that contains the associated data of the sensors and any other information required for processes along the supply chain. This database is owned and controlled by the participant and is referred to as the private database layer. Each participant is a member of the overlay network, which facilitates all off-chain communication for BlockTorrent. The admin node for each participant has access to the main chain, the overlay network as well as its organisation's private database in order to carry out the data-sharing mechanism.

BlockTorrent can be used with a private database of any form, including another blockchain. This private chain is secure from other participants on the main chain but also allows members from one participant to share and validate information amongst themselves. Fig. 3.1 displays the basic setup for a participating organisation.

As mentioned earlier, there are three key components: the main chain, the private database and the overlay network.

Main Chain: Serves as the interface between organisations and BlockTorrent. Each participant in the supply chain has access to the main chain and the ability to read and write transactions. Participants are added to the main chain when they join the supply chain consortium and use it as the source of truth in the network.

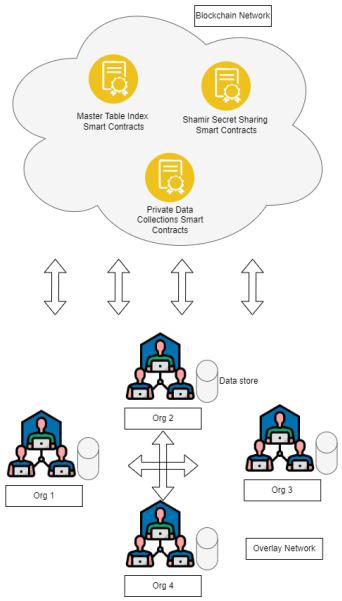

Figure 3.2: The layers and interactions of BlockTorrent architecture

The main chain facilitates the sharing of encrypted data between the parties. This includes both participant-to-participant data sharing and audit requests. The protocol has been designed to limit traffic on the main chain as it is distributed between all participants and can cause congestion in the network. As a result, metadata of shared chunks and decryption keys for these chunks are stored on the main chain.

Private Database: The private database is used for each individual member's private business data. Its purpose is to facilitate the use of the data while securing it from the public network. This can be any type of database, the only requirement is that the data is accessible from a node capable of performing BlockTorrent functions, i.e, not a computationally-constrained device. The private database can be in any format required and depending on the needs of the system can choose to prioritise security, transparency or privacy.

Overlay Network: This is the network that facilitates all non-main chain communication and storage. The chunks of data are distributed on the overlay network as per the BitTorrent protocol. Auditors on the overlay network will interface with BlockTorrent's main chain to access the metadata of a chunk and then request that chunk from a peer on the overlay network. The admin node consortium, an off-chain network consisting of at least one admin node from each participant, resides on the

overlay network. This consortium is responsible for ensuring each file passed to the overlay network is split and distributed in a random manner. This is achieved using hashed sharding [13] and is explained in Section 3.2.1.

3.2 System Functionality

Note that any solution to this problem is going to have to decide between prioritising security or privacy as there is a clear trade-off between them when discussing data availability. Table 3.1 shows the design decisions made while developing BlockTorrent and how they match up with requirements of a data availability system. Overall, there is a trade off with how much privacy you want to give participants and their data in the system, but certain sacrifices have to be made in order to ensure data availability at all times.

If the decryption keys are kept only by the owner of the file, an owner could 'lose' the key making the corresponding files inaccessible. This is a similar scenario that was highlighted in Section 1, where a participant could use BlockTorrent but refuse to share the decryption keys when requested. To address this security risk a key distribution mechanism, similar to Shamir's secret sharing scheme [31], can be implemented which ensures the availability of keys when requested.

The system introduces a data splitting function that is employed to ensure data availability while preserving the privacy of the data. Organisations that are complying with BlockTorrent will share the encrypted data they capture as well as the associated decryption key, which will be discussed in Section 3.2.2. The data is split and shared across all participants using hashed sharding to randomly distribute the chunks between participants, this is explained in more detail in Section 3.2.1. A table of contents for each file is generated as the file is chunked and distributed, this is referred to as the master table of that file. There is one master table per file distributed in BlockTorrent. Participants send acknowledgements when chunks are received which are added to the table for that file. Once all chunks have been distributed the register is encrypted and sent to the main chain along with the decryption keys for the file and the master table.

To access the stored data, an auditor must request access to the decryption key through a smart contract on the main chain. This access request is recorded and

emitted as an event to the entire network to ensure that auditors are only accessing the data when needed. Broadcasting this event is a deterrent to any unauthorised access by an auditor or even colluding participants. As the organisation has no influence on data access once it has been stored in BlockTorrent, data accessibility is guaranteed. As the data is stored in near real-time, the ability of an organisation to misrepresent the data is also lowered as it is harder to know in advance how you want to misrepresent information.

The system must support two main functions: *storing information* and *retrieving information*. We describe these functions below but first we need to explain two key concepts that BlockTorrent utilises, the Hashed Sharding and the key management mechanism.

3.2.1 Hashed Sharding

Hashed sharding [13] is the technique used by the admin node consortium to determine where each chunk is being sent.

First, each participant is given a range of hash values for which they are responsible. Then each chunk is hashed and sent to any peer that is responsible for the range that the digest falls into. To ensure accessibility, three separate hash functions are used, allowing the chunks to be replicated and stored by multiple peers. Using different hash functions allows each peer to only maintain one range of hash digests while still securely replicating the data and preventing network failure [32].

As each chunk is processed at the admin node consortium it is hashed by each node, using the pre-determined hash algorithms and then digests are compared. As long as the digests match, that chunk is distributed to every participant that is responsible for the range in which each digest falls. This concept is well used in the sharding of the Ethereum Network which has proven that sharding can be used to effectively grow distributed databases in a scalable manner.

3.2.2 Key Management

The key management system has the challenging role of generating, distributing and securely storing the key for the encrypted file. To ensure that privacy concerns are met this key needs to be shared privately. BlockTorrent uses Shamir's Secret

Sharing (SSS) technique [31] to first split up the key into distinct shards. SSS has the unique property that, if the key is split into n parts any k can recreate the key in its entirety. This is the core idea behind key management in BlockTorrent. Each key is split into n parts where n is less than the total number of participants and k is agreed upon by all participants beforehand. Then the key shards are distributed to participants randomly. How this is achieved is explained in Section 3.2.1.

The basic algorithm for SSS uses the concept of Lagrange interpolation theorem, specifically that k points on the polynomial uniquely determine a polynomial of degree less than or equal to k-1. For instance, 2 points are sufficient to define a line, 3 points are sufficient to define a parabola, 4 points to define a cubic curve and so on.

SSS has its strengths making it a good fit for many applications. These strengths include:

1. Secure: The method of creating the keys is based on a cryptographically secure process and is known to have information-theoretic security. That is, it is considered safe against adversaries with unlimited computing power and resources.

2. Minimal: The size of each share does not increase with the size of the secret being shared

3. Extensible: Shares can be added or deleted without affecting the existing shares

4. Dynamic: Security can be improved by periodically changing the polynomial used to generate the shares

5. Flexible: A different number of shares can be given to different participants so that important entities can unlock the secrets themselves while less important participants would have to collude to get the required number of shares

The main weakness of SSS is that whilst it can split up a secret into multiple shares, it does not have a mechanism for verifying that a specific share is part of a specific secret. Although SSS provides a mechanism to share a secret between multiple parties, there is still the challenge of verifying that each share is part of the same key. This is because the splitting of the original decryption key happens

on chain without any visibility or influence from participants. The shards are then automatically placed in on chain private storage for each participant meaning that the location of these shards is unknown. For this reason, the shards are used to build a Merkle Tree [33] this Merkle Tree is used to generate cryptographic proofs for each shard. Each proof is also stored on the main chain to be stored as part of the master table so that participants can verify that shards are from a particular key. Any participant who receives a shard can use the proof on the main chain to prove that the shard is valid and a part of the correct key. This process is similar to the Zero Knowledge Proof used in ZCash [34] except here, each participant acts as a prover for their own keys and a verifier for other participants' keys.

The secret keys are passed to the blockchain network using transient fields, which are ways of providing arguments to functions in chain code without them being recorded [35] and are unique to Hyperleder's Fabric Blockchain (HLF) Solution. The key shards are stored in the different organisations' private data collections. With this mechanism, no privacy-sensitive data is visible to the main chain, however, each participant is able to share their private keys with other participants securely. It also allows auditors to request k shares of a key and recreate the key without any input from the key owner.

The key management system revolves around a smart contract deployed on the main chain. This smart contract is agreed upon by participants and each participant is able to check the inputs and outputs. If a change to the smart contract is required it must be agreed upon again by all participants. This smart contract is responsible for receiving decryption keys, splitting them up and storing them in such a way that a subset of the key chunks can recreate the key. This is based on Shamir's Secret Sharing technique [31], but has been adapted to run on top of a permissioned blockchain. This mechanism can split a decryption key into n-shares but such that only k of those shares are needed to recreate it. K is known as the threshold and is decided by the consortium of participants.

Once the smart contract has split a key into shares, it must store these shares securely with different participants. For this, we make use of a function unique to HLF. Within HLF there is a world state and private data collection for each user. The world state is shared with all users and the private state can only be accessed by a single organisation. The key is split into k-shares, this is done on-chain via a smart contract and each unique share is stored in a participant's private data collection.

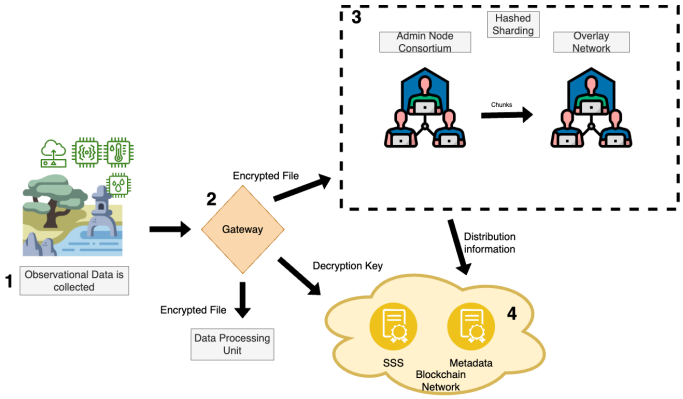

Figure 3.3: Transaction flow for the storing of data.

This smart contract also has access to each participant's private data collections, so no matter which peer is submitting the key shard, it can be securely stored in any of the participant's private data collections.

The last concern for this key management system is how the key is submitted in a transaction to this smart contract. According to the endorsement process of transactions in HLF, the key can be visible to those nodes at the time of endorsement. However, HLF provides a special field called the transient field, mentioned above, where users can submit data to smart contracts and have it concealed from endorsers. BlockTorrent uses the transient field in transactions to obfuscate the key data from other participants.

3.2.3 Storing Data

Fig. 3.3 shows the following steps for storing a new file in BlockTorrent:
Step 1: New sensor data is collected and sent to the private database.
Step 2: The new data is detected by the admin node of the organisation. The data is first encrypted and then chunked. The number of peers the chunk is sent to is determined by the total number of participating peers and is determined through hashed sharding as explained in Section 3.2.1.
Step 3a: The admin node consortium while splitting the file and determining the owner peers, stores a record of each chunk's hash, owner peers and timestamp which is sent to the main chain. Owner peers are the peers that were sent a copy of the chunk. A master table is created for each file and updated with the record of this

chunk. Each admin retains a copy of the master table until it is agreed upon and stored in the main chain.
Step 3b: The admin peer then sends each chunk to the list of determined peers. The message is in the following format:

$$M_{chunk} = (C^e|O|ts)$$

where C^e is the encrypted chunk, O is the owner of the file and ts is a timestamp.
Step 4: If this is the last chunk to be distributed, then the master table is completed, encrypted and submitted to the main chain. This information maintains the system integrity and has the following format:

$$M_{mastertable} = (H(MT)|MT|O|ts)$$

where $H(MT)$ is the hash digest of the master table, which is used as a unique identifier for each chunk, MT is the encrypted master table with the chunk distribution information in it, O is the owner of the file and ts is the timestamp of when the file was sent.
Step 5: The key is submitted to the key management smart contract on the main chain that is responsible for splitting and sharing that key between participants. This smart contract also builds the merkle tree and key share proofs.

As per the above steps, the decryption key and chunks of a file are all that is required to recreate a file and all of that information is stored in the master table for that file. Moreover, this key has been split into multiple parts and can be recreated by the auditor and a certain amount of participants (the exact amount is determined by how the key is split). The chunk distribution data is accessible to the auditor at all times.

Once a mastertable and key combination has been securely stored in the main chain, an index of the master tables is updated and stored publicly whilst the key is split and stored securely. These tables are considered a source of truth within the network and the BlockTorrent mechanism and are what an auditor will use when determining the location of chunks. Once a transaction is recorded on the main chain it is considered to be true and any participant found to be out of sync with the main chain, depending on the actions of the out-of-sync participant, they

are either penalised or ignored as a source of information. There are more severe punishments such as being excluded from the supply chain or having to pay higher costs as a result of not having accurate information, this is further explained in Section 3.2.4.

Algorithm 1 shows the pseudo code of what was implemented in the proof of concept system in regards to storing the data. It explains how an admin node receives sensor data, splits it up and fills in the master table.

Algorithm 1 Storing Data

```
data ← Sensor Data
MasterTable ← Array[]
ChunksToDistribute ← Array[]
index ← 0
while index < sizeof(data) do
    NextChunk ← data[index : ChunkSize]
    key_1, digest_1 ← SHA1(NextChunk)
    key_2, digest_2 ← SHA2(NextChunk)
    key_3, digest_3 ← SHA3(NextChunk)
    ChunksToDistribute ← (key_1, digest_1)
    ChunksToDistribute ← (key_2, digest_2)
    ChunksToDistribute ← (key_3, digest_3)
    index ← index + ChunkSize
end while
for Chunk in ChunksToDistribute do
    Peers ← HashedSharding(chunk.digest)
    MasterTable ← chunk.key
    for Peer in Peers do
        Peer ← chunk.digest
    end for
end for
MT_key, MT_digest ← SHA(MasterTable)
MainChain ← MT_key + MT_digest
```

3.2.4 File Retrieval

Fig. 3.4 shows the following steps for retrieving a file using BlockTorrent:
Step 1: The auditor requests the decryption key shares from the smart contract. This request can be validated via a vote of the admin node consortium. Once the

request is validated, the participants will submit a transaction indicating their key shard can be accessed by the auditor. When the auditor receives enough number of key shards they can recreate the decryption key.
Step 2: The auditor retrieves the master table from the main chain.
Step 3: The auditor combines the key shards and recreates the decryption key. This key can be used to decrypt the master table.
Step 4: For each chunk in the master table, the auditor looks up its location, requests the chunk from one of the owner peers. If one of the owner peers is unavailable or denying the request, another owner peer is selected until the chunk is received. The auditor can then check the hash of the chunk against the hash recorded in the master table to ensure the chunk is correct.
Step 5: The auditor uses the keys from the master table to decrypt each chunk, combines each chunk, computes the hash, and compares it with the hash stored in the main chain. If they match, the auditor has successfully recreated the file.

If any hash does not match its chunk, the auditor requests the chunk again until all chunks have been acquired and validated. If they cant find a chunk that matches then a dispute has arisen and a participant has been falsifying data somewhere along the line.

At no point is the organisation whose files are being audited required to participate, significantly diminishing their ability to lie and misconstrue information during the audit request. There is still the issue of centralisation of power at the admin node, where organisations could falsify data as it is being recorded. This can be minimised by having the captured data processed by the admin node consortium before it is processed through the data processing unit, maintained by each participant.

The functions described in Section 3.2.3 and 3.2.4, participants have access to mechanisms for sharing data with only those participants that they are engaged with. An auditor who resides on the main chain can use the data retrieval function to perform audits on participants. An auditor in BlockTorrent assumes two roles: (i) mediator for disputes between participants, and (ii) auditor for the network, either as part of a random inspection or as the result of a fault. The fault could be a dispute amongst participants or an end user (such as consumer or retailer) receiving a product that does not reach their satisfaction standard. As an authority, the auditor manually inspects the relevant products and analyses the relevant information stored

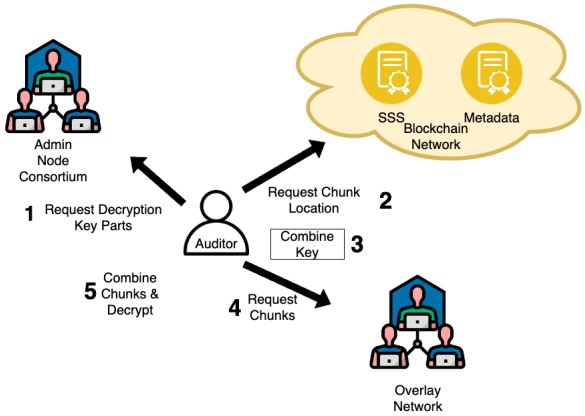

Figure 3.4: Transaction flow for file retrieving.

on the main chain. If any party is found to be in conflict with the main chain, they are penalised. The party at fault would be responsible for all costs related to the fault, including the fees and the cost of the audit. A financial penalty can also be applied by the admin node consortium if the party at fault financially profited from the fault. The penalty should be higher than any potential financial benefits gained from the conflict so that participants would be incentivised to share data honestly. BlockTorrent was also designed with a generalised architecture in mind so each of the components can be changed to accommodate different requirements. For instance, the underlying storage could be an already established distributed file storage system such as IPFS or Ethereum Swarm [20].

Algorithm 2 shows the pseudo code of what was implemented in the proof of concept system in regards to recreating data files that have been stored using BlockTorrent. It explains how An Auditor can request the decryption key shards from different peers on the main chain, then with a decrypted master table, they can request each chunk individually, recreate the file and decrypt it.

Algorithm 2 Retrieving Data

```
1: DecryptionKeyShards ← Array[]
2: No.Keys ← 0
3: PeerOwners ← MainChain.peers
4: while No.Keys < Key Threshold K do
5:     for Peer in Peer Owners do
6:         DecryptionKeyShards ← Peer.shard
7:         No.Keys ← No.Keys + 1
8:     end for
9: end while
10: for Shard in Decryption Key Shards do
11:     FinalKey ← FinalKey + Shard
12: end for
13: MasterTable ← MainChain(MasterTable)
14: ReceivedChunks ← Array[]
15: for Chunk in Master Table do
16:     for Peer in Peer Owners do
17:         NewChunk ← Peer.Chunk
18:         if Hash(NewChunk) == Chunk then
19:             ReceivedChunks ← NewChunk
20:             Break
21:         end if
22:     end for
23: end for
24: CompleteFile ← 0
25: for Chunk in ReceviedChunks do
26:     CompleteFile ← CompleteFile + Chunk
27: end for
28: DecryptedFile ← Decrypt(CompleteFile, FinalKey)
```

Table 3.1: How the design decisions of BlockTorrent align with the requirements of the system

Requirement	*Design Decision*
Ensure Data Availability	This is the main problem BlockTorrent attempts to solve. You could argue all decisions while developing BlockTorrent had this requirement in mind. However, the one aspect of BlockTorrent that could not be changed was that in order to guarantee data access it had to be shared and stored in multiple locations. Storing data in multiple places is decreases security and privacy as it increases the number of points of failure the system has to protect.
Ensure Privacy	The requirement of privacy is significant as no business or industry would adopt a system that could not protect its private and confidential information. For this reason, BlockTorrent attempts to protect all data stored within the system by encrypting it at the point of collection and having strict policies around who & when this data can be accessed
Ensure Security	Using blockchain technology provides a high level of security. Using a sophisticated consensus algorithm and ensuring that only identified participants join the network BlockTorrent is resistant to many classical network attacks. This is explained in detail with a per attack analysis in Table 4.1. One of the main reasons the decision to use blockchain was made is because of the security benefits that it provides.

Chapter 4

Evaluations

This chapter will explain the evaluations on a proof of concept system that was designed and built for this thesis. The implementation of this system will be explained and the evaluations and results will follow.

4.1 Implementation

The implementation has three interacting components. A detailed description of them are as follows:

Main Chain: BlockTorrent is implemented on Hyperledger Fabric (HLF) version 2.3[1], which is one of IBM's enterprise-level blockchains created for easy integration with business applications. HLF is a private blockchain that relies on a consortium to create a secure decentralised shared ledger. The participants in this consortium network are identified by predetermined certificate authorities, either agreed upon before or organised separately by each participant. HLF uses peer nodes to allow administrators and applications access to the ledger by exposing a set of API's for accessing certain parts of the ledger. HLF has two key features that BlockTorrent takes advantage of and were mentioned previously in Section 3.2. They are, (i) private data collections, and (ii) the transient field within transactions. Private data collections are used to give organisations the ability to store data within the HLF network but secure it so that only that organisation can access it. BlockTorrent uses these collections to store the key shares once a decryption key is submitted to the network. The transient field allows users to submit private information to

[1]https://www.hyperledger.org/projects/fabric

the blockchain without allowing validators to view this information. BlockTorrent uses the transient field to shield the decryption key from eavesdropping when it is submitted as part of the data storing mechanism. The last vital aspect of the implementation is the smart contracts that were developed and deployed. The smart contracts were developed in Go v1.11.2 and their main function is to accept and process transactions from participants. As referenced in Section 3.2 there are two smart contracts. The first implements Shamirs secret sharing algorithm and is responsible for splitting a key into different parts and storing them in participants' private data collections. HLF operates with an endorsement policy, which in essence, is a set of rules that governs how transactions are added to the ledger. An endorsement policy was developed that allows transactions from one organisation to store key shards in a distinct organisation's private data collection but not retrieve them. This is what allows BlockTorrent to guarantee key storage. The second smart contract accepts transactions that contain the metadata and master table including the peers that have stored a copy, hash digest and owner of the file. Depending on the application this second smart contract could be public within the consortium, as knowing where chunks are stored is not necessarily a security risk.

Overlay Network: The overlay network was written in Python 3.7. We made use of Python's native networking to simulate a P2P network as well as generate files to share. For encryption and hashed sharding processes, we used the SHA3-256, SHA-384 and SHA-512 algorithms. Each peer is setup to listen to two events. The first event is detecting new data in the private database and the second is listening for incoming packets from other peers. Similar to a BitTorrent network, a Distributed Hash Table is maintained for storing the address and names of each peer so that they can easily be found.

Private Database: This component can be any data storage solution that the private organisation deems necessary. The only requirement for use within BlockTorrent is that the data can be aggregated into files and that an admin node that can access the data is also a part of the admin node consortium. For this implementation, the database was just text files stored on the local computer. In theory, this can be any database or storage structure that suits an organisation's needs.

4.2 Evaluation

The evaluation has been divided into two sections, (i) a performance component, and (ii) a security analysis. The performance evaluation is to prove that the proof of concept would be capable in a real-world scenario. The security analysis is to add to the discussion of this thesis around what level of data sharing is reasonable and could be expected of organisations. The overlay and blockchain components were tested independently.

4.2.1 Performance Evaluation

We tested the performance of the main chain using Caliper[2] on a Linux server with a Intel(R) Xeon(R) W-2135 CPU with 64GB of memory. In order to evaluate the system performance for a realistic scenario, the test network included four participants each deploying an endorsing peer, a chaincode container and a regular peer. Each participant also has a process that is simulating the sensor devices by generating data files. There is also an ordering service running solo for the test network.

The overlay network was tested on an HP Pavilion-15-cc134tx with 16GB of memory and an Intel i7 processor. The time module in Python was used to calculate the time required for splitting, distributing, and retrieving files. We studied the variation in these three metrics as a function of changing the file size and the total number of chunks for each file. For testing purposes, we considered 100 files of sensor data being generated, split and distributed by six peers in the network simultaneously. We retrieve at least two files from each peer on the network each round. We tested with different file sizes (5,10,15,25,50 MB) of sensor data as well as a different number of chunks. A test with a particular number of chunks (10, 20, 40, 50, 60, 80, 100, 120, 140, 160, 180, 200) and a particular file size was carried out 100 times and averaged to minimise ant outlying errors. For example, one round of testing includes a 5MB sensor data split into 50 chunks and distributed among 6 peers to give us the splitting time and distribution time. Lastly, 10 files are retrieved on a particular peer to capture retrieval time. This process was executed 100 times for each pair of file size and number of chunks. So, in total, the network has been tested 5500 times to obtain consistent results and ensure that any outlier tests can

[2]https://www.hyperledger.org/projects/caliper

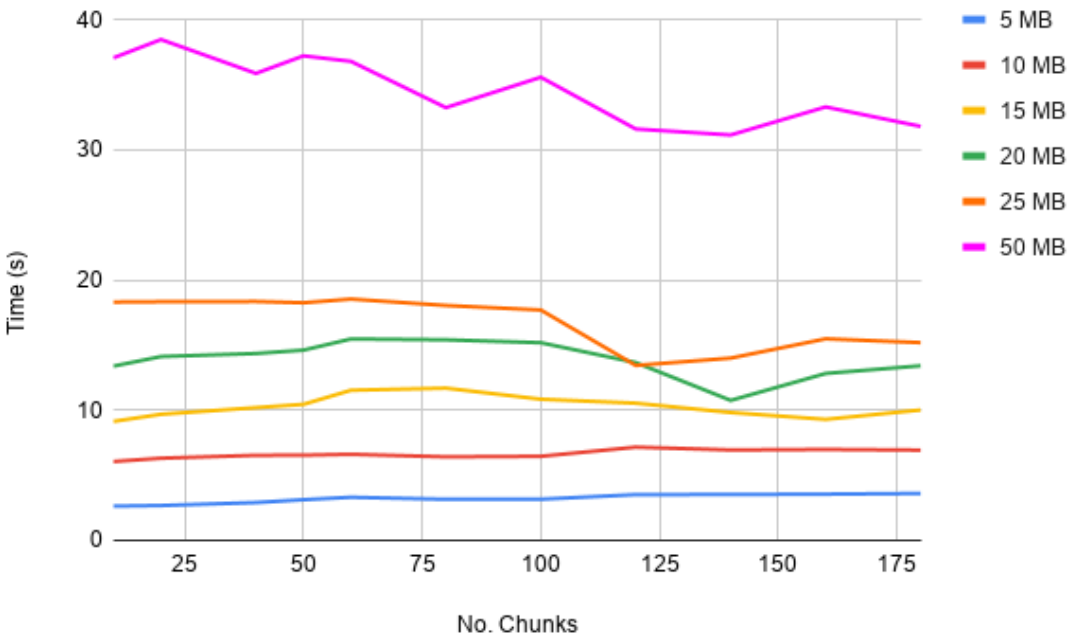

Figure 4.1: Time taken to split files into chunks

be eliminated.

Fig. 4.1 shows the results for splitting time with changing file sizes and number of chunks. For files smaller than 10MB the splitting time remains almost constant. On the other hand for larger file sizes (more than 10MB), the splitting time is higher and decreases slightly for larger number of chunks (110+). Splitting time scales with file size and not the number of chunks. This is because the larger the file size and number of chunks, the longer each chunk takes to slice, hash and encrypt, all of which increase the splitting time. Therefore, we should choose the number of chunks based on the size of the file. If the file is less than 10MB than the number of chunks can be reduced to optimise distribution time, as seen below. Therefore if the file is larger than 10MB then a larger amount of chunks should be selected, our results indicate a chunk number between 110 and 150 would be the fastest.

Fig. 4.2 shows the results for distributing file chunks to the peers. To evaluate the distribution time, we used an acknowledgement signal to indicate the end of the distribution process. Each peer sends an acknowledgement to the peer from which it received a chunk. The difference between the start of the distribution process and

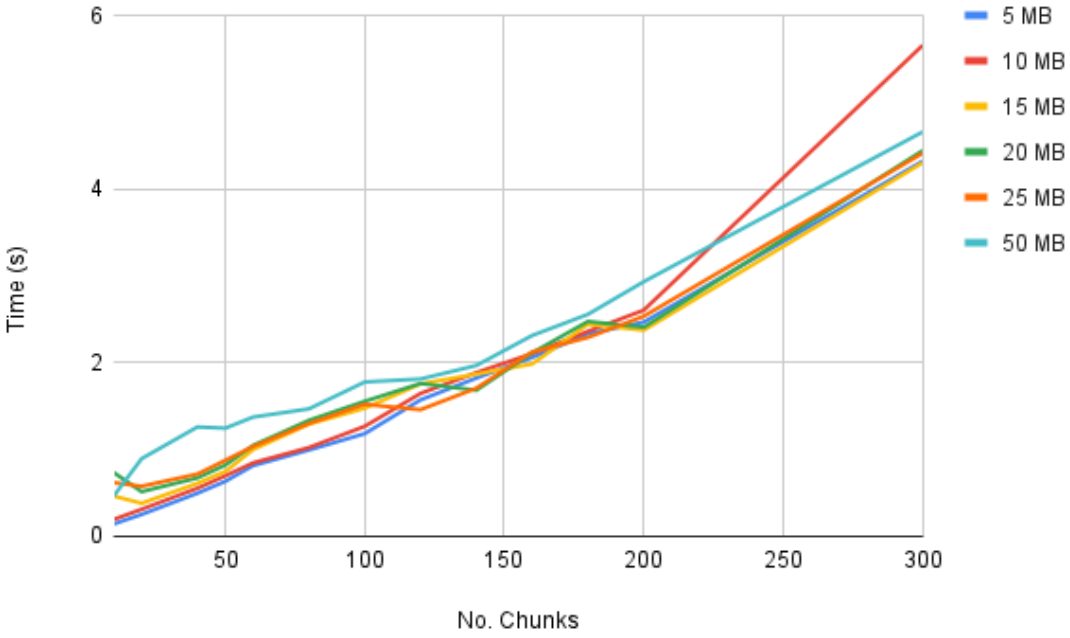

Figure 4.2: Time taken to distribute chunks across the network

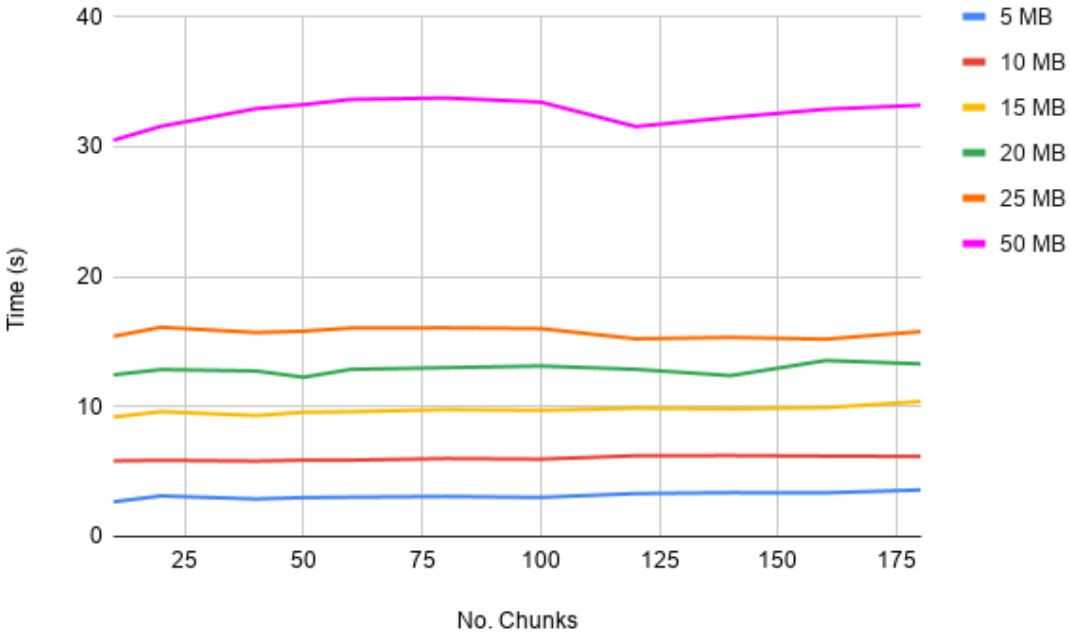

Figure 4.3: Time taken to retrieve chunks and recreate the file

the time at which the final chunk is received is the distribution time. For each file size, the distribution time increases with an increase in the number of chunks. With almost no effect on the file size, although if the file size was to become drastically large it would become an issue again. The increase in time for smaller file sizes is steeper than for the larger file sizes, this is obvious past 150 chunks. For smaller file sizes the effect of the number of chunks dominates distribution time, whilst for the larger file sizes the effect of the size of chunks dominates. Hence, if we want faster distribution in our network, a smaller number of chunks should be selected.

Fig. 4.3 shows the regeneration time which is equal to the difference between the time at which the retrieval of the first chunk starts and the time at which the complete file is regenerated and verified against the hash in the master table. The regeneration time remains almost constant, with respect to the number of chunks and increases with respect to file size. As the number of chunks increases, the size of each chunk requested reduces. Hence, the increasing number of chunks and the decreasing chunk size neutralise each other's effect. On the other hand, the regeneration time increased with the increase in the size of IoT data files. This is because with an increase in size, the amount of data to be put into the regenerated file increases.

It is to be noted that with an increase in the number of chunks, the size of each chunk reduces. These two changes have counter intuitive effects on the results, as the number of total chunks gets larger, the size of those individual chunks is smaller and faster to distribute across the network. These times would be affected by the total time it takes to split up a file into these chunks but the splitting time is constant for file sizes under 10MB.

For smaller file sizes the splitting time, distribution time and retrieval time are similar, however, for larger file sizes the splitting and retrieval time values are significantly higher than the distribution time. The file distribution was executed on a local network which would have a lower latency on average than a live network. The file sizes and number of chunks should be determined based on the application needs. Applications requiring faster distribution time can choose smaller number of chunks to reduce the impact of distribution time.

For benchmarking the blockchain layer, it was split it into two key parts, storing and splitting the private key and querying the splits. For storing we tested splitting the key up into 2,3,4 and 10 shards and then storing them in random participants'

Figure 4.4: Transaction throughput and latency of storing and splitting the private key on the blockchain.

private data collections and recorded the throughput and average latency for each transaction. The transactions were simulated using Hyperledger Caliper. Fig. 4.4 shows latency and throughput results for these tests. The system reaches saturation around 35 tps and we see a small drop off to around 30 tps between splitting the key into 2 shards and 10 shards. The latency increases with send rate as once the system is at saturation, higher send rates just create larger queues. The difference in throughput between splitting the key into 2 or 10 shards is small, indicating that the secret sharing implementation on-chain is a bottleneck in the system. This can be resolved by having participants aggregate files before encrypting and sharing them with BlockTorrent. This would reduce the number of keys that need to be submitted and split up on chain, alleviating some of the issues that come with scaling this system up.

Fig. 4.5 shows the transaction throughput and latency for querying shards of a private key. Note this is not the time taken to recombine the shards into the original key as that can be done off-chain and will not affect the blockchain performance. A query transaction will execute an access control process on the submitter to make

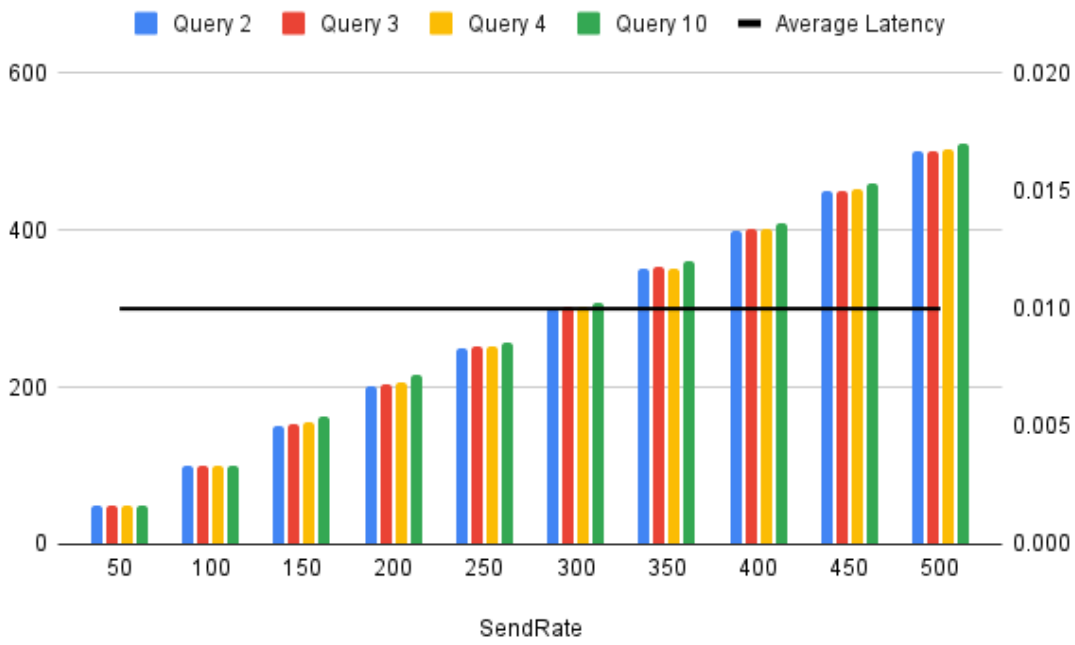

Figure 4.5: Transaction throughput and latency for querying a specific number of shards from a particular private key.

sure they are authorised and then will read the shard from the private data collection. Similar to the key storing benchmark, we tested querying 2,3,4 and 10 shards. The results show that querying shards is less computationally expensive and does not reach saturation until after 500 tps. This indicates that the system could handle a large number of queries and would be adequate to handle a larger-scale supply chain or similar system.

4.2.2 Security

This section explains how security was considered during the design and implementation of BlockTorrent. Blocktorrent is a data-transferring protocol and system architecture that exists exclusively in networks, suggesting that it would be susceptible to common network attacks as well as specific attacks targeting the protocol and its processes. BlockTorrent was designed as a decentralised protocol, that is, similar to bitcoin and other blockchain technologies the protocol is resistant to adversaries who understand the methods and processes underpinning the system.

The main reason that a security analysis was required is that BlockTorrent is intended to be used by participants that do not trust each other and have an inherent way to trust each other. BlockTorrent attempts to incentivise honesty and transparency by providing efficient trades between untrusted parties. In other words, BlockTorrent will help participants cut costs in exchange for submitting information in a distributed fashion. There is always the case that data provided by a participant could be privacy sensitive, thus protecting data security is an essential and vital feature. Unlike centralised servers, the data in a distributed system is required to be shared across multiple participants. Simply put, the more copies of data that is shared the more points of failure are created. To be completely sure that data was not being leaked BlockTorrent takes measures like splitting data into chunks and encrypting all chunks before they are transferred but that doesn't stop all attacks and it opens the protocol up to other attacks such as collusion or data falsification. For these reasons common client-server defences are not appropriate and other measures are taken to protect against these issues.

To combat some of these network attacks critical areas of BlockTorrent need to be identified. For the purpose of this analysis we assume that data transfer is efficient and complete, that is, if a chunk of data is sent then it is received. Middle-man attacks affect all networks and as BlockTorrent only transfers encrypted data there is no need to discuss middle-man attacks in this thesis other than packet sniffing which is discussed further on. With that assumption, a clear starting point for vital components in the system is the key management. The decryption keys being shared along with the encrypted files and encrypted master table implies that the most critical security risk is access to the keys. HLF allows for transient field parameters, which allows us to pass the keys to the blockchain without other peers seeing them. The keys are then split up using SSS on-chain, giving no participant any control over the process. Participants during the endorsement policy creation will decide in how many shards each key will be split into and how many shards are required to recreate the key. They will also agree on the random algorithm used to determine which shards are shared with which peers. The idea here is that if these restrictions are put in place in advance it is harder for a malicious actor to game the system or collude. In the case that participants are found to be acting maliciously or fraudulently the smart contracts can be updated. If the other participants agree on new security requirements because of changes to entities or components within

the system then these new changes can be reflected in smart contract updates and using HLF's update policy it would need to be approved via the endorsement policy setup at the network's inception.

As for accessing these key shards, an endorsement policy can be created that reflects the privacy and security concerns of the network. This endorsement policy can require a large proportion of participants to agree on access to any shards. This minimises the chance of collusion attacks as an adversary would need to compromise a large number of participants or get exceedingly lucky that the small number of compromised participants get all the shards to a specific master table decryption key. The adversaries would then also need the compromised master table and decryption key to be for the set of data that is actually useful for their attack. As an example, a group of participants could collude to share shards that are randomly assigned and transferred to them. But as this is a random process in the cases where the group receives every shard for a decryption key and master table there is the chance that this information is useless or it is too late for them to use this information maliciously.

Another threat is where an adversary launches a sniffing attack to try and determine how much data a particular participant is generating and potentially gain insights that would allow them to manipulate supply chain processes in their favour. To defend against this, BlockTorrent encrypts all files and tables regarding file distribution and then passes the key to the blockchain in a transient field. Network traffic can still be monitored, but no information about the key is leaked. Each shard is also the same size so the only information that is leaked is which participants are sharing more files than others. Based on the supply chain where BlockTorrent is implemented this could even be public knowledge. For example, a logistics company monitoring food supplies in a refrigerated truck might report temperature data every minute whereas a retail producer may only record data for when deliveries are accepted by them.

One of the last lines of defence in BlockTorrent is the audit system. Even though the main reason to implement BlockTorrent is to guarantee data availability for something like an audit, the system itself is designed to incentives honest and transparent users instead of relying on audits. This is mainly due to the fact that audits are costly and time-consuming and it is usually in the best interest of all participants to keep the supply chain operating at maximum efficiency. It is well documented

[1] that supply chain audits are a disruption to the general flow of goods and services. To defend against malicious participants who are suspected of attacking the network an audit can be called for. If a dispute is raised between two participants, then an authority already established on the main chain can act as a mediator using the main chain as a source of truth. Any participant detected with contradicting evidence on the main chain is penalised via a financial penalty or removed from the network. The goal of BlockTorrent is to not be used but in the case that an audit is occurring it can guarantee data availability without any interaction from the two disputing participants. Whilst this will not solve all disputes it will go a long way in helping supply chains with visibility into their own processes as well as incentivise all supply chain participants to act honestly as there can be serious repercussions for not complying.

4.2.3 Potential Attacks

This section outlines potential network attacks that could be executed on the BlockTorrent protocol. As the protocol itself is an integration of a secure technology stack, blockchain, and a data transfer protocol, BitTorrent, the security risks predominantly lie within the network environment and malicious users.

Table 4.1 summarises the identified attacks that could be launched against BlockTorrent and a brief description of how the attack could be executed. These are just a summary of potential attacks but depending on the network that BlockTorrent is deployed on it could be subjected to more attacks. This section will now explain the adverse effects these attacks have on the network and explain the measures designed in BlockTorrent to mitigate the damage or eliminate the attack completely.

Data Spoofing

Data spoofing is when an adversary or malicious user falsifies the data they are entering into the system. A peer or authority requesting data from this participant could receive misleading information that cannot be verified using BlockTorrent. The data is shared after it is encrypted and so the incorrect or inaccurate data would not be noticed until it was required to be looked at.

To counter this attack BlockTorrent forces participants to share encrypted data in near real-time. A malicious participant would need to know in advance what to modify the data with, which can be exceedingly difficult to accomplish consistently.

Table 4.1: Identified security attacks and proposed countermeasures using BlockTorrent.

Attack	*Description*
Data Spoofing[36]	A malicious admin node could alter the sensor data as it is being recorded and stored in the private network. The admin could then choose what to store in the network, allowing a participant to falsify information that would be used in a trade or an audit.
Sybil Attack[37]	A malicious node pretends to be multiple nodes on the network and trick other nodes into sending it more chunks than intended. The worst case is a single node controlling all the chunks of a file. This node then has control over the distribution of that file.
DoS/DDoS Attack[38]	Adversary floods the network with invalid transactions.
51% Attack[39]	Adversary takes control of more than 50 % of the admin nodes
Sniffing Attack[40]	Adversary seeks to analyse network traffic to obtain insights into participants' data.
Collusion Attack[41]	Adversaries can collude with one or more nodes to reveal confidential information.

If the adversary was lying about information that is used during trades then they would be found out once a buyer received items that did not match the main chain information. Otherwise, they will be discovered if an audit is requested. There are businesses where it would be quite easy to know in advance what data would need to be manipulated and how, this could be achieved by monitoring the data inputs and outputs for one year and then simulating what the "best" values would be for your business. In this case, simply encrypting the data would not be enough, however, solving this specific challenge is outside the scope of this research and is left as a direction for future work.

Sybil Attack

In a Sybil attack, the chunks being transferred are of an encrypted file so the adversary would only get information on the size of the file and chunks. BlockTorrent also identifies each user on entry to the network such that if a node wanted to im-

itate a node from another participant they would need to register as a node under that participant or gain access to that node's private keys to sign transactions as them. If a node's private key is compromised and they are aware, a new key can be created and re-registered in the network as a new participant whilst removing the old compromised key from the network.

DoS/DDoS Attack

In this attack, the network is slowed down to a point that valid transactions are rejected or dropped due to throttling. This is done by having a malicious node submit a substantial amount of invalid transactions causing the validators and admins on the node to waste compute time on validating pointless transactions. BlockTorrent identifies the participants on entry, so every node sending files is linked to a participant as explained in Chapter 3. Any node found to be generating large amounts of invalid transactions can be identified and have access denied or revoked as a response.

51% Attack

The 51% attack is when a single adversary or group of adversaries control more than 50% of the decision power in the network, allowing them to dictate what is stored. BlockTorrent requires votes to change the endorsement policy, any adversaries hoping to attack the internal configurations of BlockTorrent would need to submit all the corresponding transactions, and whilst they could vote and pass the changes with the majority of voting power, they would still have to do it on-chain which means any admin nodes paying attention would see it and raise the alarms.

Hyperledger Fabric can also be configured with custom consensus algorithms and rules allowing it to design a mechanism that makes the majority attack impossible. For example, a rule that transactions requesting file chunks must be validated by all participants removes the potential 51% attack completely. There is a range of BFT consensus algorithms that can be used with HLF, improving its resilience to this attack.

Sniffing Attack

The adversary can gain insights into other participants' data, potentially revealing company secrets. As mentioned in Section 3.2, each file is encrypted before being split and transferred, so only the amount of traffic will be visible to anyone on the network, and no private information is leaked. If the amount of information that is being generated and stored is sensitive you could set up nodes from that participant

to share a consistent amount of data at all times, using dummy data or empty transactions.

Collusion Attack

If the adversaries can obtain the decryption keys and the data file chunks they would be able to retrieve the files. However, as discussed above this would require luck and even then would not be consistent. Of course, if the malicious actors were able to compromise enough of the participants then the data would become available by having the compromised nodes share the chunks they receive. They could also carry out sophisticated attacks by injecting false information into the network through the compromised node(s).

Adversaries need access to both the keys and file chunks to execute a successful collusion attack. Even if an adversary gains access to the keys, they would need to collude with a large number of participants to request all chunks of a file. BlockTorrent can increase the number of chunks to increase the difficulty of this attack and change the random algorithm to increase security. In the case that the data is extremely sensitive you could use multiple randomness algorithms on rotation. Since all information is eventually stored in the master table it is not a requirement to use the same distribution technique for every file and chunk. There are also not many networks that can maintain stability and security when a large number of participants (greater than 50%) are colluding.

As with most systems, there is not much defence from insider attacks, that is if an admin of the system or a group of admins in the system decide to collude then the system would be compromised. In response to this BlockTorrent tries to incentivise honest actors within the network. This is done with a combination of punishments for malicious actors and rewards for honest actors. The rewards are mostly realised with more efficient transactions between network participants. Rather than being forced to trust other participants in the chain, you can have trust in the system itself and be provided with more trustworthy data that is secured and in the event of a dispute know that a third-party auditor will get access to the required information or at the very least to discover which participant is acting dishonestly.

Note that at the time of this thesis there were no similar enough solutions to do an exact comparison. Most solutions in existence today require a trusted third party to act as the sole arbiter and source of truth within the solution. As BlockTorrent removes the need for a single entity to be the authority comparisons to similar

solutions has been omitted.

Chapter 5

Challenges and Conclusion

This chapter will provide a discussion on the challenges and implications the BlockTorrent protocol generates. The chapter will first outline the challenges involved in the design of the system, with a specific focus on the trade-off between privacy and security. Lastly, this chapter will conclude the thesis and summarise the research question, main contributions, results and any future work considerations that were acknowledged along the way.

5.1 Challenges and Discussions

When designing BlockTorrent there was a clear lack of protocols that guaranteed data availability whilst also providing adequate levels of privacy. It became clear that there was not only a need for BlockTorrent but for a protocol that would actually incentivise participants of supply chains to implement and use it. A simple solution to the data availability problem is to bring in a trusted third-party participant that is responsible for storing any shared data among all participants. This participant would then hold all the information and effectively act as the source of truth much like the blockchain in BlockTorrent. However, through literature reviews and conversations with construction supply chain participants, this would be hard to implement and get off the ground as it creates a single point of failure for all participants and their private data. But more so there is no way to ensure that this trusted third party is actually trusted. This led to an investigation on how a protocol could ensure data availability amongst all participants in a distributed and secure manner so that there was no need to trust any single participant or system

but rather have trust in the network itself.

As mentioned above, the major trade-off for BlockTorrent is between privacy and security. The specific trade-off occurs in the key management and file distribution and retrieval components of BlockTorrent. To ensure the privacy and security of the system both the file and the key need to be distributed. This specifically avoids the issue of an owner losing the information or delaying providing the information when requested. BlockTorrent attempts to solve this issue by exploring the gap in privacy and availability in the context of the integration of blockchain and IoT. IoT devices can be equipped with state-of-the-art sensors, capable of capturing substantial amounts of accurate and new data. However, with the advancements of data capturing devices both in size and accuracy there is the concern that more and more information is collected there is private data also being collected. For instance, a resource logistics company that monitored the temperature of their refrigerated trucks might have temperature recordings taken every minute, however inside the metadata of those recordings will be location data, date & time data which could potentially reveal significant insights about the company that they may not want to be released. Therefore, a discussion about the trade-off between what information is considered private and what should be easily available is necessary.

BlockTorrent explores this trade-off through the key distribution challenge. This issue occurs whenever a decision about what secret should be kept by the owner of data to ensure privacy. If the owner keeps the master table or the decryption key a secret, then they can claim to have 'lost' the data and the files become unavailable even though they have been distributed on BlockTorrent. However, if a secret sharing algorithm is used, then the availability of the file is ensured. This creates a security risk as peers can collude and re-create the decryption key and the associated files without notifying any authority or admin on the network. The seriousness of the security risk depends on the security requirements of the application, a construction supply chain may not care whether one hundred panes of glass were delivered or two hundred as the total construction site will be using over one thousand. Whereas a food supply chain will care whether milk was transported under a certain temperature to ensure its freshness and overall quality of the product.

The other major consideration is security. A naive design choice would be to store the raw data on the main chain, but this would lead to potential information leaks. BlockTorrent employs a different approach where the complete raw data is stored

off-chain and only the transaction metadata, keys and master tables are stored on the main chain. The raw data is encrypted, chunked and then distributed on a P2P network that sits underneath the main chain. The P2P aspect of this network helps secure it as you would need to compromise a participant in order to even gain access to the network, and even then, packet sniffing or gaining access to the chunks would be pointless without the decryption keys.

The master tables contain data for tracing all the chunks of a file in the network. It can be used to reconstruct a whole file, thereby revealing the private information of a company. Although everyone in the network can see the transactions storing the master table, gaining access to the master table, is a little trickier as this information cannot be used without possessing the decryption key. This is the underlining security principle of BlockTorrent, data is divided and distributed in such a manner that even when an adversary gains access to certain information it is useless without the complete data.

In response to the differing needs of supply chains globally, BlockTorrent provides a mechanism where the level of privacy can be dynamically changed based on the application's needs. For instance, a construction supply chain may have lower privacy concerns than a specialised pharmaceutical supply chain. These supply chains will have different security regulations and access control requirements, which has inspired the design of BlockTorrent to be generalised so that it can be integrated with any supply chain. The degree of privacy present in BlockTorrent can be controlled on a technical level by varying the complexity of encryption used. A stricter endorsement policy can also be created and agreed upon at any time during the course that BlockTorrent operates. This can allow for more nodes to recreate files, and dictate who has control over the random distribution and who the audit authority is. Another way to increase privacy is that participants can agree to share less data. The goal of BlockTorrent is data availability however this may not be the need for all supply chains. In the case that only certain data is required to settle disputes and to keep operating costs down. The overall privacy of the network can also be determined at the abstract level where we design the role of the main chain in the system, which can either be a storage medium, a source of truth or a global world state that keeps track of file locations. If real-time access is not a concern, then storing whole encrypted chunks on the main chain could be a solution. This would require larger technical components and better network bandwidth but it is

assumed that supply chains with these requirements would also have the relevant and required technology to operate effectively. All of these privacy level changing levers paired with competent key-sharing mechanisms allows this system to ensure the accessibility of all files shared.

However, such a system would eventually result in an ever-increasing main chain as it is append-only and also increases search times when retrieving information from the main chain. It would also be susceptible to future brute-force attacks (such as post-quantum) that attempt to decrypt the encrypted files existing on the main chain. These problems can be alleviated by having time periods that are consolidated into new genesis blocks and moving the historic blocks to an off-network storage. This has happened in certain blockchains, where the whole chain is so large it is infeasible to expect users to download and maintain the entire copies. There are also mechanisms that have been developed to help with the processing of large blockchains such as Tree-chain [42].

Moving onto a discussion around the purpose and goal of BlockTorrent where it is compared to blockchain's most famous and popular use case, cryptocurrencies. A cryptocurrency system values consistency over availability and fault tolerance, due to the adverse effect of having monetary transactions go "missing". This extra focus on consistency is what has led blockchains to value security and privacy over speed and scalability. There is also no way to reverse or roll back incorrect transactions on a cryptocurrency so there are other security concerns that do not apply in a consortium blockchain like BlockTorrent. However, trying to store all participant data in BlockTorrent on the main chain is not feasible and will lead to the main chain becoming the bottleneck of the network. This is the motivation behind BlockTorrent using the main chain as a global world state and only storing the hashes and keys of data so that the network isn't throttled by storing all data on the main chain but can still validate transactions and verify those keys and master tables are being stored.

5.1.1 Future Work

With the challenges outlined above, there were some aspects of BlockTorrent that would have been nice to have implemented and evaluated but were not due to

time constraints and complexity of the challenge. Most of these revolve around the simulation and evaluation of BlockTorrent. The aspects that have been left as future work consist of the following:

1. Using AI to identify when misleading data is being submitted to the chain with fraudulent intent. This would allow BlockTorrent to catch lying participants in real time rather then relying on a dispute to be found and handled for the inaccurate data to be used and found.

2. Complete System simulation rather than simulating individual components separately.

3. Simulate results with different database mediums, both in the private data store and overlay network.

4. Integrate with real IoT sensors and data rather than generating data files.

5. Test with a live overlay network such as the internet or another active P2P network.

Although the results of this thesis in evaluating BlockTorrent are positive and prove that a system like BlockTorrent could exist in a real-world supply chain, there is still more to accomplish in proving that the system will have realised benefits and not just an extra complication in the world of complex supply chains.

Overall, BlockTorrent provides a platform that allows supply chain participants to perform their day-to-day tasks and share information with other participants simultaneously, without impacting their current supply chain processes. It ensures that public information is accessible to all the participants, and private information is safeguarded and shared with only verified participants or when access to that private information is required by authorities to conduct an audit or resolve disputes.

5.2 Conclusion

This thesis proposed BlockTorrent as a novel privacy-preserving data availability protocol aimed at solving the data availability problem within supply chains. BlockTorrent is designed with a generalised architecture so that it can be used with any

supply chain management data system. It revolves around the idea of splitting data into smaller chunks, encrypting those chunks and sharing the information required to retrieve and recreate the files in a secure and fair fashion.

Currently, too many solutions rely on centralised storage mediums and participant compliance to access data, leaving authorities without a guaranteed means for accessing data. By distributing chunks of data among supply chain participants, it is possible for honest participants to share their private data securely. Doing so in a supply chain environment means that data availability in the event of a dispute or audit is guaranteed. A solution like BlockTorrent helps avoid the issue of having every participant use a centralised storage solution and then requiring compliance for any relevant information. Having quick access to this data allows for faster exchanges, reduces supply chain costs and provides a more seamless process for data access. It also provides a mechanism of data retrieval where the original data owner has no involvement, leading to less collusion, data manipulation and less untrustworthy data in the system as a whole.

From the evaluation and simulations performed on BlockTorrent, the results indicate that a smaller number of chunks will improve file distribution time without severely impacting the security of the protocol. The speed and scalability of BlockTorrent indicate that it could be used in small to medium supply chains as is, and if it wanted to be applied to large-scale supply chains, the requirements of security and privacy can be adjusted to fit the needs and scalability of said supply chain. The security analysis also highlighted that there are few attacks that could penetrate BlockTorrents defences and even if they did the information gained would likely be useless in the case of a malicious goal.

The trade-off between privacy and security in a data-sharing protocol like BlockTorrent was also discussed and every effort in the design process went to balancing out this trade-off. Supply chain participants have a need to trust other entities on the network but have a distinct lack of mechanisms to do so. In a system where there is no inherent trust between the users, the trust needs to come from using the system itself. This is common in blockchain networks but with BlockTorrent this property is being transferred to supply chain logistics as well.

Overall, BlockTorrent creates an immutable digital history for all transactions that occur on a supply chain, which can be used as a single source of truth for determining the truth when disputes arise. It can be applied to new or existing

supply chains and will help lower costs, improve trust and ultimately eliminate the data availability problem if applied correctly.

Bibliography

[1] Microsoft, *How Blockchain will transform the modern supply chain*. Microsoft White Paper, 2018. [Online]. Available: https://azure.microsoft.com/mediahandler/files/resourcefiles/how-blockchain-will-transform-modern-supply-chain/how-blockchain-will-transform-modern-supply-chain.pdf.

[2] Y. K. Tomov, "Bitcoin: Evolution of blockchain technology", in *2019 IEEE XXVIII International Scientific Conference Electronics (ET)*, 2019, pp. 1–4. DOI: 10.1109/ET.2019.8878322.

[3] Microsoft, *How Blockchain will transform the modern supply chain*. Microsoft White Paper, 2018. [Online]. Available: https://azure.microsoft.com/mediahandler/files/resourcefiles/how-blockchain-will-transform-modern-supply-chain/how-blockchain-will-transform-modern-supply-chain.pdf.

[4] P. Catchlove, "Smart contracts: A new era of contract use", 2017. DOI: 10.2139/ssrn.3090226. [Online]. Available: https://ssrn.com/abstract=3090226.

[5] M. Asante, G. Epiphaniou, C. Maple, H. Al-Khateeb, M. Bottarelli, and K. Z. Ghafoor, "Distributed ledger technologies in supply chain security management: A comprehensive survey", *IEEE Transactions on Engineering Management*, 2021.

[6] S. Pal, T. Rabehaja, A. Hill, M. Hitchens, and V. Varadharajan, "On the integration of blockchain to the internet of things for enabling access right delegation", *IEEE Internet of Things Journal*, vol. 7, no. 4, pp. 2630–2639, 2019.

[7] S. Malik *et al.*, "ProductChain: Scalable blockchain framework to support provenance in supply chains", *NCA 2018 - 2018 IEEE 17th International Symposium on Network Computing and Applications*, 2018. DOI: 10.1109/NCA.2018.8548322.

[8] K. Korpela *et al.*, "Digital Supply Chain Transformation toward Blockchain Integration", *Proceedings of the 50th Hawaii International Conference on System Sciences (2017)*, 2017. DOI: 10.24251/hicss.2017.506.

[9] R. Monfared and S. Abeyratne, "Blockchain ready manufacturing supply chain using distributed ledger", *International Journal of Research in Engineering and Technology-IJRET*, no. 09, pp. 1–10, 2016. [Online]. Available: http://esatjournals.net/ijret/2016v05/i09/IJRET20160509001.pdfMetadataRecord:https://dspace.lboro.ac.uk/2134/22625.

[10] M. S. Ali *et al.*, "Applications of Blockchains in the Internet of Things: A Comprehensive Survey", *IEEE Communications Surveys Tutorials*, pp. 1676–1717, 2019, ISSN: 1553877X. DOI: 10.1109/COMST.2018.2886932.

[11] E. Elrom, *Hyperledger White Paper*. IBM Hyperledger, 2019, pp. 299–348. DOI: 10.1007/978-1-4842-4847-8{_}8.

[12] J. Pouwelse *et al.*, "The Bittorrent P2P file-sharing system: Measurements and analysis", *Lecture Notes in Computer Science*, vol. 3640 LNCS, pp. 205–216, 2005, ISSN: 03029743. DOI: 10.1007/11558989{_}19.

[13] N. Venkateswaran and S. Changder, "Simplified data partitioning in a consistent hashing based sharding implementation", in *IEEE Region 10 Annual International Conference, Proceedings/TENCON*, vol. 2017-December, Institute of Electrical and Electronics Engineers Inc., Dec. 2017, pp. 895–900, ISBN: 9781509011339. DOI: 10.1109/TENCON.2017.8227985.

[14] J. Pouwelse, P. Garbacki, D. Epema, and H. Sips, "The Bittorrent P2P file-sharing system: Measurements and analysis", *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 3640 LNCS, pp. 205–216, 2005, ISSN: 03029743. DOI: 10.1007/11558989{_}19.

[15] Y. Xiao, N. Zhang, W. Lou, and Y. T. Hou, "A survey of distributed consensus protocols for blockchain networks", *IEEE Communications Surveys & Tutorials*, vol. 22, no. 2, pp. 1432–1465, 2020. DOI: 10.1109/COMST.2020.2969706.

[16] H. Sukhwani, J. M. Martínez, X. Chang, K. S. Trivedi, and A. Rindos, "Performance modeling of pbft consensus process for permissioned blockchain network (hyperledger fabric)", in *2017 IEEE 36th Symposium on Reliable Distributed Systems (SRDS)*, 2017, pp. 253–255. DOI: 10.1109/SRDS.2017.36.

[17] X. Hao, L. Yu, L. Zhiqiang, L. Zhen, and G. Dawu, "Dynamic practical byzantine fault tolerance", *2018 IEEE Conference on Communications and Network Security, CNS 2018*, no. February, pp. 1–14, 2018. DOI: 10.1109/CNS.2018.8433150.

[18] "Blockchains and Their Applications - A critical review",

[19] Bittorrent Foundation, "BitTorrent (BTT) White Paper", *BitTorrent Official Website*, no. February, pp. 1–21, 2019. [Online]. Available: https://www.bittorrent.com/btt/btt-docs/BitTorrent_(BTT)_White_Paper_v0.8.7_Feb_2019.pdf.

[20] Swarm Team, "SWARM - Storage and Communication Infrastructure for a Self-Sovereign Digital Society", pp. 1–13, 2021.

[21] I. Baumgart and S. Mies, "IPFS Whitepaper", *Proceedings of the International Conference on Parallel and Distributed Systems - ICPADS*, vol. 2, no. Draft 3, 2007, ISSN: 15219097. DOI: 10.1109/ICPADS.2007.4447808.

[22] B. GmbH, *BigChainDB*, 2020. [Online]. Available: https://www.bigchaindb.com/.

[23] G. Ayoade, V. Karande, L. Khan, and K. Hamlen, "Decentralized iot data management using blockchain and trusted execution environment", Jul. 2018, pp. 15–22. DOI: 10.1109/IRI.2018.00011.

[24] L. Zhu, Y. Wu, K. Gai, and K.-K. R. Choo, "Controllable and trustworthy blockchain-based cloud data management", *Future Generation Computer Systems*, vol. 91, Sep. 2018. DOI: 10.1016/j.future.2018.09.019.

[25] H. Shafagh *et al.*, "Towards blockchain-based auditable storage and sharing of iot data", Nov. 2017, pp. 45–50, ISBN: 978-1-4503-5204-8. DOI: 10.1145/3140649.3140656.

[26] H. Huang, J. Lin, B. Zheng, Z. Zheng, and J. Bian, "When Blockchain Meets Distributed File Systems: An Overview, Challenges, and Open Issues", *IEEE Access*, vol. PP, p. 1, 2020. DOI: 10.1109/ACCESS.2020.2979881.

[27] H. T. Vo, A. Kundu, and M. K. Mohania, "Research directions in blockchain data management and analytics", in *EDBT*, 2018.

[28] R. K. Raman and L. R. Varshney, "Distributed storage meets secret sharing on the blockchain", *2018 Information Theory and Applications Workshop, ITA 2018*, Oct. 2018. DOI: 10.1109/ITA.2018.8503089.

[29] M. Fukumitsu, S. Hasegawa, J. Iwazaki, M. Sakai, and D. Takahashi, "A proposal of a secure P2P-type storage scheme by using the secret sharing and the blockchain", *Proceedings - International Conference on Advanced Information Networking and Applications, AINA*, pp. 803–810, May 2017. DOI: 10.1109/AINA.2017.11.

[30] M. Meulpolder, D. H. Epema, and H. J. Sips, "Replication in bandwidth-symmetric bittorrent networks", *IPDPS Miami 2008 - Proceedings of the 22nd IEEE International Parallel and Distributed Processing Symposium, Program and CD-ROM*, no. May, 2008. DOI: 10.1109/IPDPS.2008.4536194.

[31] A. Shamir, "How to Share a Secret", *Communications of the ACM*, vol. 22, no. 11, pp. 612–613, Nov. 1979, ISSN: 15577317. DOI: 10.1145/359168.359176.

[32] S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina, "The EigenTrust algorithm for reputation management in P2P networks", in *Proceedings of the 12th International Conference on World Wide Web, WWW 2003*, 2003, pp. 640–651, ISBN: 1581136803. DOI: 10.1145/775152.775242.

[33] D. Benarroch, M. Campanelli, D. Fiore, K. Gurkan, and D. Kolonelos, "Zero-Knowledge Proofs for Set Membership: Efficient, Succinct, Modular", *IACR Cryptology ePrint Archive*, no. 2019/1255, pp. 1–67, 2019. [Online]. Available: https://filecoin.io.

[34] E. Ben-Sasson, A. Chiesa, C. Garman, *et al.*, "Zerocash: Decentralized anonymous payments from bitcoin", *Proceedings - IEEE Symposium on Security and Privacy*, pp. 459–474, 2014, ISSN: 10816011. DOI: 10.1109/SP.2014.36.

[35] IBM, *Private Data in Hyperledger Fabric*, 2020. [Online]. Available: https://hyperledger-fabric.readthedocs.io/en/release-2.2/private-data/private-data.html.

[36] A. Hadid, N. Evans, S. Marcel, and J. Fierrez, "Biometrics systems under spoofing attack: An evaluation methodology and lessons learned", *IEEE Signal Processing Magazine*, vol. 32, no. 5, pp. 20–30, Sep. 2015, ISSN: 1558-0792. DOI: 10.1109/MSP.2015.2437652.

[37] J. R. Douceur, "The sybil attack", in *Peer-to-Peer Systems*, P. Druschel, F. Kaashoek, and A. Rowstron, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, pp. 251–260, ISBN: 978-3-540-45748-0.

[38] C. Douligeris and A. Mitrokotsa, "Ddos attacks and defense mechanisms: Classification and state-of-the-art", *Computer Networks*, vol. 44, no. 5, pp. 643–666, 2004, ISSN: 1389-1286. DOI: https://doi.org/10.1016/j.comnet.2003.10.003. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1389128603004250.

[39] C. Ye, G. Li, H. Cai, Y. Gu, and A. Fukuda, "Analysis of security in blockchain: Case study in 51%-attack detecting", in *2018 5th International Conference on Dependable Systems and Their Applications (DSA)*, Sep. 2018, pp. 15–24. DOI: 10.1109/DSA.2018.00015.

[40] P. Anu and S. Vimala, "A survey on sniffing attacks on computer networks", in *2017 International Conference on Intelligent Computing and Control (I2C2)*, Jun. 2017, pp. 1–5. DOI: 10.1109/I2C2.2017.8321914.

[41] M. Z. A. Bhuiyan and J. Wu, "Collusion attack detection in networked systems", in *2016 IEEE 14th Intl Conf on Dependable, Autonomic and Secure Computing, 14th Intl Conf on Pervasive Intelligence and Computing, 2nd Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress(DASC/PiCom/DataCom/CyberSciTech)*, Aug. 2016, pp. 286–293. DOI: 10.1109/DASC-PICom-DataCom-CyberSciTec.2016.67.

[42] A. Dorri and R. Jurdak, "Tree-chain : A fast lightweight consensus algorithm for iot applications", in *Proceedings of the 2020 IEEE 45th Conference on Local Computer Networks (LCN)*, ser. Proceedings - Conference on Local Computer Networks, LCN, H.-P. Tan, L. Khoukhi, and S. Oteafy, Eds., United States of America: Institute of Electrical and Electronics Engineers Inc., Nov. 2020, pp. 369–372. DOI: 10.1109/LCN48667.2020.9314831. [Online]. Available: https://eprints.qut.edu.au/208959/.