BlockTorrent: A Blockchain Enabled Privacy-Preserving Data Availability Protocol for Multi-stakeholder Scenarios

1st Ambrose Hill
School of Computer Science and Engineering
University of New South Wales
Brisbane, Australia
ambrose.hill@student.unsw.edu.au

2nd Shailesh Mishra
Department of Electrical Engineering
IIT Kharagpur
West Bengal, India
mshailesh0511@iitkgp.ac.in

3rd Atharv Singh Patlan
Department of Computer Science
IIT, Kanpur
Kanpur, India
atharvsp@iitk.ac.in

4th Ali Dorri
School of Computer Science
Queensland University of Technology
Brisbane, Australia
ali.dorri@qut.edu.au

5th Volkan Dedeoglu
Data61
CSIRO
Brisbane, Australia
volkan.dedeoglu@data61.csiro.au

6th Raja Jurdak
School of Computer Science
Queensland University of Technology
Brisbane, Australia
r.jurdak@qut.edu.au

7th Salil Kanhere
School of Computer Science and Engineering
University of New South Wales
Sydney, Australia
salil.kanhere@unsw.edu.au

2021 IEEE International Conference on Blockchain (Blockchain) | 978-1-6654-1760-0/21/$31.00 ©2021 IEEE | DOI: 10.1109/BLOCKCHAIN53845.2021.00024

***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 the event of 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 onto 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. BlockTorrent allows participants to securely share their data in near real-time with other participants without the risk of information leakage or allowing the falsification of data, whilst guaranteeing data availability for auditors. This is achieved using a novel combination of distributed storage and on-chain secret sharing. This paper provides an implementation and evaluation of BlockTorrent, highlighting its performance and a security discussion. Lastly, we provide a discussion on the privacy challenges that were considered when designing BlockTorrent.**

***Index Terms*—Blockchain, Supply Chain, Multi-Stakeholder, IoT, BitTorrent**

1. Introduction

There are many systems in the world today that heavily rely on participant compliance when it comes to data sharing and accessibility. The most common of these systems are supply chains [1]. Supply chains involve a group of organisations that are responsible for facilitating the transfer of goods and information from suppliers to customers. Recently, IoT sensor devices have been integrated with some supply chains allowing for the automatic tracking of items during their journey from supplier to the customer. Furthermore, the sensorisation of supply chains has given the stakeholders access to real-time data allowing 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 participants in supply chains poses privacy challenges. Sharing data can lead to information leaking e.g. contextual details such as quality or quantity of supplies. Thus, it is highly critical for the supply chain entities to keep IoT data secure and private. In response, many supply chain entities prefer centralised solutions for managing their digital data. This causes an issue when an audit is required. Each entity has the ability to manipulate or misrepresent the data that is supplied. In existing solutions, some auditors require hashes of the data to be shared at the point of capture, so that the data cannot be changed at a later date. Storing only the hash cannot solve this problem as the participant that provided the hash can claim that the data has been lost.

One potential technology to facilitate this sharing is Blockchain. Blockchain supports multi-stakeholder applications, such as a supply chain, with data immutability, audibility and access control [2]. Blockchain based supply chains have seen increased interest due to their improved scalability as seen in [3]–[6]. Although blockchain is a promising solution to address supply chain challenges [7], there is still the overarching problem that data stored on the blockchain can be accessed by anyone on the network. To solve this issue there

978-1-6654-1760-0/21/$31.00 ©2021 IEEE
DOI 10.1109/Blockchain53845.2021.00024

are a few avenues available to ensure the privacy of data on a blockchain, such as using a permissioned chain or using channels on Hyperledger [8]. Even these solutions come with challenges when trying to share data that only exists on a channel or permissioned chain, such as participants colluding to hide information or selectively storing partial information in the first place. Furthermore, supply chain entities could store their data off the chain and only submit hashes of the data to the blockchain, to prevent the leakage of privacy-sensitive information, similar to the solution discussed earlier. Instead of sharing hashes, some solutions require sharing encrypted data, however a malicious party could claim they lost the decryption key. While this approach ensures data privacy, it impacts the accessibility of the data. The main concern is that this data can still be made intentionally unavailable, such as when a supply chain entity lies about what data is being captured. There is thus a critical need for mechanisms that ensure data availability and integrity to authorised parties while preserving the privacy of data contributors.

This paper proposes BlockTorrent, a novel blockchain-based data management protocol, enhances data availability while protecting the privacy of the participants. BlockTorrent is an integration of BitTorrent [9], a Peer-To-Peer (P2P) file sharing protocol and Blockchain. Although BlockTorrent can be used by any application that requires data availability among multiple stakeholders, we focus on the supply chain context as a representative case study. To ensure data accessability, BlockTorrent distributes data among a set of peers in the blockchain. The participating nodes should be able to read the data for auditability; however, this compromises the user privacy, as outlined earlier. In BlockTorrent, data is split into multiple chunks and each chunk is sent to a randomly chosen node 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. Data splitting helps prevent unauthorised access, as even if an adversary is able to decrypt a chunk, they still need every other chunk to retrieve the 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 [9] protocol distributes files. We provide a solution on how the data will be maintained through its lifecycle. As the data is being dissected, there is a need 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 on the supply chain performance. The major contributions of this paper 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 is able to manipulate
- A discussion about performance and privacy trade-offs that came into consideration while designing the BlockTorrent protocol.
- 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.

The paper is structured as follows: Section 2 discusses the related works, while Section 3 provides an overview of the proposed privacy-preserving data sharing solution. Section 4 describes our implementation and presents results from our evaluations. Section 6 considers the main challenges faced when designing a distributed but secure database and provides a discussion on the design trade-offs discovered in this paper. Finally the paper concludes with Section 7.

2. Related Work

Table 1 gives a summary of the key features used in related literature and production systems. This section will provide some brief discussion on the related work in distributed data storage, data sharing, and data availability.

The authors in [17] 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 [18], 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 [14] 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 [15] 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

Related Work	Decentralised	Availability	Privacy	Integrity	Generalised Data Store
Ethereum Swarm [10]	✓	✓	×	✓	✓
IPFS [11]	✓	×	×	✓	✓
BTT [12]	✓	×	×	✓	✓
BigChainDB [13]	✓	✓	×	✓	×
IoTSmartContract [14]	×	×	✓	✓	✓
Controllable BC Data [15]	×	✓	✓	✓	×
Secure IoT [16]	✓	×	✓	×	×
BlockTorrent	✓	✓	✓	✓	✓

TABLE 1: Summary of BlockTorrent's comparison to related work.

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.

In [12], the authors propose BitTorrent Token (BTT), a crypto token that is attached to the BitTorrent network. It 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 accessability.

In [16], 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 [19] the authors propose using Shamir's sharing technique in combination with a partitioned blockchain to improve data integrity and privacy. 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 [20] propose a secure online storage system that splits up the storage of information and meta data. 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.

So far, distributed storage approaches have been unable to strike a balance between protecting 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.

3. BlockTorrent: A Privacy-Preserving Data Availability Protocol

This section 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 is incentivised to share supply chain related data but in a way that it is secured from non-authorised parties. This incentive is two fold: *firstly*, BlockTorrent uses penalties for non-compliance, that can be financial or in the extreme case result in removal from the system, furthering the incentive for honest participant behaviour. *Secondly*, a participant honestly complying with BlockTorrent will be able to provide reliable evidence that shows they are adhering to legal industry regulations. 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.

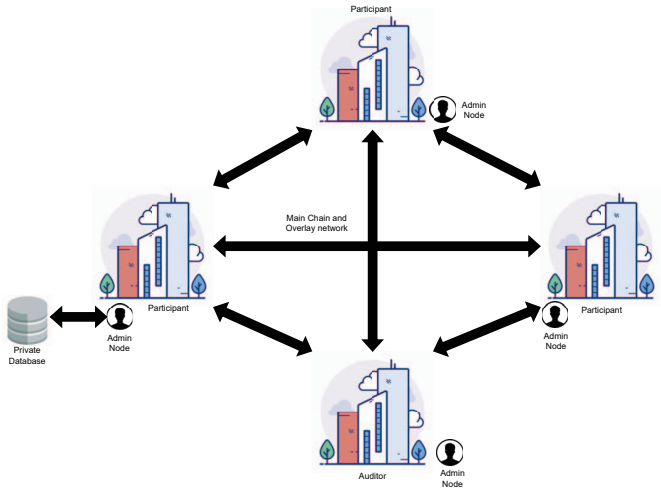


Figure 1: A high level overview of the proposed framework.

First, we will introduce the network layers and their interactions within system architecture. BlockTorrent relies on three key networks set up in parallel in order to reduce bottlenecks in network traffic. Next, we will outline the functionality that the proposed system offers and describe how it solves the data sharing challenges explained in the previous sections.

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 or state governments.
- *Admin Node:* 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 dispute resolution 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 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. 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, particularly when supply chains have multiple paths and processes involving a subset of participants from the main chain. 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. 1 displays the basic setup for a participating organisation.

Main Chain: Serves as the interface between organisations and BlockTorrent. Each participant of 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. 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.

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 [21] and is explained in Section 3.2.1.

3.2. System Functionality

The framework 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. Participants send acknowledgments 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 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 for 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.

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

3.2.1. Hashed Sharding. Hashed sharding [21] 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 against network failure [22].

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.

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 [23] 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.

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. For this reason, the key is split on chain and the shards are used to build a Merkle Tree [24]. The Merkle Tree is used to generate cryptographic proofs for each shard. Each proof is also sent to the main chain to be stored 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 [25] except here, each participant acts as a prover for their own keys and a verifier for other participant's keys.

The secret keys are passed to the blockchain network using transient fields, which are ways of providing arguments to functions in chaincode without them being recorded [26]. 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 yet 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.

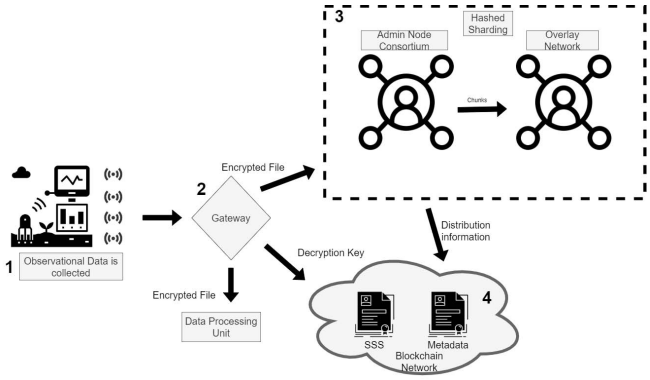


Figure 2: Transaction flow for the storing of data.

3.2.3. Storing Data. Fig. 2 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 is all that is required to recreate a 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.

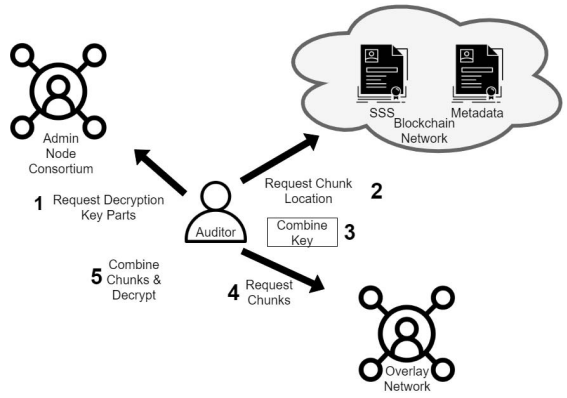


Figure 3: Transaction flow for file retrieving.

3.2.4. File Retrieval. Fig. 3 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 combines each chunk, computes the hash, and compares it with the hash stored in the main chain. If they match, the auditor uses the key from Step 3 to decrypt the file. If any hash does not match its chunk, the auditor requests the chunk again until all chunks have been acquired and validated.

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.

With 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: (1) mediator for disputes between participants, and (2) 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 (e.g. consumer or retailer) receiving a product that is not up to their satisfaction. As an authority, the auditor manually inspects the relevant products and analyses the relevant information stored 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 framework in mind so each of the components can be changed to accomodate different requirements. For instance, the underlying storage could be an already established distributed file storage system such as IPFS or Ethereum Swarm.

4. Implementation

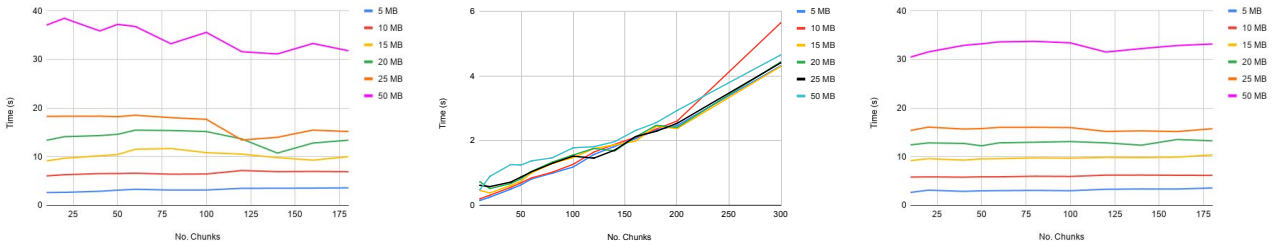
In this section, we will explain the implementation and evaluation of BlockTorrent. Section 4.1 will explain what technologies we used to implement our test network. Section 5 will provide an overview of our results, highlighting our key findings and lastly, Section 5.2 will provide security analysis of BlockTorrent against some of the common network attacks.

4.1. Implementation

The implementation has three interacting components. They 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. 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 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 key aspect of the implementation is the smart contracts that were developed and deployed. We developed smart contracts in Go v1.11.2. that accept and process transactions from participants. As referenced in Section 3.2 there are two smart contracts. The first implements SSS and is responsible

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



(a) Splitting Time (b) Distribution Time (c) Retrieval Time

Figure 4: Time taken to split distribute and retrieve files over the overlay network.

for splitting a key into different parts and storing them in participants' private data collections. An endorsement policy was developed that allows transactions from one organisation to store key shards in a distinct organisations private data collection but not retrieve them. The second chaincode accepts transactions that contain the metadata including the peers that have stored a copy, hash digest and owner of the file the chunk. A second smart contract was developed for the key management process as described in Section 3.2.2.

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 to listen 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 our implementation, we assume each participant is managing their own private database.

5. Evaluation

We have split the evaluation into a performance component and a security analysis. The overlay and blockchain components were tested independently.

5.1. 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 62GB 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.

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

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 number of chunks. For testing purposes, we considered 10 files of sensor data being generated, split and distributed by six peers in the network simultaneously. We retrieve at least one file 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 10 times and averaged to minimise error. 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, 5 files are retrieved on a particular peer to capture retrieval time. This process was executed 10 times for each pair of file size and number of chunks.

Fig. 4a 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.

Fig. 4b 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. For each file size, the distribution time increases with increase in 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. Hence, if we want faster distribution in our network, a smaller number of chunks should be selected.

Fig. 4c 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.

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 bench marking the blockchain layer we 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' private data collections and recorded the throughput and average latency for each transaction. The transactions were simulated using Hyperledger Caliper. Fig. 5 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 on the system. This can be resolved with 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. 6 shows the transaction throughout 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 sure they are authorised and then will read the shard from the private data collection. Similar to the key storing bench mark, 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 handling a larger scale supply chain or similar system.

5.2. Security

Data provided by a participant could be privacy sensitive, thus protecting data security is essential. Unlike centralised servers, the data in a distributed framework is required to be shared across multiple participants. Hence, common client-server defences may not be appropriate.

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 will decide in advance how many shards each key will be split into and how many shards are required to recreate the key. The chaincode can be updated if the participants agree on new security requirements because of changes to entities or components within the system.

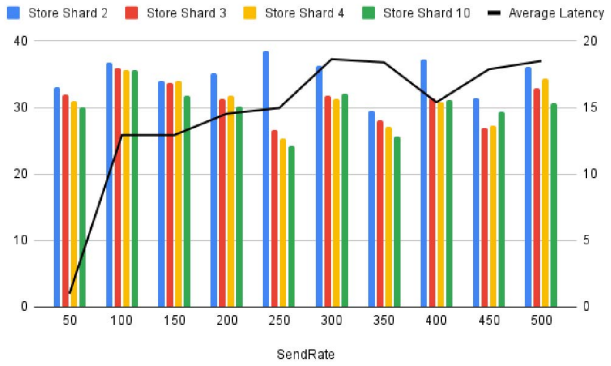


Figure 5: Transaction throughput and latency of storing and splitting the private key on the blockchain.

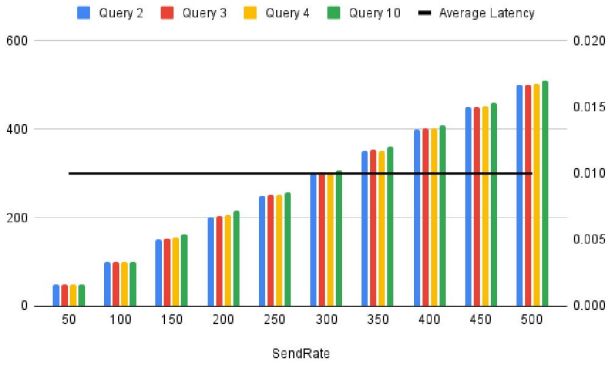


Figure 6: Transaction throughput and latency for querying a specific number of shards from a particular private key.

As for accessing these key shards, an endorsement policy can be used that requires 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. An adversary could launch 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.

BlockTorrent also uses an audit system to defend against malicious participants. 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.

Table 2 summarises the identified attacks that could be launched against BlockTorrent and how it defends against them.

TABLE 2: Identified security attacks and proposed countermeasures using BlockTorrent.

Attack	*Description*	*Adverse Effects*	*Possible Countermeasures*
Data Spoofing [27]	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.	Adversary can falsify the data. A peer or authority requesting data from this participant could receive misleading information that can be verified using BlockTorrent.	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 to which can be exceedingly difficult to accomplish consistently. 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.
Sybil Attack [28]	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.	Adversary can gain access to every chunk of a file.	The chunks 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 imitate a node from another participant they would need to register as a node under that participant.
DoS/DDoS Attack [29]	Adversary floods the network with invalid transactions.	The network is slowed down to a point that valid transactions are rejected or dropped due to throttling.	BlockTorrent identifies the participants on entry, so every node sending files is linked to a participant as explained in Section 3. Any node found to be generating large amounts of invalid transactions can be identified and have access denied or revoked as a response.
Sniffing Attack [30]	Adversary seeks to analyse network traffic to obtain insights into participants' data.	Adversary can gain insights into other participant's 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, no private information is leaked.
Collusion Attack [31]	Adversaries can collude with one or more nodes to reveal confidential information.	If the adversaries can obtain the decryption keys and the data file chunks they would be able to retrieve the files.	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.

6. Challenges and Discussions

The major trade-off for BlockTorrent is between privacy and security, in relation to key and file distribution. Both the file and the key need to be distributed to avoid the issue of an owner 'losing' the information. BlockTorrent attempts to solve this issue by exploring the gap in privacy and availability in the context of integration of blockchain and IoT. IoT devices can be equipped with state of the art sensors, capable of capturing substantial amounts of private data. 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 he 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 collaborate and create the decryption key and the associated files.

Security is also a major consideration for BlockTorrent. 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 raw data is stored off-chain and only the transaction metadata is stored on the main chain. A master table contains data for tracing all the chunks of a file in the network. It can be used to reconstruct a whole file, thereby revealing private information of a company. Although everyone in the network can access the master table, this information cannot be used without possessing the decryption key.

BlockTorrent provides a mechanism where the level of privacy can be dynamically changed based on the application 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 us to keep the design of BlockTorrent 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. To increase privacy, participants can agree to share less data. This 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 or a 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. Paired with competent key sharing mechanisms, this system ensures the accessibility of the files.

7. Conclusion

We have proposed BlockTorrent as a novel privacy-preserving data availability protocol that can be used with any supply chain management data. Currently, 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. Having quick access to this data allows for faster exchanges, reduces supply chain costs and provides a more seamless process for data access. Our results indicate that smaller number of chunks will improve file distribution time without severely impacting the security of the protocol. 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 faults.

References

[1] 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.

[2] 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.

[3] S. Malik, S. S. Kanhere, and R. Jurdak, "ProductChain: Scalable blockchain framework to support provenance in supply chains," *NCA 2018 - 2018 IEEE 17th International Symposium on Network Computing and Applications*, 2018.

[4] K. Korpela, J. Hallikas, and T. Dahlberg, "Digital Supply Chain Transformation toward Blockchain Integration," in *Proceedings of the 50th Hawaii International Conference on System Sciences (2017)*, Hawaii International Conference on System Sciences, 2017.

[5] 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.

[6] M. S. Ali, M. Vecchio, M. Pincheira, K. Dolui, F. Antonelli, and M. H. Rehmani, "Applications of Blockchains in the Internet of Things: A Comprehensive Survey," 2019.

[7] Microsoft, *How Blockchain will transform the modern supply chain*. Microsoft White Paper, 2018.

[8] E. Elrom, *Hyperledger White Paper*. IBM Hyperledger, 2019.

[9] 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.

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

[11] I. Baumgart and S. Mies, "IPFS Whitepaper," *Proceedings of the International Conference on Parallel and Distributed Systems - ICPADS*, vol. 2, no. Draft 3, 2007.

[12] Bittorrent Foundation, "BitTorrent (BTT) White Paper," *BitTorrent Official Website*, no. February, pp. 1–21, 2019.

[13] B. GmbH, "BigChainDB," 2020.

[14] G. Ayoade *et al.*, "Decentralized iot data management using blockchain and trusted execution environment," in *2018 IEEE International Conference on Information Reuse and Integration (IRI)*, pp. 15–22, 07 2018.

[15] L. Zhu *et al.*, "Controllable and trustworthy blockchain-based cloud data management," *Future Generation Computer Systems*, vol. 91, 09 2018.

[16] H. Shafagh, L. Burkhalter, A. Hithnawi, and S. Duquennoy, "Towards blockchain-based auditable storage and sharing of iot data," in *Proceedings of the 2017 on Cloud Computing Security Workshop*, CCSW '17, (New York, NY, USA), p. 45–50, Association for Computing Machinery, 2017.

[17] 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.

[18] H. T. Vo *et al.*, "Research directions in blockchain data management and analytics," in *EDBT*, 2018.

[19] R. K. Raman and L. R. Varshney, "Distributed storage meets secret sharing on the blockchain," *2018 Information Theory and Applications Workshop, ITA 2018*, 10 2018.

[20] 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, 5 2017.

[21] 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, pp. 895–900, Institute of Electrical and Electronics Engineers Inc., 12 2017.

[22] 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*, pp. 640–651, 2003.

[23] A. Shamir, "How to Share a Secret," *Communications of the ACM*, vol. 22, pp. 612–613, 11 1979.

[24] 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.

[25] E. Ben-Sasson, A. Chiesa, C. Garman, M. Green, I. Miers, E. Tromer, and M. Virza, "Zerocash: Decentralized anonymous payments from bitcoin," *Proceedings - IEEE Symposium on Security and Privacy*, pp. 459–474, 2014.

[26] IBM, "Private Data in Hyperledger Fabric," 2020.

[27] A. Hadid *et al.*, "Biometrics systems under spoofing attack: An evaluation methodology and lessons learned," *IEEE Signal Processing Magazine*, vol. 32, pp. 20–30, Sep. 2015.

[28] J. R. Douceur, "The sybil attack," *Springer Berlin Heidelberg*, pp. 251–260, 2002.

[29] 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.

[30] P. Anu and S. Vimala, "A survey on sniffing attacks on computer networks," in *2017 International Conference on Intelligent Computing and Control (I2C2)*, pp. 1–5, June 2017.

[31] 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*, pp. 286–293, Aug 2016.