BlockIPFS - Blockchain-Enabled Interplanetary File System for Forensic and Trusted Data Traceability

Conference Paper · July 2019

DOI: 10.1109/Blockchain.2019.00012

4 authors, including:

Reza M. Parizi
Kennesaw State University, Atlanta, USA
150 PUBLICATIONS **1,969** CITATIONS
SEE PROFILE

Qi Zhang
IBM Research
67 PUBLICATIONS **656** CITATIONS
SEE PROFILE

Kim-Kwang Raymond Choo
University of Texas at San Antonio
974 PUBLICATIONS **20,677** CITATIONS
SEE PROFILE

BlockIPFS - Blockchain-enabled Interplanetary File System for Forensic and Trusted Data Traceability

Emmanuel Nyaletey
College of Computing and Software Engineering
Kennesaw State University
GA, USA
enyalete@students.kennesaw.edu

Reza M. Parizi
College of Computing and Software Engineering
Kennesaw State University
GA, USA
rparizi1@kennesaw.edu

Qi Zhang
IBM Thomas J. Watson Research
Yorktown Heights NY, USA
q.zhang@ibm.com

Kim-Kwang Raymond Choo
Department of Information Systems and Cyber Security
The University of Texas at San Antonio, TX, USA
raymond.choo@fulbrightmail.org

***Abstract*—The Interplanetary File System (IPFS) is a distributed file system that seeks to decentralize the web and to make it faster and more efficient. It incorporates well-known technologies, including BitTorrent and Git, to create a swarm of computing systems that share information. Since its introduction in 2016, IPFS has seen great improvements and adoption from both individuals and enterprise organizations. Its distributed network allows users to share files and information across the globe. IPFS works well with large files that may consume or require large bandwidth to upload and/or download over the Internet. The rapid adoption of this distributed file system is in part because IPFS is designed to operate on top of different protocols, such as FTP and HTTP. However, there are underpinning concerns relating to security and access control, for example lack of traceability on how the files are accessed. The aim of this paper is to complement IPFS with blockchain technology, by proposing a new approach (BlockIPFS) to create a clear audit trail. BlockIPFS allows us to achieve improved trustworthiness of the data and authorship protection, and provide a clear route to trace back all activities associated with a given file using blockchain as a service.**

Keywords— Blockchains, Blockchain as a Service, Smart Contracts, Hyperledger Fabric, IPFS, Distributed File Systems.

I. INTRODUCTION

The InterPlanetary File System (IPFS) is a peer-to-peer distributed file system that seeks to connect all computing devices with the same system of files [1]. The distributed network, incorporating technologies such as BitTorrent and Git, achieves high throughput and allows users to share files and information efficiently across the network. IPFS network is ideal for sharing large files that may consume or require large bandwidth to upload and/or download over the Internet, and it has been designed to operate on top of different protocols. The central idea that developers built IPFS on is to model all data as part of the same Merkle DAG [1]. However, the popularity and effectiveness of IPFS as a distributed file system also create security and access control concerns. In a distributed network such as IPFS, when an object is uploaded to the network, anyone who has access to the hash address of the file can access its content [1]. Although this is a desirable feature for a distributed file system, the user who uploads a file cannot control access to the uploaded file once the hash address has been shared.

Currently when a file is uploaded to IPFS, the hash is shared with peers and those peers could access the file using the hash address. Once uploaded, the file owner will no longer have control over the file to which who else the hash will be shared with. In such case, peers can share the hash with anyone, and those users could also access the file. From a security perspective, this issue could lead to serious consequences and violation of IP and accessibility, and many fundamental questions remain unresolved. For example, how can we ensure that only those who we share the hash with can access the file or how can we know who else accessed the file? How do we ensure that ownership of content on IPFS can be protected? If someone downloads a file on IPFS, say a research manuscript, makes some changes to that file and re-uploads the file as their original work, how can IPFS protect the rights of the legitimate owner of the original file? How can we ensure that there is a trail of access events on IPFS for easy audit and verification in case of dispute?

The main focus of this work is to address the traceability problem so that all activities concerning a specific object on a distributed file system can be traced and access controlled. Even on a private IPFS network, users may wish to limit access of a file they uploaded to the network. They may wish that only certain members of the private network should have access to the file. Imagine in an organization, management may want to limit access to certain files by lower level management or other employees and make the same file accessible to higher level management. In the current IPFS, there is no way for an organization to create such a rule for accessing files on their private IPFS network. If a file meant to be shared among only high-level management is shared on the network, any member of the organization's IPFS network with access to the hash of the file can simply access it. That being said, the IPFS provides no traceability capabilities to record and audit access to files on the network.

Blockchain is a decentralized data management platform that provides immutability [2]; hence, it is a good fit to support file traceability metadata in a distributed file system such as IPFS. There are generally two categories of blockchains: permissionless and permissioned. A permissionless blockchain (e.g., Ethereum, Bitcoin) is open to the public, and every transaction is to be validated by every or majority of participants [7]. While only the authenticated users can join a permissioned blockchain (e.g., Hyperledger Fabric), validation in this type of blockchain is performed by only pre-selected nodes. Thus, it usually achieves higher performance than public blockchains.

978-1-7281-4693-5/19/$31.00 ©2019 IEEE
DOI 10.1109/Blockchain.2019.00012

In this paper, we propose a new approach (hereafter referred to as BlockIPFS) that attempts to enhance IPFS with Hyperledger Fabric to provide a trail, which can be used to audit and safeguard authorship of the files and be used as evidence for both forensic investigation and dispute resolution. In the event that someone takes the file off IPFS and illegally shares it with others, our approach can help to tell who has downloaded the file; thus, narrowing down the scope of investigation. As demonstrated in experimental results, by integrating our approach within IPFS, we create a more secure file sharing platform with a clear audit trail to improve the trustworthiness of the data, as well as of authorship protection. The proposed approach provides a clear route to trace back all activities associated with a given file.

The rest of the paper is organized as follows: Section II presents the relevant background on IPFS and Hyperledger Fabric and explains the design choices. Section III presents our proposed solution (BlockIPFS) and its inner workings. Section IV provides the evaluation and results. Section V discusses the implications and limitations. Finally, Section VI gives the conclusion and future work.

II. Background on IPFS and Hyperledger Fabric

A. Interplanetary File System (IPFS)

The Interplanetary File System (IPFS) is a peer-to-peer hyper media protocol designed to make the internet faster, safer, and more open [1]. In a P2P network such as IPFS, if one node is down, other nodes in the network can serve needed files. Files are content addressed rather than location addressed on IPFS [1]. When a file is added to the network, IPFS creates a multi-hash address of the file based on its content and the node ID ensuring that no two files share the same multi-hash. The node ID is the cryptographic hash of the node's public key [1]. To allow other nodes to access the file, the multi-hash can be published on the network or shared among nodes. IPFS looks for files through the Content Identifier (CID) instead of searching by the location of the file [1]. This means that any node with the hash of the content can search for the specific file on the network and any node that has a matching file will serve it. As a result, content on the network can be served so long as there is a node that can serve it. The P2P network is therefore robust and can withstand server attacks. Since content can be downloaded from the network, nodes can access local content offline. IPFS eliminates content duplication by ensuring that files with the same content are stored only once [1]. This is in respect to files uploaded by a specific node. For instance, if node A adds a file myfile.txt that contains "Hello world", IPFS creates only one instance of this file on the network even if the user of node A adds myfile.txt multiple times.

IPFS can be used to share files on a private network or on the internet through the IPFS gateway. Most users of IPFS use it to share large files since IPFS uses local hosting which reduces the need for bandwidth which is usually associated with sharing large files on the internet. Others have started using IPFS to host websites. When a file is uploaded to IPFS, all peers with the hash address can access the content, download and view content locally or they can view content from the author's node [4]. The greatest advantage of using IPFS in this way is that nodes become individual servers that can serve the content to others. For the IPFS network to go offline, all the nodes on the network must be down simultaneously, which is very unlikely.

B. Why Hyperledger Fabric

The Hyperledger Fabric (www.hyperledger.org) is part of a larger project known as the Hyperledger. Hyperledger Fabric is a permissioned blockchain technology unlike permissionless blockchains such as Ethereum and Bitcoin [13]. It is a private blockchain technology in that members of the blockchain network are known to each other and members are permissioned to join. On permissionless blockchains, membership is open to the public and any one can join and perform transactions on the network [8].

Hyperledger Fabric [16] is a modular blockchain framework which acts as a foundation for developing blockchain-based products, solutions, and applications using plug-and-play components that are aimed for use within a private enterprise environment [6]. In the following subsections, we present the specific reasons why Hyperledger Fabric is suited for solving the stated problems with IPFS in this research:

1) Performance

This is probably the most important feature of Hyperledger Fabric that makes it suitable for this research in compared to open blockchains. As Fabric is permissioned, all nodes that participate in the network have an identity, as provided by a modular membership provider (MSP) [10]. Therefore, there is no mining process, which can be both resource and time consuming in transaction flow [12]. This is critical to achieve minimal performance overhead when combining IPFS with blockchain.

2) Dynamic Channel Management

Hyperledger Fabric introduces the concept of channels, which can be dynamically managed within a single blockchain network. Participants in the same channel share the same set of ledger data, which cannot be accessed from outside of this channel [11]. Therefore, by using Fabric underneath, it allows IPFS users to flexibly form groups and securely share files within a group.

3) Modularity & Extendability

One of the main characteristics of Hyperledger Fabric that differentiates it from other blockchain technologies such as Ethereum, R3 Corda, is its modularity. Fabric intends to provide a modular and extendable architecture that can be employed in various environments [6]. The modularity of Hyperledger Fabric makes it suitable to be integrated with other technologies such as IPFS and enables us to integrate the needed modules with IPFS to extend the capabilities of IPFS.

4) Plug & Play API

Hyperledger Fabric provides SDKs that can be used to interact with the blockchain. Fabric has SDKs in Node.js and Go [5]. This capability enables IPFS to interact efficiently with Fabric. Through the capabilities provided by the SDK we can efficiently develop a client to query the blockchain and provide audit data.

5) Node Heterogeneity

Within Hyperledger Fabric, nodes are differentiated based on whether they are clients, peers, orderers, or certificate

authorities [9]. While IFPS also consists of multiple nodes, there is no strict mapping between the nodes in Hyperledger Fabric and that in IPFS. In BlockIPFS, for example, a node can be an IPFS node only, or it can take multiple roles to be an IPFS node as well as a Hyperledger Fabric peer node.

III. BLOCKIPFS: PROPOSED SOLUTION

In this section, we introduce our proposed solution that enhances the security of content on IPFS by providing traceability, access control, and authentication capabilities to the private IPFS networks.

A. Architecture of BlockIPFS

IPFS is commonly used as a storage platform for data sharing. It has the advantages of high availability and good performance, but lacks the capability of tracing access and authentication, which makes it difficult to investigate unauthorized access and authorship. In this design, a blockchain is integrated in IPFS. File operations such as adding or accessing a file generates metadata that is logged on a Hyperledger Fabric blockchain. The blockchain is responsible for storing and managing the metadata of the file. Figure 1 shows an overview of the architecture of BlockIPFS, where the user can perform all traceability operations by querying the fabric ledger through the client.

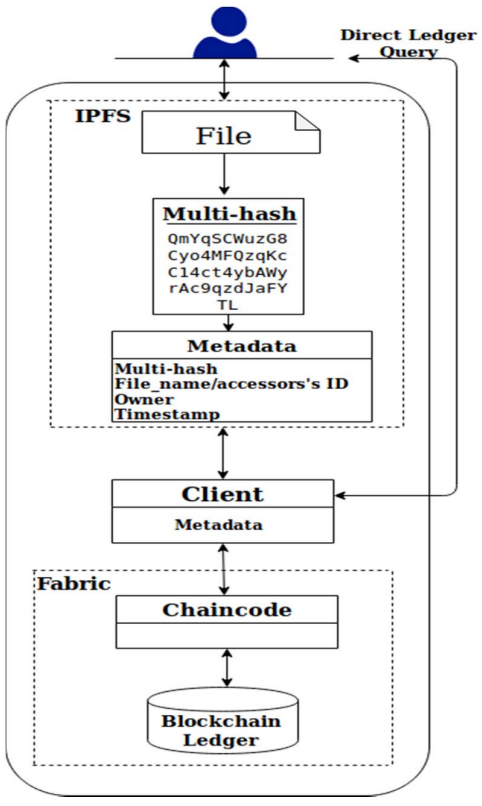

Fig. 1. BlockIPFS architecture showing the basic flow of file upload and access from IPFS to the integrated Fabric Blockchain.

In BlockIPFS, the user interacts with IPFS and can perform all the operations available to him/her. In this implementation, when a user accesses file on other nodes, no records are pushed to their local blockchain. However, access logs are recorded in the file owner's BlockIPFS. The user can retrieve metadata from their local BlockIPFS to trace activities related to a specific file and or all files they have added to IPFS.

It worth noting that the Hyperledger Fabric blockchain in BlockIFPS only stores the metadata of the files for tracing purposes, while the files themselves are still managed by IFPS. In other words, the blockchain is almost transparent to the users when they read/write files, and any file access control mechanisms in IFPS will be inherited by BlockIPFS. The users only query the blockchain to exam the metadata of a file, and such metadata is recorded on the blockchain ledger in an encrypted manner so that only the owner of the file or the granted users will be able to read.

As discussed above, BlockIPFS takes the advantage of channels in Hyperledger Fabric to enable flexible and secure file sharing. Similarly, the files themselves are encrypted and managed by the IPFS, while the keys to decrypt the files are stored in the specific channels on the blockchain.

B. Design

The goal of BlockIPFS is to provide a trusted, easy to use, and high-performance distributed file system by combining the blockchain and IPFS. In BlockIPFS, the blockchain is responsible for managing the metadata of the file system and the blockchain ledger only stores the transactions-related metadata. This will not incur high storage or network overhead since we assume amount of metadata is much smaller than the raw data. In this case, the blockchain ledger can have a full historical and trusted records of the files, such as who created the file and when, who accessed the file and how, who can access what file. For efficient searching, ledger data will be indexed. Raw data is stored in the storage managed by the file system (in this case IPFS), but not the blockchain. The blockchain also provides an added means of sharing multi-hash of content. Once a file is added to IFPS and the multi-hash added to the blockchain, all members of the blockchain will have instance access to the hash. IPFS operations and their related metadata storage in the blockchain are synchronized however, to ensure continuity, when the blockchain operations fail, the user is alerted of the failure and given the options to continue without blockchain operations or abort the entire process.

Next, we describe the main components of BlockIPFS and it functions in reaching our traceability and authentication goals.

1) User (owner)

The user is the actor that adds and/or accesses a file on the IPFS network. The user can perform all file operations such as add, cat, and get on files on the network. The operation can be performed as follows:

```
$ ipfs add myfile.txt
```

To carry out normal file operations stated above, the user does not directly interact with the underlying Hyperledger Fabric. In fact, the user does not directly interact with the embedded blockchain unless they perform traceability functions through the client.

2) File

A file in our approach is any object that is added to IPFS. The file can be of any types, e.g., image, audio, video, etc. There is no limitation to the size of a file added to the network. Once a file is added to the network, it is processed through the various stages of the proposed system (see Figure 1). IPFS stores the file on local storage and hosts the file's hash address on the network [1].

3) Hash

IPFS use cryptographic hashing functions to create fingerprints. IPFS implements SHA256 hashing algorithm that is tamper-proof and ensures security and authenticity of files added to the network [3]. The content of files is represented with multi-hash format and base58 encoding. Basically, it takes the raw content and runs that data through hash function to produce a digest. This digest is guaranteed to be cryptographically unique to the contents of the file, and that file only. If someone changed that file by even one bit, the hash would become completely different [1].

4) IPFS network

The proposed system is implemented on a private IPFS network which comprises of nodes that are known to each other. In this implementation, we created a private IPFS (NodeJS version) network.

5) Client

The client is the application that we used to interact with the blockchain to retrieve information about files. In this implementation, we used the JS API to interact with the ledger. After a file hash address is obtained, IPFS invokes the client to connect to the fabric network and execute the chaincode either to add data and or retrieve data.

6) Chaincode

The chaincode is the 'smart contract' [22] that runs on the peers and creates transactions [5]. In this implementation, the chaincode contains methods that create a file record that is added to the ledger. It also contains methods for querying the ledger for a specific or multiple file hash addresses that belong to one user. It also contains methods for retrieving file access data.

7) Ledger

The ledger is the immutable record of all transactions in the Hyperledger Fabric network. Due to its immutability, the ledger is the perfect tool that ensures that files added to the network are secured and their authenticity can be easily verified.

8) File Upload

In the proposed solution, the user initializes IPFS and adds a file to the network using IPFS add command. IPFS obtains the file's hash address by running it through a SHA256 algorithm which generates 256-bit hash address. After obtaining the hash address of the file, two things happen. First, IPFS stores the original file in local storage and adds the hash address to its network. The hash can then be published on the private network by the user (author) or shared with specific nodes (other users) on the network. Second, IPFS invokes the Hyperledger Fabric client which accepts the hash address of the file, the name of the owner (in this case, the user who adds the file to IPFS), the current date and time.

The fabric client invokes the chaincode and adds the record to the ledger. This process follows standard Hyperledger Fabric process for adding a record to the ledger. If this process fails, the user is alerted and is given the option to continue or cancel. Otherwise, the user is alerted of successful transaction.

9) File Access

To access a file, the user searches the file on the network using its hash address as follows:

```
$ ipfs cat
/ipfs/QmYwAPJzv5CZsnA625s3Xf2nemtYgPpHdWEz79
ojWnPbdG

$ ipfs get
/ipfs/QmYwAPJzv5CZsnA625s3Xf2nemtYgPpHdWEz79
ojWnPbdG
```

A file can be accessed using the 'cat' command (as above), which reads file content, and the 'get' method (as used above) downloads the file. When either of these above lines is ran, two main things happen. First, IPFS locates the file that corresponds with this multi-hash and returns the content of the file . Second, IPFS invokes the Fabric client which in turn invokes the chaincode on the blockchain and submits an access record to the ledger. The metadata added to the ledger includes *the hash of the file being accessed*, *the public address of the node accessing the file*, *owner of the file*, and *timestamp*.

IV. Evaluation and Results

To evaluate BlockIPFS, we developed 3 metrics that enable us to measure the performance against a regular IPFS network (serving as the comparative baseline in the experiment) and a typical Hyperledger Fabric blockchain. The metrics include *performance*, *traceability*, and *traceability access*. We implemented the proposed solution using three computers. The first is an Intel core i5 at 3.4 GHz, 4GB of RAM running Microsoft Windows 10 64-bit version on 500GB hard drive. The second is an Ubuntu Linux 18.4 64-bit version with an Intel Core i5 CPU at 1.6 GHz with 4GB RAM with a 250 GB hard drive. The third is an Arch Linux 64 bit with Intel Core i3 CPU at 1.6 GHz with 4GB RAM with a 120GB hard drive. We installed on each of these systems their respective operating systems, versions of Docker and Docker-Compose [19], Nodejs [18], Golang [17]. We also installed the JavaScript version of IPFS (js-ipfs) [20] on each of the three machines. We followed the instructions on the official Hyperledger Fabric read-the-docs website [21] to install and properly instantiate required files and docker images.

On each machine, we spawned a local IPFS node and obtained their public IDs. Using the Ubuntu Linux 18.4 system as the main system, we created a private IPFS network and enrolled the other two systems onto the network. These nodes become part of the IPFS swarm. All of these computers were on the same WIFI network. In this experiment, only the main system's IPFS network is embedded with the Fabric blockchain. Also, the main system is the only node that uploads files to its BlockIPFS network, and the other two nodes are used to access the files. We selected files in 5 sizes ranging from 500MB to 2500MB in the evaluation to demonstrate the efficiency of BlockIFPS. Although other file sizes could have be chosen, the

ones used in this paper are examples to produce representative results.

A. Performance

In this experiment, we attempt to provide a seamless experience to the IPFS user so that they can use IPFS without the added blockchain interference. But adding a blockchain to IPFS introduces an overhead that consumes computing resource and time. Hence, we measured the performance difference in IPFS functions such as adding and accessing a file with and without the integrated blockchain to identify any performance differences. We used IPFS without an embedded blockchain on the Windows 10 machine to perform upload and access operations. Using IPFS on this machine, we uploaded 5 files and collected the time it took to complete each upload. We then used the Arch Linux machine to access the added files using IPFS without an embedded blockchain and recorded the time it took to complete each access. The results are shown in Figure 2.

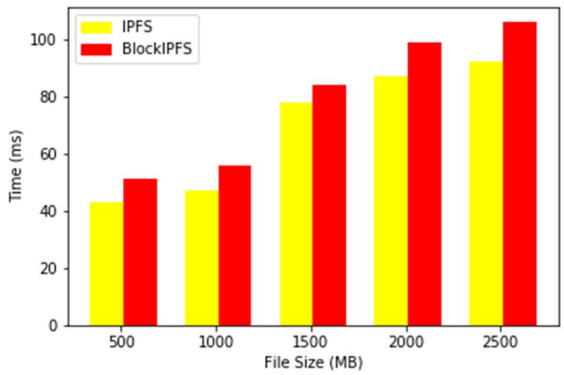

Fig. 2. Comparing elapsed time of file upload transactions between IPFS and BlockIPFS.

We then repeated the same process on the Ubuntu machine running our proposed solution. We uploaded the same 5 files as above, accessed them using the Windows 10 computer and the machine running Arch Linux and recorded the time it took to complete each transaction. Figure 3 shows our results.

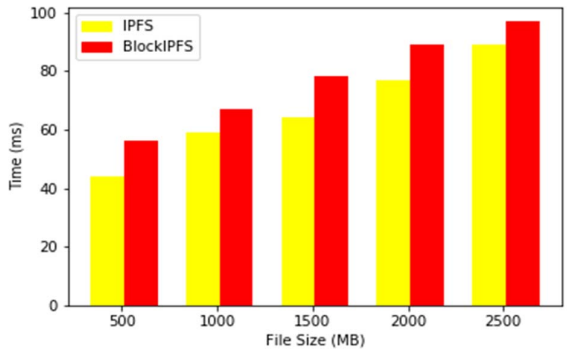

Fig. 3. Comparing elapsed time of file read transactions between IPFS and BlockIPFS.

In Figure 4, we show the results of download operation using the Arch Linux machine. The results show that the added blockchain incurs extra CPU processing and time, making it a little slower than regular IPFS. Naturally, we expect that it takes more time to process larger files on both our solution and IPFS. But we noted in our experiment that since the blockchain transactions are dependent on the completion of IPFS processes, the extra time consumed to add larger files to IPFS may affect the overall performance of our solution. We assume that the size of metadata recorded on the blockchain does not change irrespective of the size of the file being added to IPFS. It must be noted that our transaction times may be affected by the speed of our local network and/or the capabilities of our machine running.

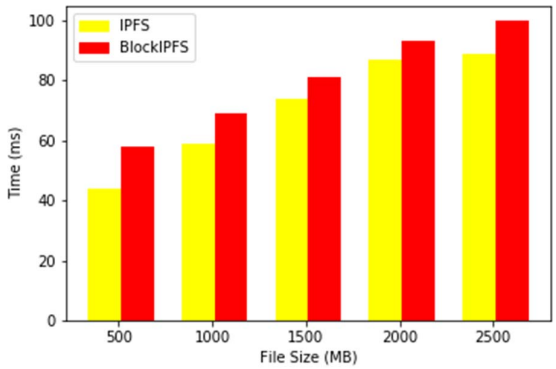

Fig. 4. Comparing elapsed time of file download transactions between IPFS and BlockIPFS

As mentioned in previous sections, IPFS serves as the storage and the fabric ledger logs metadata of each file and access activities. This ensures that large files are stored on IPFS and only metadata which are relatively very small get recorded on the fabric ledger. Figure 5 shows how the size of the fabric ledger grows in relation to files added to the IPFS network. We must point out that the average size of metadata recorded on the ledger is around 119 bytes for each activity performed on the overlaying IPFS network. Consequently, the ledger size grows but at a much slower rate and remains very stable even when the size of files added to the IPFS network drastically increases.

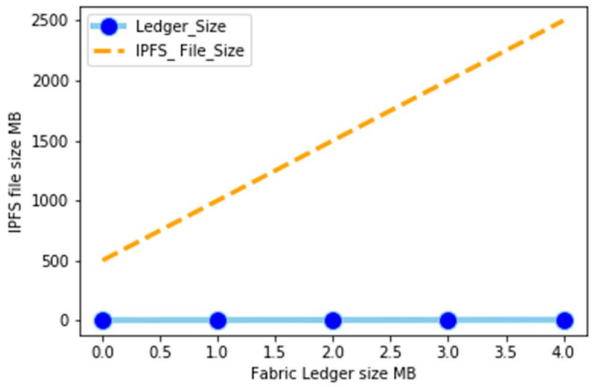

Fig. 5. Comparing the size of files on BlockIPFS and the size of file metadata on blockchain ledger.

B. *Traceability*

One important outcome we intend to achieve with the proposed solution is traceability. Using the Ubuntu machine running BlockIPFS, we uploaded 5 files of different sizes. Metadata of each file uploaded was recorded on the blockchain ledger. After this process, we were able to query the blockchain and retrieved records of each upload transaction as shown in Figure 6. We also accessed the uploaded files using both the machines running Arch Linux and Windows 10. The access transaction records metadata on the blockchain of the main Ubuntu machine. We then queried the blockchain to retrieve access records on the ledger as demonstrated in Figure 6. Figure 6 shows query results of all access transactions of files shown in Figure 6. Both query results show that files uploaded to BlockIPFS can be traced for all upload and especially access activity on the specific file. By obtaining this information, we have proved that files added to BlockIPFS can be reliably traced back to the original owner providing file history of all access activity for a specific file or multiple files.


```
"File1":{"hash":"QmRyCymXw8LKY3oAkMPJfmuzqCyz2LSThUKFAR1pUFvQxU",
"name":"my_file1",
"owner":"Earth",
"time":"2019-02-13 12:21:18"},

"File2":{"hash":"QmUzCbRpN9fCkMMG3qRopCemVEAVNSmEyo2i39QJCEfEgb",
"name":"my_file2",
"owner":"Earth",
"time":"2019-02-13 12:27:16"},

"File3":{"hash":"QmQ2pqNJtSN4e4HbKR7h5At17GYzXqPGAgFCuzaTrFhEv2",
"name":"my_file3",
"owner":"Earth",
"time":"2019-02-13 12:29:34"},

"File4":{"hash":"QmTJxeb9o8pNUn3fa8JvACVf3yx8WVSPHjh4iabXtbhBH2",
"name":"my_file4",
"owner":"Earth",
"time":"2019-02-13 12:31:48"},

"File5":{"hash":"QmR6qUj7cmNBBNttHCgsLumbw2P58DUYRgWbgd6sCiRbtR",
"name":"my_file5",
"owner":"Earth",
"time":"2019-02-13 17:06:29"}
```


Fig. 6. On-chain metadata for files newly added to BlockIPFS.

Figure 6 shows the results of a query of the ledger of all the files added to BlockIPFS. The results are shown as a series of key-pair values. Each record on the ledger has a key which is used to identify each transaction. For example, the first key in this screenshot is File1. Following each key is the actual record. Our record includes the multi-hash of the file, the name of the file, the owner's name, and the timestamp.

Also, Figure 7 shows the query results of all access activity on the files we uploaded to BlockIPFS. As stated above, each record has a key that identifies it. The key is followed by the record which contains the multi-hash of the file accessed, the owner's name, the public ID of the node that accessed the file (shown as accessor), and the timestamp.

By adding the IPFS hash address to the fabric network, we have limited access to files added to BlockIPFS ensuring that only members of the fabric network can access the hash address that can be used to obtain the file on IPFS network, assuming all member nodes are responsible. This also narrows down the number of possible users that can access a file and reduces possible culprits in case of unauthorized access. Furthermore, in IPFS, user access activities are not logged so users can freely access any file on the network without any repercussion. In this experiment, user file access is logged in an immutable secure-by-design ledger which becomes a deterrent for unauthorized access.


```
"File1":{"accessor":"QmeBPUP8LCBukcJXBoGKjvZX2pkDTL4g62jZW5uek1512a",
"filehash":"QmUzCbRpN9fCkMMG3qRopCemVEAVNSmEyo2i39QJCEfEgb",
"owner":"Earth",
"time":"2019-02-13 15:37:11"},

"File2":{"accessor":"QmQJrfkhFomXHP7YxxLX5BeeKbfSyptMSeZdWQ6xDpRaoZ",
"filehash":"QmRyCymXw8LKY3oAkMPJfmuzqCyz2LSThUKFAR1pUFvQxU",
"owner":"Earth","time":"2019-02-17 13:00:01"},

"File3":{"accessor":"QmfNakL2fmYxDS9VZtuAhxwt9TnJYf9PYnHszGSqWnjXGY",
"hash":"QmQ2pqNJtSN4e4HbKR7h5At17GYzXqPGAgFCuzaTrFhEv2",
"owner":"Earth",
"time":"2019-02-17 13:08:19"},

"File4":{"accessor":"QmafRVVvjzEbtmDo6ArREXVDWXtDiAetjMTTKiKzSa8qAa",
"filehash":"QmTJxeb9o8pNUn3fa8JvACVf3yx8WVSPHjh4iabXtbhBH2",
"owner":"Earth",
"time":"2019-02-17 13:09:41"},

"File5":{"accessor":"QmQJrfkhFomXHP7YxxLX5BeeKbfSyptMSeZdWQ6xDpRaoZ",
"filehash":"QmR6qUj7cmNBBNttHCgsLumbw2P58DUYRgWbgd6sCiRbtR",
"owner":"Earth",
"time":"2019-02-17 13:12:55"}
```


Fig. 7. On-chain metadata for files be accessed via BlockIPFS.

C. *Scalability*

All distributed file systems use the similar technologies as IPFS and provide some metadata that can be used to trace files added to the network. Most distributed file systems return a hash address of files. The returned hash may become part of the metadata that serves as input to our solution, providing an extra functionality to enhance access and provide traceability capabilities. For instance, Hadoop Distributed File System (HDFS) is an integral part of the Hadoop ecosystem and serves as the primary data storage technology in Hadoop systems. HDFS uses a NameNode and DataNode architecture to implement a distributed file system that provides high-performance access to data across highly scalable Hadoop clusters [14]. Our solution can be applied to HDFS to enhance access to files and protect authorship and check authenticity of files by collecting extra metadata from HDFS and storing those metadata on the blockchain.

We also were interested to know how our proposed solution performed as the number of nodes on the BlockIPFS increased. To this end, we performed a particular analysis to measure the upload, read, and download transaction times of each of the files we added to the network with the number of nodes ranging from 3, 9, 18, and 27 respectively. The results are shown in Figures 8, 9, and 10. The files were uploaded using the Ubuntu machine running BlockIPFS and we used the Windows 10 and Arch Linux machines to access them. At each level we increased the number of nodes on the network from 3, 9, 18, and finally 27. We noticed that as the number of nodes increased there is a slight change in the transaction time. We conclude that this extra overhead does not cause a significant slow-down in the overall performance of the system as the number of nodes increases.

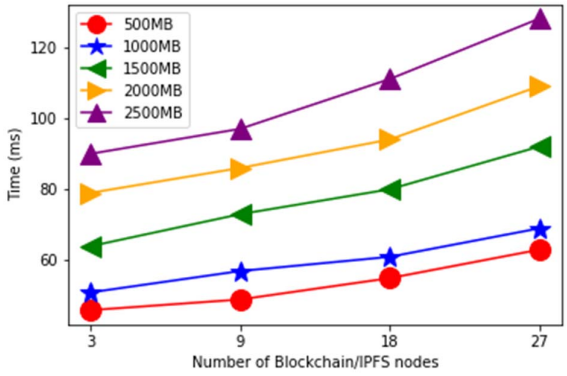

Fig. 8. Elapsed time of file upload transactions on BlockIPFS.

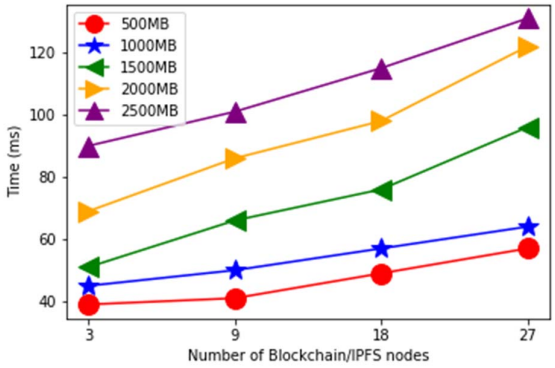

Fig. 9. Elapsed time of file read transactions on BlockIPFS.

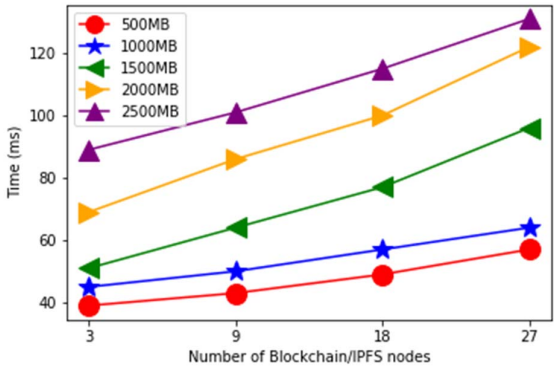

Fig. 10. Elapsed time of file download transactions on BlockIPFS.

V. Discussion

A. *Implications on data forensics*

Our goal in this work is to provide means to trace and audit access of files added to IPFS. Although it may prove close to impossible to prevent users from sharing content offline, BlockIPFS makes efforts to deter unauthorized access or prove whether a user has accessed some file on IPFS that he/she is not supposed to. This information could be very useful in tracing back access to a file either in legal or organizational environments.

IPFS generates a hash address of any uploaded file which in turn becomes part of the metadata that is stored on the blockchain. Metadata in our approach include the file's hash, owner's name, date and time of the upload were successfully added to the blockchain. Access activity metadata such as who accessed the file, file name, what kind of access, and timestamp are also recorded and added to the ledger. Whenever needed, a user would be able to query the blockchain to retrieve these metadata providing a record of all activities related to the file.

Users on the blockchain can retrieve the hash of a specific file and use it to access content on IPFS. However, anyone not a user on the blockchain who has access to the hash can still access content on the IPFS network but each of performed operations will be logged into the blockchain giving the author the capability to audit and trace access activities on the network and take any relevant actions.

B. *Limitations*

Although the proposed solution provides several new functionalities to a distributed file system such as IPFS, we have also noted some limitations to the solution as implemented. First, the limited similarity checking. Most distributed file systems are built for ease of access and storage rather than authorship protection. This inherent characteristic of distributed file systems makes it difficult for us to perform similarity checking to identify possible plagiarism. Second, the proposed solution does not completely limit access to hash addresses. The current implementation of the proposed solution is based on the assumption that users in the private network will not share hashes of files uploaded to the network with others. There is currently no functionality that prevents a legitimate user from sharing hashes with others outside the network, for instance through email or social media. Or preventing an illegitimate user from accessing a file. Third, our proposed solution adds new functionalities and technologies that consumes extra processing power and time. Our goal has been to minimize these limitations and to provide a seamless experience for the user in upcoming works.

VI. Conclusion and Future Work

In this paper, we explored the potential of integrating blockchain with a distributed file system (i.e. IPFS), in order to create a better system that provides both the capabilities of security and traceability of a blockchain technology [23], and efficiency of distributed files systems. Although distributed file systems are efficient and fast at sharing files, they are not very effective when it comes to security and traceability. Blockchain technologies on the other hand provide traceability and integrity. BlockIPFS combines these capabilities of both technologies to provide traceability, integrity, and to protect authorship of files uploaded to distributed file system networks.

Currently, BlockIPFS pushes only metadata of files added to the distributed file system. For future work, BlockIPFS can be expended to track all upload, download, and access activities in the distributed file system. Additionally, a major problem that IPFS needs to resolve is how to handle updates to content on its

network. To provide a way to maintain access even when a file's hash address changes, the community has most recently introduced the concept of IPNS. IPNS name maps objects to the public ID of each node in the network. It is associated with a record containing information about the hash it links to that is signed by the corresponding private key [15]. This feature provides a means for content to be linked to a name that is stable so that even when content hashes change, they can be access using the stable IPNS. Thus, a future goal of this research is to provide capabilities that tracks and logs all user activities performed via IPNS. This will enable users to manage access control for different copies of files they have added to IPFS.

REFERENCES

[1] J. Benet, "IPFS-Content Addressed, Versioned, P2P File System", arXiv:1407.3561v1, 2014.

[2] J. Yli-Huumo, D. Ko, S. Choi, S. Park, K. Smolander, "Where Is Current Research on Blockchain Technology?—A Systematic Review", PLoS ONE 11(10): e0163477, 2016.

[3] Y. Chen, H. Li, K. Li and J. Zhang, "An improved P2P file system scheme based on IPFS and Blockchain", *IEEE International Conference on Big Data (Big Data)*, Boston, MA, pp. 2652-2657, 2017.

[4] Protocol Labs, "Filecoin: A Decentralized Storage Network", 2017.

[5] Hyperledger Architecture Working Group, "Hyperledger Architecture, Volume 1". Available at: https://www.hyperledger.org/wp-content /uploads/2017/08/Hyperledger_Arch_WG_Paper_1_Consensus.pdf.

[6] Hyperledger Performance and Scale Working Group, "Hyperledger Blockchain Performance Metrics". Available at: https://www.hyperledger.org/wp-content/uploads/2018/10/HL_ Whitepaper _Metrics_PDF_V1.01.pdf.

[7] V. Buterin, "A Next Generation Smart Contract & Decentralized Application Platform". Available at: http://blockchainlab.com/pdf/ Ethereum_white_paper-a_next_generation_smart_contract_and_ decentralized_application_platform-vitalik-buterin.pdf

[8] T. Sato and Y. Himura, "Smart-Contract Based System Operations for Permissioned Blockchain", *9th IFIP International Conference on New Technologies, Mobility and Security (NTMS)*, Paris, pp. 1-6, 2018.

[9] M. Valenta, P. Sandner, "Comparison of Ethereum, Hyperledger Fabric and Corda", Frankfurt School Blockchain Center, 2017.

[10] 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)", *IEEE 36th Symposium on Reliable Distributed Systems (SRDS)*, Hong Kong, pp. 253-255, 2017.

[11] C. Cachin, "Architecture of the Hyperledger Blockchain Fabric", IBM Research - Zurich , 2016.

[12] F. Benhamouda, S. Halevi and T. Halevi, "Supporting Private Data on Hyperledger Fabric with Secure Multiparty Computation", *IEEE International Conference on Cloud Engineering (IC2E)*, Orlando, FL, pp. 357-363, 2018.

[13] .L. Koonce, "The Wild, Distributed World: Get Ready for Radical Infrastructure Changes, from Blockchains to the Interplanetary File System to the Internet of Things", *Intellectual Property & Technology Law Journal* , vol. 28, pp. 1-6, 2016.

[14] M. Rouse, "What is Hadoop Distributed File System (HDFS)", Available at: https://searchdatamanagement.techtarget.com/definition/Hadoop-Distributed-File-System-HDFS, 2019.

[15] IPFS Project, "IPNS", Available at: https://docs.ipfs.io/guides/concepts/ipns/

[16] Hyperledger Fabric, Available at: https://hyperledger-fabric.readthedocs.io/en/ release-1.4/write_first_app.html

[17] Golang, Available at: https://golang.org/dl/

[18] Nodejs, Available at: https://nodejs.org/en/

[19] Docker, Available at: https://www.docker.com/get-started

[20] JS-IPFS, Available at: https://docs.ipfs.io/introduction/install/

[21] Hyperledger Fabric, Available at: https://hyperledger-fabric.readthedocs.io/en/release-1.4/whatsnew.html

[22] R. M. Parizi, Amritraj, and A. Dehghantanha, "Smart Contract Programming Languages on Blockchains: An Empirical Evaluation of Usability and Security," *1st International Conference on Blockchain (ICBC'18)* , Seattle, USA, LNCS, Springer, pp. 75-91, 2018.

[23] P. J. Taylor, T. Dargahi, A. Dehghantanha, R. M. Parizi , K. K. R. Choo, "A Systematic Literature Review of Blockchain Cyber Security", *Digital Communications and Networks*, https://doi.org/10.1016/j.dcan.2019. 01.005, Elsevier, 2019.