Cloud Data Provenance using IPFS and Blockchain Technology

Syed Saud Hasan
Indian Institute of Information Technology Guwahati
Guwahati, Assam
saudh58@gmail.com

Nazatul Haque Sultan
Indian Institute of Information Technology Guwahati
Guwahati, Assam
nazatul@iiitg.ac.in

Ferdous Ahmed Barbhuiya
Indian Institute of Information Technology Guwahati
Guwahati, Assam
ferdous@iiitg.ac.in

ABSTRACT

Cloud is widely used for data storage. A user who has uploaded his/her private or commercial data to the cloud is always keen to know whether the data that he/she has stored is secure or not. Access logs of stored data can be used to trace the integrity of the data. Access logs are also known as provenance data. Provenance data contains private information of users. As provenance data can be used to check the integrity so, it becomes very important to securely store the provenance data. The stored provenance data should be immutable and also unreachable to adversaries, as it contains private information of users. Cloud users will be assured of their stored data, and Cloud Storage Providers can use this implementation to improve their brand value and performance. This paper aims at providing an efficient way to store provenance data securely using Blockchain technology and InterPlanetary File System (IPFS) so that it is out of reach of adversaries. This paper also proposed a framework through which a user can verify the integrity of its own data. This model is implemented, tested and analyzed using IPFS which is a decentralized storage mechanism backed by blockchain to store cloud provenance data and uses publicly available Tierion api to store the hash value of the provenance entries.

CCS CONCEPTS

• **Security and privacy → Security services**; • **Security services →** *Authorization*; Auditing;

KEYWORDS

Data Provenance; Cloud Storage; Blockchain; IPFS

ACM Reference Format:
Syed Saud Hasan, Nazatul Haque Sultan, and Ferdous Ahmed Barbhuiya. 2019. Cloud Data Provenance using IPFS and Blockchain Technology. In *The Seventh International Workshop on Security in Cloud Computing(SCC '19), July 8, 2019, Auckland, New Zealand.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3327962.3331457

1 INTRODUCTION

Normally, the provenance data are stored in a centralized database, under the surveillance of a trusted entity which is generally the Cloud Storage Provider (CSSP) itself [8]. However, it is not wise to fully trust the CSSP, as it may tamper the data for its own benefits. For example, CSSP can alter the stored provenance data to hide any unauthorized access, as it has full control on that database. This problem can be solved by storing the provenance data in a distributed manner, which is a challenging task.

Blockchain can be a suitable decentralized technology to address the above-mentioned issue [10]. Blockchain technology which is a distributed and decentralized network where nodes in the blockchain network can transfer values among themselves without using any third party. Blockchain technology has broadly four principles- *auditability, efficiency, immutability* and *transparency* [13]. Every node in the Blockchain network takes part in the consensus mechanism to add a block to the Blockchain network. Each block contains a set of the transaction which has been witnessed and verified by network nodes. Using Blockchain technology each and every action can be added to the public Blockchain ledger. As data stored in Blockchain is immutable, the trust between users and CSSP can be enhanced if the provenance data are kept in the Blockchain instead of the centralized database of the CSSP. However, the storage of lot's of data on Blockchain is very costly due to it's distributed nature (this is broadly explained in Section 3.1). This problem can be catered using a decentralized file storage system, like IPFS [4].

In this paper, we propose a Blockchain and IPFS based architecture to provide assured data provenance mechanism for cloud storage application, while enhancing privacy and availability at the same time. The main contribution of this paper is as follows:

(1) Provenance database is distributed among all the users and CSSP using IPFS. Data stored in IPFS is immutable hence can't be altered by the adversary.
(2) As the provenance data are stored with each user in the Blockchain network, there is no single point of failure, unlike the centralized CSSP. This also provides high availability.
(3) The proposed model is implemented, tested and analyzed, and the implementation results demonstrate practical usability of the proposed model in terms of computation and storage overhead.

To the best of our knowledge, this is the first work that addresses the problem of cloud provenance data storage and auditing in a fully decentralized manner.

Rest of the paper is organized as follows. In Section 2, a brief overview of the existing works is presented. Section 3 presents some preliminaries which will be used throughout this paper. The proposed scheme is presented in Section 4. Implementation results and analysis of the proposed scheme are presented in Section 5 and finally, Section 6 concludes this paper.

ACM ISBN 978-1-4503-6788-2/19/07...$15.00
https://doi.org/10.1145/3327962.3331457

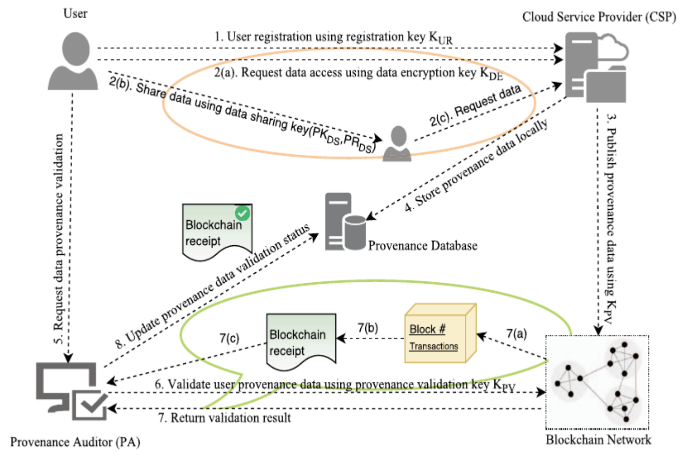

User
Cloud Service Provider (CSP)
1. User registration using registration key K_{UR}
2(a). Request data access using data encryption key K_{DE}
2(b). Share data using data sharing key(PK_{DS},PR_{DS})
2(c). Request data
3. Publish provenance data using K_{PV}
4. Store provenance data locally
Provenance Database
Blockchain receipt
5. Request data provenance validation
8. Update provenance data validation status
7(c)
Blockchain receipt
7(b)
Block # Transactions
7(a)
6. Validate user provenance data using provenance validation key K_{PV}
7. Return validation result
Provenance Auditor (PA)
Blockchain Network

Figure 1: *ProvChain* System Interaction [8]

2 RELATED WORKS

Data provenance is among one of the oldest ways, how people used to represent the access log of any object. The advent of cloud computing allows users to keep their personal data away from the physical control of the user. It was difficult for cloud users to rely on the CSSP. This issue of reliability leads many scholars to think of using data provenance mechanism on cloud data, to improve the reliability. This section deals with a few of the existing cloud data provenance approaches in detail.

Data provenance describes how a particular piece of data has been produced. Provenance data is the access log which is generated, and it is vital for post-incident investigation. To improve the availability of the files, CSSP generally keeps the copy of the data at multiple locations. Due to several reasons like machine failure and machine upgrade, cloud data can be shifted to some other machine either within the same data center or among remotely situated data centers. This transition of stored data which is generally hidden from cloud user. It can create a problem/difficulty in data provenance as collecting provenance data and finding errors in such a system is difficult. One way to resolve this problem to collect logs from all the machines wherever data moves. However, different machines running in such cases can be of different architecture and might be running different software. Hence, collecting provenance data is challenging. So, the provenance data must include each and every action done either on the data or machine associated with that data.

J Freire *et al.* [7] proposed a technique which can automatically collect and maintain provenance or history of an item. In 2013, S. sultana *et al.* proposed *A file provenance system* [12] which is a scalable file system and is capable for capturing provenance record at various degree of granularity and transforming them into secure information. The scheme also directs the resulting provenance data to various persistence storage system. *S2logger* [11] is a data-centric logging technology which can trace data activities like (file creation, edit, duplication, transfer, deletion, etc). The data events are captured at the file and block level.

Provenance data may contain personal information of users, hence it must be stored in a very secure manner. Several attempts have been made to propose an architecture for the secure provenance storage system. In 2012, M. Asghar *et al.* proposed a scheme for securing data provenance in the cloud while offering encrypted search [3]. In the proposed scheme, neither an adversary nor the cloud storage provider can learn about the data provenance or query. Liang *et al.* proposed a blockchain based cloud data provenance architecture *ProvChain* [8]. *ProvChain* provides a way to collect and verify cloud provenance data. The architecture (please refer Figure 1) of the *ProvChain* is broadly divided into three phases, namely *Provenance data collection, Provenance, data storage and Provenance data validation* which are briefly described as follows.

- **Provenance data collection**: This phase of *ProvChain* collects data from the hypervisor using data loggers. Data collection takes place from the hypervisor level of cloud servers so, it becomes difficult for an adversary to harm the provenance data at the collection level.
- **Provenance data storage**: Once the data is collected at the OS level, provenance data gets stored in a centralized database, and its corresponding hash is stored in the Blockchain. The Centralized database is completely under the control of the CSSP. This could be a threat to the cloud users data as CSSP can itself be malicious.
- **Provenance data validation**: Once the cloud user has opted for the cloud provenance mechanism it can request for its provenance data validation at any point of time. Provenance auditor in the network will collect data from the Blockchain and centralized database and will cross check if any of the provenance entry has tampered.

The main drawback of the *ProvChain* is that it stores the cloud provenance data into the centralized database of the CSSP. This enables CSSP to misuse and manipulate users provenance data.

3 PRELIMINARIES

Few concepts like Blockchain Technology [10], secure Hash function (e.g., SHA-256), InterPlanetary File System (IPFS) [4] would be required to understand the model perfectly. A brief overview of these technologies is given in the following subsections.

3.1 Blockchain Technology

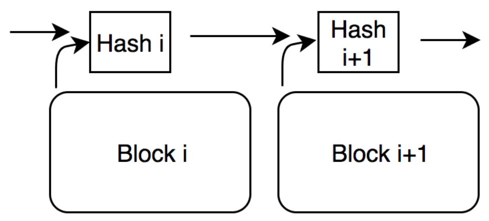


Figure 2: Blockchain architecture

Blockchain is a distributed decentralized digital ledger which is used to transfer values among multiple nodes connected via peer-to-peer network. Transactions stored in Blockchain ledger are immutable and are logged with a time-stamp. Trust among the nodes comes as the transactions are secured by cryptography. Compromising a Blockchain based system is very difficult as the adversaries have to employ more computational power than the distributed Blockchain system has. Huge Computational power is required to solve large and complex mathematical problems to break the cryptographic SHA-256 hash, which is not practically feasible with the current technology.

Public Blockchain possesses four basic properties, namely auditable, efficient, immutable and transparent.

- **Auditable** - Public Blockchain is open ledger, and anyone can check and verify the transactions at anytime.
- **Efficient** - Although it requires lots of computational power to add a block to Blockhain but, once it is added it is very easy to cross check the transactions.
- **Immutable** - Transactions get added to Blockchain after a consensus algorithm. Different Blockchains uses different consensus algorithm. These consensus algorithms work in order to make sure that all nodes in the network have the same copy of the ledger.
- **Transparent** - A public Blockchain is transparent in nature any participant can come and see which transactions are accepted by the network and which are not accepted.

Blockchain technology provides trust among two transacted parties, which may be unknown to each other without using the third party. It works on consensus mechanism. Blockchain can be of following types.

- **Public Blockchain** - Anyone is allowed to read, write the ledger and can take part in consensus. Bitcoin is backed by public Blockchain where everyone in the network has equal authority.
- **Permissioned Blockchain** - It allows only a few of the nodes to take part in the consensus. Consortium Blockchain is the example of permissioned Blockchain, where only group of companies can be those permissioned nodes and rest of their customers will be just there.
- **Private Blockchain** - It permits only one entity to control the network.

Different Blockchain have different consensus algorithm like Proof of Work, Proof of Stake, Proof of Authority, etc.

- **Proof of Work (PoW)** - Involves solving a computationally challenging puzzle in order to create a new block in Blockchain. The proof is hard to generate but very easy to verify. This algorithm requires huge computation hence it is energy intensive. This consensus algorithm is used in Bitcoin [10].
- **Proof of Stake (PoS)** - This is a lottery based consensus algorithm. Nodes are selected randomly to validate a particular transaction. Random selection depends upon the share that the particular node has. This is computationally less exhaustive. This consensus algorithm is used in Ethereum [5].
- **Proof of Authority (PoA)** - This algorithm is used for permissioned Blockhain [6]. This uses a permissioned authority which is allowed to create new blocks. This algorithm is again based upon voting mechanism. Ledgers using PoA algorithm requires sign-off by the majority of authorities for the block to be generated.

For a transaction to get complete, it has to go with certain steps. The sender of the transaction would generate a transaction by deciding the receiver's address, the value that it wants to send, and would sign the transaction by its own keys. The node using which the sender has generated the transaction would cross-check whether the transaction has been signed by the sender. Once the signature of the transaction is verified at local level, node will broadcast it to the network where miner nodes would select them. Based upon the consensus algorithm, they will add the transaction to the block and will broadcast the newly formed block to the network where rest of the nodes will add the block to their own Blockchain to get synced with the network. Figure 2 shows how blocks are attached in the Blockchain in a chronological order. These blocks contain several transactions and are attached using cryptography.

Blockhain network is a distributed network of nodes, hence every node has the same ledger information. This becomes a problem when a huge amount of data is added to the Blockhain network. This problem can be solved by using IPFS. The IPFS stores the data effectively in the distributed manner. This technology is broadly explained in the next subsection.

3.2 Cryptographic Assumptions

The secure hash algorithm is a family of cryptographic hash functions. Given a unique value as an input SHA-256 produces almost a unique 256-bit signature. 256-bit key makes it good partner function for Advanced Encryption Standard (AES). SHA-256 is a one-way cryptographic function, i.e., it is very easy to get a hash value of a string but it is impossible to get back the string from the hash value. This is cited as impossible as reversing back would require a huge amount of mathematical computational, which is not possible with the current technology.

3.3 Inter-Planetary file system (IPFS)

InterPlanetary File System is a Blockchain based distributed file storage system. It's a peer-to-peer file storage system that seeks to connect all computing devices with the same system of files. IPFS is a content addressable file storage system, i.e., the location of the content depends upon the content itself. This is decentralized file storage system working on the principle of different distributed web technologies like Git [1], SFS [9], etc.

In this peer-to-peer file system, nodes have permission to store and retrieve data to and from the network. As the data stored in IPFS is content addressable hence, once the user stores some data on the IPFS network, network replies with a unique hash pointing to the stored data. The returned hashed value can be further used to retrieve same data from the IPFS network. IPFS is also known as permanent web, as data is not stored on any of the central servers but is distributed across all the participants. This distribution is in such a way that a person can't exactly locate the address of a particular file, i.e., on which particular node a file is stored.

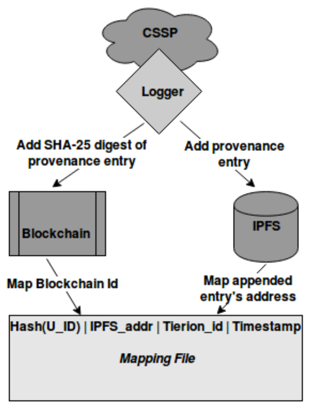

CSSP
Logger
Add SHA-25 digest of provenance entry
Add provenance entry
Blockchain
IPFS
Map Blockchain Id
Map appended entry's address
Hash(U_ID) | IPFS_addr | Tierion_id | Timestamp
Mapping File

Figure 3: Data collection and storage

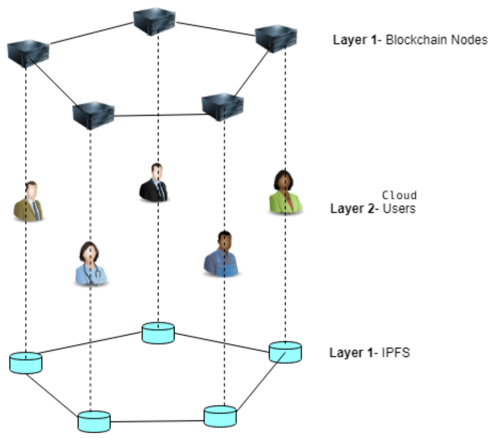


Figure 4: Layered view of the proposed model

IPFS is presently available in public version, i.e., anyone from anywhere having an Internet connection can connect to IPFS network. However, IPFS people are working on a private version of IPFS where only participants with specific permission can join the network.

4 PROPOSED SCHEME

Provenance data corresponding to each and every action performed on the stored data is stored in IPFS based distributed file and a hash value corresponding to the provenance data is stored in Blockchain as a transaction. A list of these transactions has formed a block. If someone wants to modify the provenance data, she/he has to employ lots of computing power. Later using the data stored in IPFS and hashes in Blockchain Provenance Auditor can verify the integrity of the stored data.

The proposed model is a Blockchain and IPFS based fully decentralized architecture for cloud data provenance mechanism. It can be thought of as a layered architecture which is shown in Figure 4, where the middle layer consists of all the actors such as cloud users and CSSP itself. Corresponding to each of the actors in the top layer there will be a Blockchain node and an IPFS node in the bottom layer.

The proposed model can logically be divided into 3 parts- 1). **Provenance data collection**: as shown in Figure 3, loggers act at the hypervisor level of the cloud servers and collect cloud provenance data. 2). **Provenance data storage**: again as shown in Figure 3, IPFS is used to store the cloud provenance data and Blockchain is used to store the hash value of cloud provenance data. 3). **Provenance data validation**: here, the user can ask the provenance manager for provenance data validation. Provenance manager will collect provenance data from IPFS and it's corresponding hash from Blockchain, and it will run the verification algorithm to validate the provenance data.

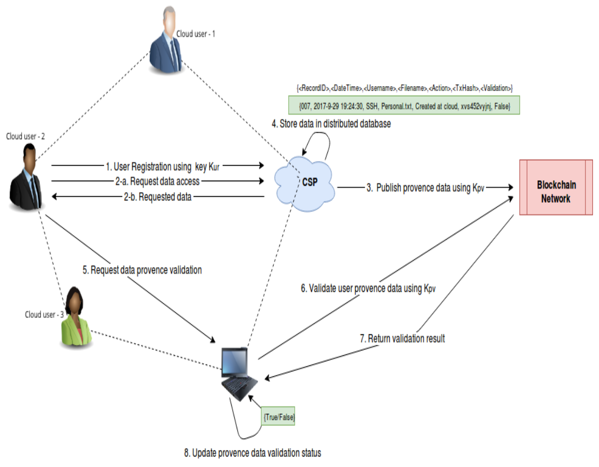

Cloud user - 1
{<RecordID>,<DateTime>,<Username>,<Filename>,<Action>,<TxHash>,<Validation>}
{007, 2017-9-29 19:24:30, SSH, Personal.txt, Created at cloud, xvs452vyjnj, False}
4. Store data in distributed database
Cloud user - 2
1. User Registration using key Kur
2-a. Request data access
2-b. Requested data
CSP
3. Publish provence data using Kpv
Blockchain Network
5. Request data provence validation
6. Validate user provence data using Kpv
Cloud user - 3
7. Return validation result
{True/False}
8. Update provence data validation status

Figure 5: System Interaction

4.1 Benefits

Using decentralized techniques like Blockchain [Section 3.1] technology and IPFS [Section 3.3] for proving the integrity of stored files on CSSP provides several benefits-

- Provenance database is distributed among all the users and CSSP, hence it brings even more trust among the cloud users.
- There is no single point of failure.
- Digest values stored in Blockchain, corresponding to provenance data stored in IPFS is completely immutable, due to one of the property of Blockchain.
- It is not possible for any node to find the exact storage location of a particular provenance entry stored in IPFS, as data stored in IPFS is anonymized [refer Section 3.3].

4.2 System Model

Figure 5 depicts planer view of the proposed system. It constitutes of few major entities like Cloud User, Cloud Storage Service Provider (CSSP), IPFS, Blockchain Network and Provenance Auditor whose role are explained briefly as follows:

Algorithm 1 Data Storage

```
//Adding provenance entry to IPFS and collecting it's hash
IPFS_hash = echo <provenance_entry> | ipfs add

// Add SHA256 value of provenance entry into Blockchain
Blockchain_id = add_Hash(sha256(provenance_entry))

// Update mapping.txt file
update_mappingFile(sha256(UserID), Blockchain_id, IPFS_hash, timestamp){
        append(sha256(UserID, Blockchain_id, IPFS_hash, timestamp) -> mapping.txt file
    }
```

(a) Data Storage Algorithm

Algorithm 2 Provenance algorithm

```
//User can request for provenance data validation by sending it's mapping file
request_validation(Report)

//Provenance auditor will validate
bool validate_report(Report){
    For all entry R in Report{
    IF{ Valid(Blockchain_id) == true}{
                X = get_hash_from_Blockchain(Blockchain_id)
                Y = get_value_from_IPFS(IPFS_hash)
                            if(X != SHA256(Y))
                                    return False; // Data is tampered
                            } //if
                    } //for all
        return True; // All good
    }
```

(b) Data Provenance Algorithm

Figure 6: Algorithms for Data Storage and Provenance

- **Cloud User.** As shown in Figure 5 all those actors which use the cloud storage service are cloud users including the owners (who own data). To opt for the cloud storage service, cloud users have to first complete *user registration* with CSSP. A user can modify data and permissions associated with the data stored. Once the file is uploaded cloud user can opt for the data provenance service.
- **Cloud Service Provider (CSP).** CSP or CSSP (Cloud Storage Service Provider) provides storage as a service to the cloud users. It is also responsible for user registration. Only CSSP knows the real identity of cloud users. Provenance manager within CSSP is responsible for storing provenance data in the stored database and uploading the hash value of the provenance data onto the Blockchain network.
- **IPFS.** This database is distributed among CSSP, Provenance Auditor and all cloud users. Upon detection of any of the provenance data provenance entry, CSSP appends the newly received provenance data onto IPFS-Distributed database. Only CSSP and Provenance Aanager will have the write permission onto this database and all the cloud users can only read and check the activity of CSSP and Provenance Auditor. Once the Provenance Auditor has validated some data, it will update the status corresponding to the entry in the distributed data.
- **Blockchain Network.** It stores the Hashed value of provenance data. Blockchain is distributed among all cloud users, CSSP, and Provenance Auditor. Only CSSP can write to this database and all other actors will be just able to read the Blockchain data.
- **Provenance Auditor.** Upon getting a request for data provenance validation, Provenance Auditor will collect provenance data from Blockchain and IPFS to validate the data provenance. Provenance Auditor can validate but can't relate the provenance data to the real identity of users.

4.3 Algorithmic Explanation

Once the provenance data is collected, it is stored in IPFS and Blockchain network. Also, the mapping file will be updated accordingly. The algorithms are as follows:

- **Data Collection**: Once an action (write, update, etc.) is initiated on files by a user, the generated provenance data are collected in the loggers. This collected provenance data will be further added to IPFS and its hashed value will be added to the Blockchain. This is shown in Algorithm 6a.
- **Provenance Data Storage**: Once the provenance data is collected, calculate SHA-256 value of the collected provenance entry. Insert the provenance entry into IPFS and insert the SHA-256 value into Blockchain as explained in Algorithm 6a. IPFS network will reply with IPFS_id and Blockchain network will reply with Blockchain_id. Store the mapping of IPFS_id and Blockchain_id along with the time-stamp in the mapping file which will be stored at the cloud user end.
- **Cloud Data Provenance**- Once the user has demanded the provenance data validation, Provenance Auditor will collect the provenance data associated with that particular user from the mapping file. In algorithm 6b, provenance data from IPFS and SHA-256 value from the Blockchain are fetched and are matched for each row. If the values corresponding to all the row are matched then provenance auditor returns True (i.e, the data is not tempered), otherwise returns False (i.e., the data is tempered).

4.4 Architectural Flow

Figure 7 explains the process flow. A detailed explanation is given as follows.

- **Register**- Cloud user starts the process by registering with the CSSP. CSSP will ask all the required information from the cloud user to register with its service.
- **Get Keys**- In response to the registration, cloud user will get a pair of key, which will be used to encrypt and decrypt data while storage and retrieval.
- **Upload Data**- Cloud user will upload its file to the cloud.

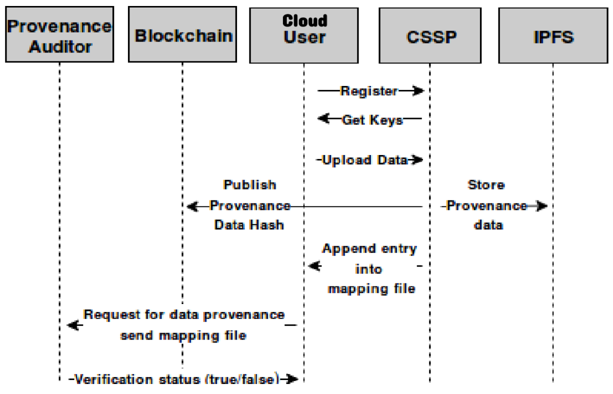

Provenance Auditor
Blockchain
Cloud User
CSSP
IPFS
Register
Get Keys
Upload Data
Publish Provenance Data Hash
Store Provenance data
Append entry into mapping file
Request for data provenance send mapping file
Verification status (true/false)

Figure 7: Architecture flow.

- **Store Provenance Data-** Upon any of the users (cloud users or adversaries) attempt to access the stored file, CSSP will generate cloud provenance data which will be stored in IPFS and Blockchain networks.
  - **Publish Provenance Data-** SHA-256 value of the provenance data is calculated and is stored in the Blockchain. In response to this storage, Blockchain will respond with BlockchainId.
  - **Store Provenance Data in IPFS-** The collected provenance data will be stored as such in IPFS. IPFS network will respond with a *IPFS_hash*, which would be used to retrieve data in future.
- **Append Entry to Mapping File-** Append an entry containing mapping of received BlockchainID and IPFS_hash into mapping file.
- **Request for data provenance-** User can request for validity of it's stored data from the provenance auditor. Along with this request user will send it's mapping file also.
- **Verification Status-** By accessing the mapping file, provenance auditor will request data from IPFS and Blockchain and will cross-check the data stored in the IPFS, and will reply with the validity of the data to the corresponding user.

4.5 Improvement

The model proposed in section 4.3 works fine with the proposed algorithm. The result and analysis corresponding to the proposed model are shown in graph 9b. Graph 9b clearly shows a linear variation of number of entries with file size. Although a linear increase in the storage database is not a very big issue, this can also be minimized.

Once the user has opted for cloud data provenance, up to a certain number of entries, those many entries from the file *mapping.txt* can be removed. So, if a user is usually opting for the provenance service in a certain interval of time, then the resultant *mapping.txt* for that user will be of almost a constant size [Figure 10].

Technologies Used	Version
Node.js	6.12.0
IPFS	0.4.10
Tierion Chainpoint	3.0

Table 1: Technologies Used

5 IMPLEMENTATION RESULTS AND ANALYSIS

This section first presents the implementation details of the proposed scheme and its results, and then presents its analysis.

5.1 Implementation Results

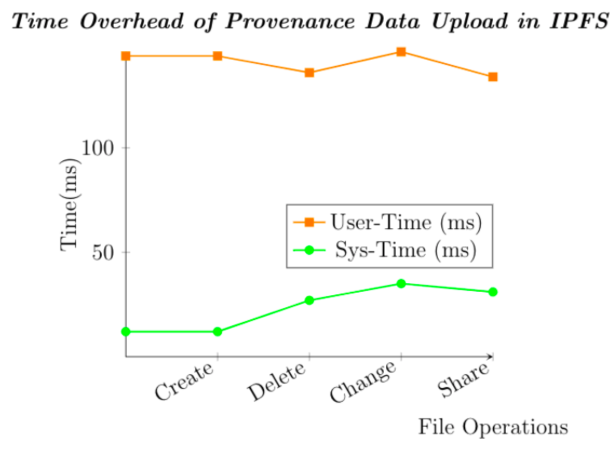
Implementation is broadly divided into 3 modules- **Provenance data collection**, **Provenance data storage**, **Provenance data validation**. All of these modules work independently. Implementation has been done in Node.js. Publicly available Tierion blockchain api [2] is used to store hash value and IPFS is used as a data storage layer. Details about used technology are mentioned in Table 1.

Each implemented module works independently. Following is the detailed explanation how each of the modules is implemented and how they are connected.

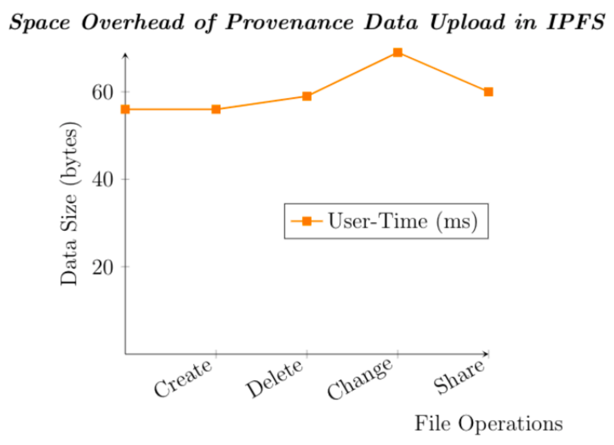
- **Provenance data collection-** This module is responsible for the collection of provenance data. Provenance data was collected from the OS level using S2-Logger. Several details regarding the file access are collected and the username of the owner is hashed to provide the anonymity to the system. Figure 11 is the example of how provenance entry looks like.
- **Provenance data storage-** This module stores the provenance data and it's corresponding hash into the Blockchain.
  - **Storing into IPFS-** Collected provenance entry is directly stored in IPFS, and in reply to that, node receives the IPFS_hash. This IPFS_hash can be used in future to extract this data from IPFS network.
  - **Storing into Blockchain-** The SHA-256 value of the provenance entry is stored into Tierion Blockchain. Upon successful anchoring of data, Tierion Blockchain will respond with Tierion_id. This Tierion_id will be used for the proof of existence of the data from the Blockchain.

  Moreover, IPFS_hash received after storing provenance data into IPFS, Tierion_id received after storing the SHA-256 value of provenance data into Blockchain, timestamp from the machine's internal clock and hash value of the current user_id will be stored in a mapping file as shown in Figure 3.
- **Provenance data validation-** This module is required when the user of cloud requests the verification of it's stored provenance data and it's corresponding hash into IPFS and Blockchain. This module returns true if hash value of the provenance data stored in IPFS matches with the hash value stored in Blockchain. Else it returns false.

We have executed the program and obtained the results on a machine with configurations mentioned in Table 2. Figure 8 represent the overhead caused by the system due to storing data in the IPFS. Figure 8a represents the extra computation caused due to storing

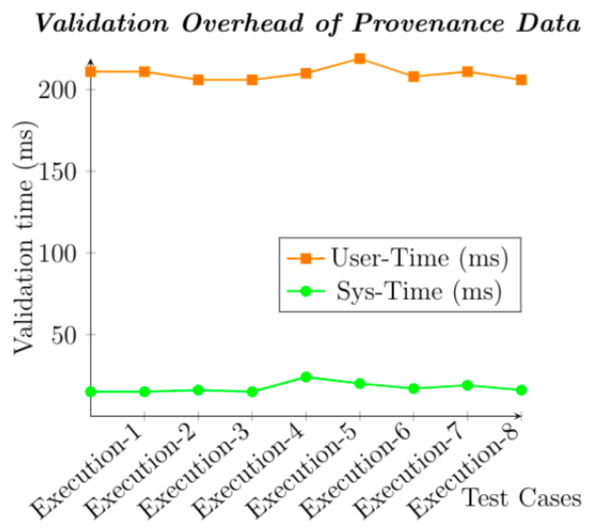


(a) Time overhead for provenance data upload in IPFS

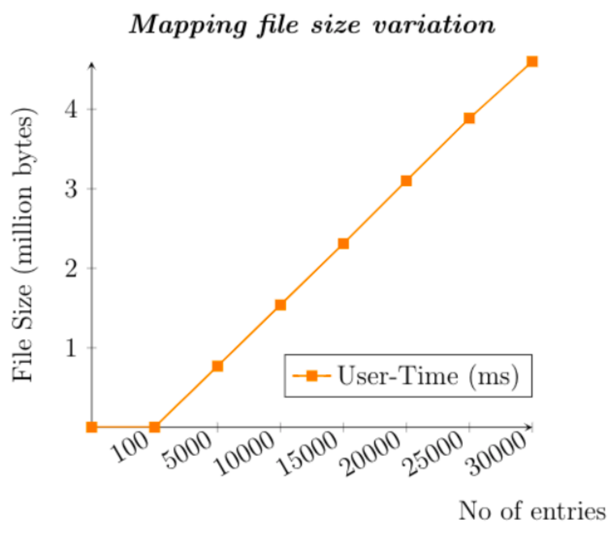


(b) Space overhead for provenance data upload in IPFS

Figure 8: Redundancy due to storing data into IPFS and Blockchain.



(a) Delay in Validation



(b) Mapping file size variation

Figure 9: Validation time & mapping file size.

Resource Name	Configuration/details
Machine Model	Lenovo G405
Operating System	Ubuntu 14.04 LTS
CPU	AMD A4
RAM	4 GB
CPU clock speed	1.5 GHz

Table 2: Resource Used

provenance data and Figure 8b shows the amount of data stored in different file operations. Figure 9 corresponds to the validation overhead and storage overhead. Figure 9a represents the time taken by CPU for validating a record, and Figure 9b represents the mapping file size with different number of entries in the file. Figure 10 shows the result of the improvement that has been done in Section 4.5. Results in Figure 10 is clearly an improvement over the result in Figure 9b, as Figure 9b is linear whereas Figure 10 is constant in nature. This graph is calculated assuming that user will request for the provenance service once number of entries in *mapping.txt* reaches 10,000. Assuming above condition to be true, then average file size of *mapping.txt* will always be around 1,540,000 bytes, which is a constant value.

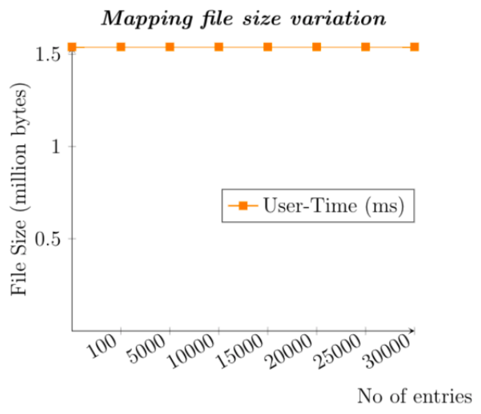
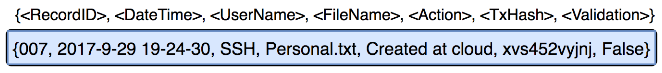


Figure 10: Improved result



{<RecordID>, <DateTime>, <UserName>, <FileName>, <Action>, <TxHash>, <Validation>}

{007, 2017-9-29 19-24-30, SSH, Personal.txt, Created at cloud, xvs452vyjnj, False}

Figure 11: Provenance Entry

5.2 Analysis

This paper deals with the problem of cloud data provenance and it's security. It also discusses existing techniques for the raised problem. In this paper, we have proposed, implemented and analyzed a Blockchain and IPFS based cloud data provenance mechanism which deals with the existing problem of cloud data provenance. Cloud provenance data is collected by using loggers from the OS level of the cloud. Provenance data is stored with the IPFS in a decentralized manner, which makes very difficult for the adversary to locate data of a particular user. The corresponding digest of the provenance data stored in the IPFS is stored in Blockchain, which makes the stored digest immutable.

The proposed model is a unique fully decentralized approach of handling the problem of cloud data provenance and has its own pros and cons. Hence, the results of this model cannot be completely compared with the existing approaches.

Trust among the users and CSSP comes at the cost of storing *mapping.txt* file at the cloud users end, which is an extra overhead that this model offers. As explained in section 4.5, this file will be of constant size. The cost of running this model is proportional to the security it provides. This algorithm uses, SHA-256 algorithm for finding the digest values of different entities. Other low-cost digest algorithms could be used to implement this algorithm at the cost of low security. This model uses a public version of IPFS for storing the provenance entry. The public version of IPFS has its own problems, as it's still in the development phase.

6 CONCLUSION

In this paper, we have presented the design and implementation of a Blockchain and IPFS based cloud data provenance storage and validation mechanism. This proposed technique is fully decentralized in nature unlike the presently available centralized mechanisms for providing trust among the user and CSSP. Our proposed architecture solves the problem by decentralizing the whole architecture. This decentralization pulls the control from the hand of CSSP to provide fair service.

REFERENCES

[1] Git. https://git-scm.com/ [Online accessed: 13-March-2019].
[2] Tierion's Hash API. https://tierion.com/ [Online accessed: 13-March-2019].
[3] M. Asghar, M. Ion, G. Russello, and B. Crispo. Securing Data Provenance in the Cloud. *Open Problems in Network Security*, pages 145–160, 2012.
[4] J. Benet. IPFS-Content Addressed, Versioned, P2P File System. *arXiv preprint arXiv:1407.3561*, 2014.
[5] V. Buterin et al. A Next-Generation Smart Contract and Decentralized Application Platform. *white paper*, 2014.
[6] C. Cachin. Architecture of the Hyperledger Blockchain Fabric. In *Proceedings of the Workshop on Distributed Cryptocurrencies and Consensus Ledgers*, 2016.
[7] J. Freire, D. Koop, E. Santos, and C. T. Silva. Provenance for Computational Tasks: A Survey. *Computing in Science & Engineering*, 10(3), 2008.
[8] X. Liang, S. Shetty, D. Tosh, C. Kamhoua, K. Kwiat, and L. Njilla. ProvChain: A Blockchain-Based Data Provenance Architecture in Cloud Environment with Enhanced Privacy and Availability. In *Proceedings of the 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, pages 468–477, 2017.
[9] D. Mazieres. *Self-certifying File System.* PhD thesis, Cambridge, MA, USA, 2000. AAI0802720.
[10] S. Nakamoto. Bitcoin: A peer-to-peer electronic cash system, 2008.
[11] C. H. Suen, R. K. Ko, Y. S. Tan, P. Jagadpramana, and B. S. Lee. S2logger: End-to-end data tracking mechanism for cloud data provenance. In *Proceedings of the 12th IEEE International Conference on Trust, Security and Privacy in Computing and Communications*, TrustCom'13, pages 594–602, 2013.
[12] S. Sultana and E. Bertino. A file provenance system. In *Proceedings of the third ACM conference on Data and application security and privacy*, pages 153–156, 2013.
[13] Z. Zheng, S. Xie, H. Dai, X. Chen, and H. Wang. An Overview of Blockchain Technology: Architecture, Consensus, and Future Trends. In *Proceedings of the IEEE International Congress on Big Data*, BigData Congress'17, pages 557–564, June 2017.