International Journal of Engineering Research and Technology. ISSN 0974-3154, Volume 13, Number 12 (2020), pp. 4245-4250
© International Research Publication House. http://www.irphouse.com

Performance Evaluation of Data Provenance System with Blockchain-Based Cloud Environment

Yurim Kwon[1], Eun-Kyu Lee[2] and Junghee Jo[3]

[1,2] *School of Information Technology, Incheon National University, Korea (South).*
[3] *Department of Computer Education, Busan National University of Education, Korea (South).*
ORCIDs: 0000-0003-1459-5458 (Eun-Kyu Lee), 0000-0002-9068-3620 (Junghee Jo)

Abstract

With regard to data source verification, recent researches that use the blockchain technology for data provenance only consider registered files. This paper proposes a data provenance system that ensures to verify all the change history of data file on cloud environment. A technical challenge in the system is how to minimize latency required to share files on the cloud. In particular, the latency is primarily affected by the unit for one transaction record. The proposed system takes into account three methods: (i) a file is considered as transaction record, (ii) a file's metadata is determined as transaction record, and (iii) an activity record, a unique characteristic of the cloud, is judged as transaction record. We implement all of them, run experiments, and compare their impacts on latency. Based on results, this paper classified a suitable scenario for each method, and the cloud-based method showed the best performance.

Keywords: Data provenance, Cloud, Blockchain, Security, Internet of Things, Distributed system

I. INTRODUCTION

In a modern digitalized society, it becomes common for people to exchange various files via the Internet. If the source is unclear, however, there exists a risk of malicious code or a copyright problem. In particular, copyright infringement (a.k.a., piracy) has been a critical social issue for a while and is expected to be more challenging because a number of digital contents are being generated and consumed by individuals every day. Korea Copyright Commission reported that 1,763 million copies and 204 million copies of pirates in online and offline markets, respectively, were found in Korea in 2018 [1]. It also presented that price of such pirates reached to 377,009 million Korean Won (KRW) in total (351,794 million KRW from online and 25,215 million KRW from offline). The report also recorded the number of online monitoring that referred to an action by content providers to stop copying and transmitting illegal copies, which was less than 1% of total numbers of the pirates. To resolve the issue, recent researches have proposed solutions that used a blockchain technology to guarantee the copyright of files stored on the cloud. Fig. 1 pictures a general concept of data sharing on the cloud. The cloud allows users to share files in a convenient way. Alice creates a file on the cloud that can be easily accessed and changed another user, Bob. Eventually, the file remains changed and all the change history is logged. However, files are distributed through various channels. To ensure the history of these files, limit them to cloud environments. Moreover, most of previous researches managed only one file registered in a system.

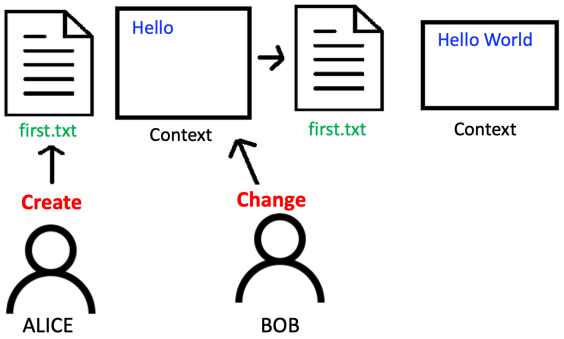

Fig. 1. The cloud allows users to share files in a convenient way. Alice creates a file on the cloud that can be easily accessed and changed by another user, say Bob.

In order to tackle the limitation, this paper proposes a data provenance system that ensures to verify all the change history of data file on cloud environment. A technical challenge in the system is how to minimize latency required to share files on the cloud. In particular, the latency is primarily affected by the unit for one transaction record. Because of increasing volume of information to be processed, it becomes a new technical challenge to minimize latency required to share files on the cloud. Latency is primarily affected by the unit for one transaction record. To scrutinize the performance metric, this paper takes into account three methods: (i) a file is considered as transaction record, (ii) a file's metadata is determined as transaction record, and (iii) an activity record, a unique characteristic of the cloud, is judged as transaction record. We compare their impacts on latency performance via extensive experiments. Results demonstrate that there is a suitable environment for each method, and the cloud-based method shows the best performance.

This paper is organized as follows. Section 2 gives a brief introduction to a blockchain technology. Section 3 reviews research works related to integration of blockchain with cloud storage. Section 4 designs and develops the proposed data provenance system, which is followed by description of delay issues in file sharing on cloud with preliminary experimental results. Section 6 explains our experiments and results. Finally, Section 7 concludes the paper.

II. BACKGROUND

II.I Blockchain

Blockchain is a distributed ledger technology that ensures the reliability of the ledger without a third trust agency. It distributes the ledger of transaction information generated between participants to the network rather than the central server of a particular institution as shown in Fig 2.

The blockchain technology composes of three main components: Miners, consumers who trade, and nodes that are network participants who verify and approve their transactions.

Relevant information is generated once a user takes an action, e.g., generation and copy of a file, in the cloud. Transmitting this information to one node initiates validation by broadcasting to adjacent nodes. When the verified transaction information shows a miner who has gained the right to create a block, other transaction records are combined to create one block. The generated blocks are connected to the previous blocks and are called blockchain. Because blockchain is a distributed data storage technology that is not stored on a specific server, no one can falsify or change blocks [2].

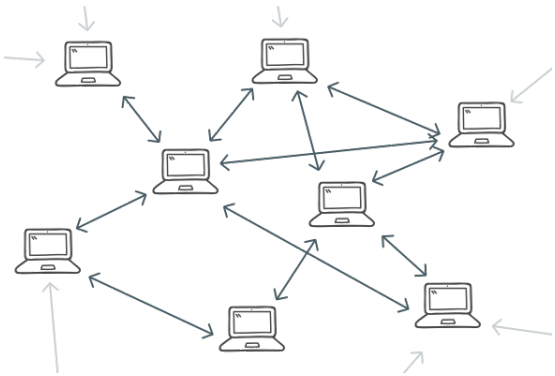

Fig. 2. Blockchain is a distributed data storage that operates in steps of transaction, block generation, verification and connecting the block to a chain.

Blockchain receipts provide evidence that some data existed at a certain time. The receipt contains all the information needed to prove that the individual hash is part of the Merkle tree [3]. It is possible to track the path from the Merkle root to the target hash to generate the Merkle evidence proving that one of the elements is in the Merkle tree without having to know the entire tree.

II.II SHA-256 Hash and Merkle Tree Root

Secure Hash Algorithm (SHA) is a standard hash function published by the National Institute of Standards and Technology (NIST) as a U.S. Federal Information Processing Standard (FIPS). Out of the algorithm family, this paper employs SHA-256 that uses 32-bit words. In SHA-256, input data is divided into blocks by 512 bits in padding blocks, and each block is further divided into 16 32-bit words through pre-processed blocks. The K constant block is a block with an array of constants used in the computation with W and has 64 32-bit constants, providing one K value in hash blocks for each round. The hash computation block is provided with the K-constant and W-word values required for the computation in the K-constant block and the pre-processed block for each round to perform 64 round operations. At the end of the operation, the final hash value of 256-bit length is printed.

A Merkle tree is a hash of all transactions contained by the blockchain and is stored in disk space. Its operation repeats the process of hashing and re-hashing the multiple transactions that make up the block in pairs. In this process, it is summarized as a single hash called a Merkle root as shown in Fig. 3. Thus, it is able to detect attempts to fabricate transactions.

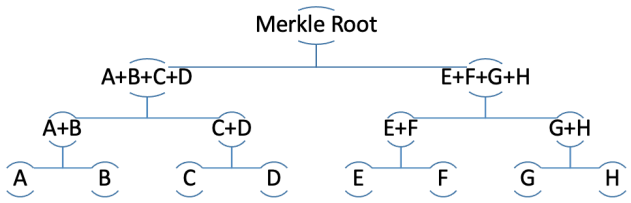

Fig. 3. A Merkle tree hashes all transactions contained in blockchain, which is eventually summarized by a root node.

III. RELATED WORKS

Authors in [4] proposed ProvStore, the first online public repository to support W3C's new PROV standard. It allowed users and applications to store the provenance of their data on the web. The RESTful API (OAuth support) enabled the conversion, visualization, and sharing of provenance documents into various serializations available in third-party applications. Eckert et al. presented a workflow model implemented in accordance with the principles of RESTful Web Services and linked data principles [5]. They showed that RDF-based specifications for Web services, workflows, and runtime information established a complete chain of provenances for all resources created within these workflows. Tosh et al. discussed design problems of consensus protocols for a blockchain-based cloud provenance system [6]. In particular, they highlighted performance and security issues when adopting various Proof of Work Agreement protocols within a data provenance framework.

Sultana and Bertino proposed a provenance model that could express the source of all data objects captured in any abstract layer and present the abstract schema of the model [7]. The expressive characteristics of the model have the advantage of enabling various source queries. They introduced a data provenance middleware system consisting of several components and services that could capture provenances according to the model and store them safely in a central repository. Liang et al. designed and developed ProvChain, a distributed and reliable cloud data provenance architecture using blockchain technology [8]. Consisting of three operation steps of (1) collecting provenance data, (2) storing provenance data, and (3) validating provenance data, it used cloud storage scenarios and selected cloud files in data units to detect user activity to collect data from sources. Xia et al. proposed MeDShare, a data sharing system based on blockchain [9]. It provided data provenance, audit, and control of healthcare data in a cloud repository shared between large data entities. All the actions on the system including transmitting and sharing data from one entity to another were regarded in a tamper-proof manner. MeDShare were able to track data behaviors, to detect violation of permissions on data by using smart contracts and access control mechanisms, and to revoke access to problematic entities if found. Li et al. developed a method that could track a full life cycle of data by using blockchain techniques [10]. Authors, in particular, focused on data transmissions and data regenerations between cloud data centers. Authors in [11] developed a digital forensic architecture that used blockchain, smart contract, and software defined networking (SDN) technologies. To solve integrity and reliability problems in centralized forensic management, they

proposed secure ring verification based authentication scheme that could ensure authorized user accesses only. Hasan et al. proposed a framework that could store provenance information in a secure manner using blockchain technology and InterPlanetary File System (IPFS) [12]. Without relying on the trust of a cloud storage provider, it allowed users to check integrity of own data. Roemsri and Hewett addressed user authentication in the cloud environment [13]. They proposed a new authentication method that used data provenance and location information of a user. Yue et al., proposed a blockchain-based data integrity checking framework that worked on a decentralized network such as a peer-to-peer cloud storage [14]. Consisting of three components of the blockchain, a cloud storage service provider and a data integrity service, it used Merkle trees for data integrity verification.

Sharma et al. presented a systematic survey that reviewed applications of blockchain technology for cloud storage security [15]. Nguyen et al. also reported a similar technical review where authors addressed Cloud of Things (CoT) enabled by the combination of cloud computing and Internet of Things [16]. They surveyed blockchain applications and related development in CoT.

IV. PROPOSED SYSTEM - DESIGN & DEVELOPMENT

IV.I Design

In the proposed system, a cloud server provides access to files, allowing users to share files between them. Fig. 4 illustrates an overall architecture of the system. When a user uploads a file to the cloud, the server generates a SHA-256 hash of the file and sends it to a node that is currently active in the block chain network. Here, a file can be hashed in three different ways; (i) hashing a file entirely, (ii) hashing metadata of a file, or (iii) hashing a cloud activity of a file. The node transmits the hash value to the blockchain network by broadcasting it to the surrounding nodes. When a block is created after verification on the network, a blockchain receipt is returned to the cloud and stored. Later, the blockchain receipt can be used to access the blockchain network to verify that the hash record is valid.

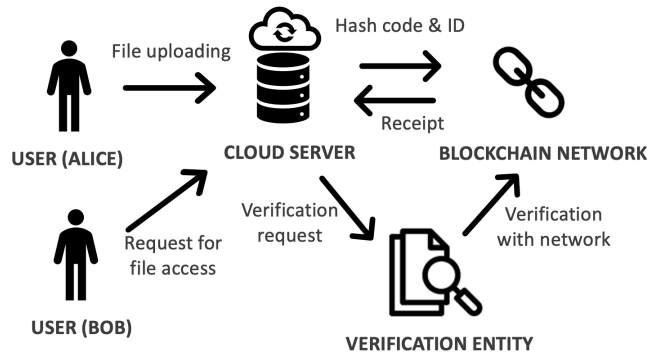

Fig. 4. An overall architecture of the proposed data provenance system; when a user uploads a file, the cloud server generates a corresponding hash code that is then delivered to the blockchain network for future verification.

In the proposed system, the blockchain network frequently communicates with the cloud and the verification entity. The followings describe the details of the communications.

(a) To Request a list of blockchain nodes currently active in the network. Because the blockchain network operates in a decentralized manner, it is necessary to know which nodes are currently active inside the network. To this end, the cloud sets url using a HTTP API method to access the blockchain network. Accessing the url of the blockchain network returns a list of active nodes in JSON format. To reduce delay in this process, we employ a cache prefetching technique that is explained further later in this section.

(b) To publish a hash value to the blockchain network (*/hashes*). The cloud generates a variable *json_hash*, a hash value representing a transaction on a target file, creates a message including the variable, sends the message *(/hashes* interface) to an active node of the blockchain network that is selected from the list received in step (a). This returns a result array that contains information about when the hash value is registered in which anchor as well as a variable *hash_id_core* that is necessary when issuing a receipt later.

(c) To issue a receipt of a transaction (*/proofs*). Using *hash_id_core* obtained in set (b), the cloud is able to send a HTTP GET message to a url, "http://blockchain_node/proofs/hash_id _core" in order to obtain a corresponding receipt. The blockchain network returns a proof encoded by Base 64. The proof value contains contents of the blockchain receipt: Hash value of the transaction (*targetHash*), the height of the block, the hash value of the previous block, the Merkle root, and the transaction ID [17].

(d) To verify the receipt (*/verify*). The cloud or the verification entity can check validation of the receipt received in step (c). The Merkle tree, taking the form of a binary tree, is created by adding each transaction record to the hash operation. In order to confirm that the target hash is included in the blockchain, therefore, we can compute it first with the hash record on a sibling node of the target hash. An outcome can be computed with a hash record of its sibling node, and this computation is repeated until a value is finally left. If the final outcome is same to the Merkle root value on the receipt, the receipt is proved to be valid.

The proposed data provenance system provides users with the following capabilities when sending and receiving files in the cloud.

(a) Guaranteeing the source of the file in real time: If a user takes a file-related action in the cloud, information is sent directly to a blockchain network to ensure the source of the file in real time.

(b) Anti-modification: History information of the file is collected and sent to the blockchain network that protects the source data. All data in the blockchain is shared among nodes. The cloud creates timestamp logs for all user operations on cloud data. All source items are assigned blockchain receipts for future validation.

(c) Source verification: Source records are posted on the blockchain network worldwide, and numerous blockchain nodes provide verification for all blocks. Thus, it is possible to check all sources using the blockchain receipt.

(d) Copyright guarantee: It uses history information of files to track files changes, and guarantee copyright. If a file is downloaded out of the cloud, copyright is guaranteed by posting the information of the file in the blockchain network.

IV.II Development

This paper implements the proposed system on top of cloud environment. To this end, we employ an open source cloud server, ownCloud software [18, 19], and build it on a Linux Ubuntu system. We modify it so that it is able to record all the activities on files and thus to make logs of such activities.

Our system uses a blockchain API called Chainpoint for the cloud data server [20]. Table 1 lists APIs for communicating with the blockchain network. Chainpoint is an open standard for creating timestamp attestation of data, files, or processes. It connects the data hash to the blockchain and returns the timestamp proof. Nodes in it use the Merkle tree to receive a hash and encrypt data to connect to the bitcoin blockchain. A Proof contains information to ensure that the hash of some data is fixed to the blockchain. Thus, it can be demonstrated that the data existed at a particular point in time. Chainpoint provides API via HTTP request and offers Calender and Bitcoin as anchor. Bitcoin is an online crypto that is based on blockchain technology. It takes about 90 minutes for Chainpoint to post a hash with Bitcoin. The useful thing is Calender. Developed for application to embedded system, it makes of being able to publish in less than 10 seconds.

Table 1. Chainpoint provides HTTP-based APIs to communicate with the blockchain network.

Function	GET/ POST	Usage
To publish a hash value to the blockchain network	POST	/hashes
To receive a receipt from the blockchain network	GET	/proofs{hash_id_core}
To verify validation of a receipt	POST	/verify
To search transactions included in the calendar	GET	/calendar/{tx_id}/data
To fetch information of the latest block generated locally	GET	/recent
To fetch information of a blockchain node	GET	/config

Since the blockchain provides API via HTTP, we update our storage server by installing Apache Web Server. In this way, the cloud server is able to communicate with the blockchain network.

IV.III Modification for improved delay performance - prefetching

The proposed system employs a cache prefetching technique that speeds up the execution of a program by bringing data from memory in advance before a processor needs it. That is, the prefetching reduces memory access latency. Using the technique, our system is designed and developed to reduce the number of GET/POST API calls, resulting in reduced service time. We also employ link prefetching, a browser-provided feature that improves site performance by specifying files that need to be preloaded on a website and making them easy to use on demand.

V. DELAY ISSUE IN FILE SHARING ON CLOUD

Once a transaction occurs, blockchain sends this event information to the blockchain network. Then, it is important what you see as a “one deal” in a system that manages file sources using the cloud. This is the hash of the file mentioned in the previous section. To capture a delay issue in the process, this section conducts a preliminary experiment and records its result in Fig. 5. It shows time duration when a file is uploaded to a typical cloud without adding blockchain functionality. As shown, it takes up to more than 4 seconds to upload a file of 512 KB to the cloud. We note that the uploading delay may be different on different cloud settings because they are on various network conditions and use different optimization techniques. This paper uses the preliminary results as base data to evaluate performance of blockchain-related operations.

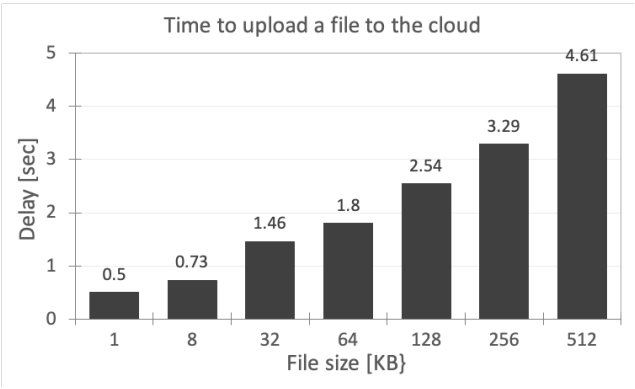

Fig. 5. Time [sec] to upload a file to the cloud without adding the blockchain hashing.

To resolve the delay issue, this paper tries to minimize time to share files on the cloud. Our system, in particular, takes into account three methods. At the first method, a file is considered as transaction record, that is, the entire file is hashed regardless of its size. Second, a file's metadata is determined as transaction record. Since metadata is only hashed, this method can reduce processing delay. Last, an activity record, a unique characteristic of the cloud, is judged as transaction record.

VI. EXPERIMENTS AND RESULTS

This research makes use of Apache JMeter to measure delay performance of the proposed data provenance system. Apache JMeter is an open source software based on Java application designed for performance testing of client-server-structured software like Web applications [21]. It has been used in many places, such as unit/performance/stress tests. JMeter supports general-purpose protocols such as TCP, FTP, and HTTP(S); it only sends and receives messages between the client and the server in line with the communication protocols. It does not work on web browsers and does not perform operations performed by the client itself.

Using JMeter, this section conducts experiments that measure time to upload a file to the cloud after running the blockchain hashing with three methods mentioned in the previous section. Table 2 summarizes the experimental results.

Table 2. Time [sec] to upload a file to the cloud after running blockchain hashing

File size [KB]	1	8	32	64	128	256	512
Delay [sec] with A	1s	2.48s	4.12s	6.8s	8.25s	15.4s	24s
Delay [sec] with B	0.88s	1.25s	2.92s	3.13s	4.79s	5s	6.4s
Delay [sec] with C	0.88s	1.06s	2.26s	2.97s	3.41s	4.66s	5.3s

VI.I Hashing a File Entirely (A)

This method considers a file to be a transaction. Table 2 shows the upload time by the file size when a file itself is hashed. It also includes the speed of the upload process at the rate at which the file is uploaded and hashed in the SHA-256 format until the result is reached. For example, suppose the test.txt file has a string called "Hello World!" stored. If another user clears "!" then test.txt is considered to be a different file and is re-hashed. The larger the size of the file, the more pressure it is in terms of speed. Fig. 6 compares the delay performance of method A (hashing an entire file before uploading it to the cloud) to those cases of not hashing files. The figure presents that the delay value grows exponentially as a file size increases when an entire file is hashed. Delay increases up to longer than 24 seconds even (6 times longer comparing to the base case) when using a file of 512 KB size, which is generally not acceptable to end users in a real world. In this sense, the first method cannot be applied to a cloud-based data provenance system directly. However, if a similarity testing process is included and user names and timestamps are added, they are appropriate for examples related to copyright using blockchain, such as Proof of Experience (PoE) [22].

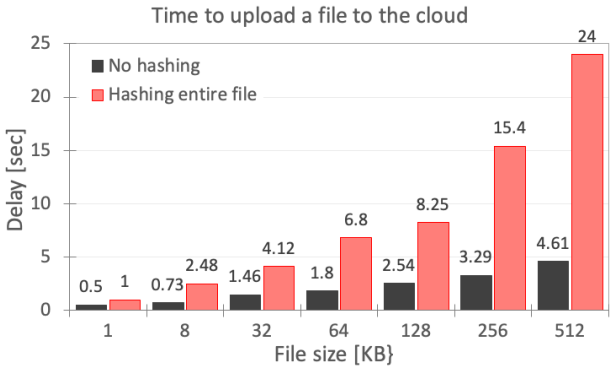

Fig. 6. Hashing an entire file and then uploading it to the cloud add a long processing delay. Thus, method A, as it is, is not appropriate to the cloud environment.

VI.II Hashing Metadata of a File (B)

When a user uploads a file, the file's metadata is hashed in method B. The third row in Table 2 shows its performance output. As above, the total upload time includes the time it takes to upload the file. Compared to hashing files, this method takes shorter time to upload files. Metadata of data may include author, file name, file size, timestamp, and file type. Therefore, even if the contents of the file are different, the same metadata is treated as the same file.

VI.III Hashing Cloud Activity of a File (C)

This hash method is a possible way to work in a cloud environment. Most clouds keep track of user activity. This is a way to report and hash activity records as a transaction. For example, "A user named Alice uploads a file named addressbook.txt at 19:35 on November 10, 2020" is interpreted as an activity record and treated as a transaction. As with the other cases, the file upload process is also included. It's much faster than hashing the file itself, and a bit faster than hashing the metadata. It's hard to say that it's related to the file. If users X and Y upload the same file separately, the hash value of the record will be different, so both can be identified as sources.

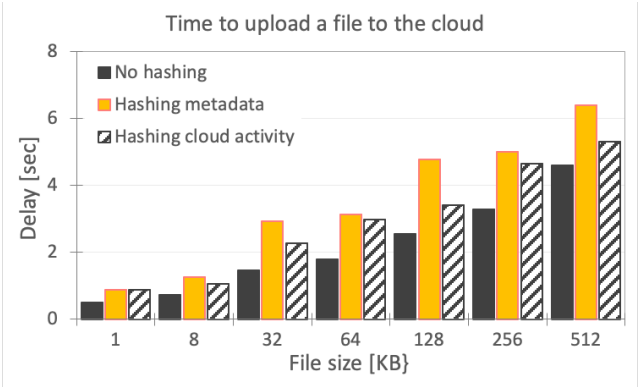

Fig. 7. File hashing takes much longer than other as data size increases. Two hashing methods do not introduce overhead much comparing to base delay that uploads files without blockchain hashing.

Fig. 7 compares the results of the experiments at once. The X-axis shows the file size [KB] while Y-axis represents the time taken [seconds]. The figure presents that the delay value grows almost linearly as a file size increases in both methods B and C. In the case of method B (hashing metadata), there are 74% of more overhead with 64 KB of data, but this value decreases to 39% when data is 512 KB in size. It is expected that the overhead decreases as the file size increase. To capture our expectation, this section represents these overheads in the format of seconds as follows. It takes 1.33 more seconds to process metadata hashing with 64 KB of data, and this value increases to 1.71 and 1.79 seconds with 256 KB and 512 KB of data. Since method B introduces hashing overhead of only meta information of a file, and we do not expect that the size of meta information increases along with the file size, we can conclude that the overhead becomes smaller as the file size gets bigger. It is also expected that the overheads in method B vary according to file types; a specific file type including a large size metadata will result in weighted overhead. In the case of method C (hashing cloud activities), there are 65% of more overhead with 64 KB of data, but this value decreases to 15% when data is 512 KB in size. Except for two cases of smallest data size (1 KB and 8 KB), the average overhead is 0.98 seconds, which is shorter than delay values in method B. It is also expected that the overheads in method C does not vary much unlike the case of method B. This is mainly attributed to the fact that contexts of cloud activities are somewhat limited and small in size.

Based on observation, this paper recommends that the second method (hashing metadata of a file) is suitable for the purpose of copyright protection while the third method (hashing cloud activities of a file) works well for tracking the history of files.

VII. CONCLUSION

In general, it is recommended to hash a file in order to prevent forgery of a file itself and its origin. In a file sharing environment like the cloud, however, this paper observes that it is suitable to hash metadata or activity records for data proceeding. The problem mentioned in the case of hashing the activity record can be solved with a blockchain receipt. The blockchain receipt also includes a timestamp so that one can identify who created data earlier.

ACKNOWLEDGMENT

This research was supported by Incheon National University Research Grant in 2019. Authors thank to Jueun Yoo for efforts to this research. E.-K. Lee (eklee@inu.ac.kr) is the corresponding author.

REFERENCES

[1] Korea Copyright Commission. Available online: https://www.copyright.or.kr/ [accessed on 11/24/2020]

[2] Becoming a blockchain node. Available online: https://brunch.co.kr/@mobiinside/1186 [accessed on 11/24/2020]

[3] R. Merkle, "A Digital Signature Based on a Conventional Encryption Function," Advances in Cryptology, vol. 293, pp. 369-378, 1987.

[4] T. D. Huynh and L. Moreau, “ProvStore: A Public Provenance Repository,” International Provenance and Annotation Workshop (IPAW), June 2014.

[5] K. Eckert, D. Ritze, K. Baierer, and C. Bizer, "RESTful open workflows for data provenance and reuse," ACM International Conference on World Wide Web, April 2014.

[6] D. K. Tosh, S. Shetty, X. Liang, C. Kamhoua and L. Njilla, "Consensus protocols for blockchain-based data provenance: Challenges and opportunities," IEEE Ubiquitous Computing, Electronics and Mobile Communication Conference, October 2017.

[7] S. Sultana, and E. Bertino, "A Distributed System for The Management of Fine-grained Provenance," Journal of Database Management, 26(2), pp. 32-47, April, 2015.

[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," IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, May 2017.

[9] Q. Xia, E. Sifah, K. Asamoah, J. Gao, X. Du, and M. Guizani, "MeDShare: Trust-Less Medical Data Sharing Among Cloud Service Providers via Blockchain," IEEE Access, vol. 5, pp. 14757-14767, July 2017.

[10] H. Li, K. Gai, Z. Fang, L. Zhu, L. Xu, and P. Jiang, "Blockchain-enabled Data Provenance in Cloud Datacenter Reengineering," ACM International Symposium on Blockchain and Secure Critical Infrastructure, July 2019.

[11] M. Pourvahab and G. Ekbatanifard, "Digital Forensics Architecture for Evidence Collection and Provenance Preservation in IaaS Cloud Environment Using SDN and Blockchain Technology," IEEE Access, vol. 7, pp. 153349-153364, 2019,

[12] S. S. Hasan, N. H. Sultan, and F. A. Barbhuiya, "Cloud Data Provenance using IPFS and Blockchain Technology," ACM International Workshop on Security in Cloud Computing, July 2019.

[13] P. Roemsri and R. Hewett, "Provenance Location-based Authentication in Cloud Computing," ACM International Conference on Advances in Information Technology, July 2020.

[14] D. Yue, R. Li, Y. Zhang, W. Tian, and C. Peng, "Blockchain based data integrity verification in p2p cloud storage," IEEE International Conference on Parallel and Distributed Systems, Dec. 2018.

[15] P. Sharma, R. Jindal, and M. D. Borah, "Blockchain Technology for Cloud Storage: A Systematic Literature Review," ACM Computing Surveys, 53(4), pp. 1–32, 2020.

[16] D. C. Nguyen, P. N. Pathirana, M. Ding and A. Seneviratne, "Integration of Blockchain and Cloud of Things: Architecture, Applications and Challenges," IEEE Communications Surveys & Tutorials, 22(4), pp. 2521-2549, 2020,

[17] Gateway HTTP API. Available online: https://github.com/chainpoint/chainpoint-gateway/wiki/Gateway-HTTP-API [accessed on 11/24/2020]

[18] ownCloud. Available online: https://owncloud.org/ [accessed on 11/24/2020]

[19] ownCloud Installation. Available online: https://doc.owncloud.org/server/10.3/admin_manual/installation/ [accessed on 11/24/2020]

[20] Chainpoint. Available online: https://chainpoint.org/ [accessed on 11/24/2020]

[21] Apache JMeter™. Available online: https://jmeter.apache.org/ [accessed on 11/24/2020]

[22] Proof of Existence. Available online: https://en.wikipedia.org/wiki/Proof_of_Existence [accessed on 11/24/2020]