A blockchain-based log storage model with efficient query

Gang Xu (gangxu_bupt@163.com)
North China University of Science and Technology
Fan Yun
North China University of Technology
Yiying Yu
North China University of Technology
Shiyuan Xu
North China University of Technology
Xiu-Bo Chen
Beijing University of Posts and Telecommunications
Mianxiong Dong
Muroran Institution of Technology

Research Article

Keywords: blockchain, log storge, IPFS, Merkle tee

Posted Date: June 2nd, 2022

DOI: https://doi.org/10.21203/rs.3.rs-1331640/v1

A blockchain-based log storage model with efficient query

Gang Xu[1], Fan Yun[1], Yiying Yu[1], Shiyuan Xu[1], Xiu-Bo Chen[2*] and Mianxiong Dong[3]

[1]School of Information Science and Technology, North China University of Technology, Street, Beijing, 100144, State, China.
[2]Information Security Center, State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Street, Beijing, 100876, State, China.
[3]Muroran Institute of Technology, Organization, Street, Muroran, 050-8585, State, Japan.

*Corresponding author(s). E-mail(s): flyover100@163.com;
Contributing authors: gangxu_bupt@163.com; 1752572892@qq.com; 1244685667@qq.com; 13501199447@163.com; mx.dong@csse.muroran-it.ac.jp;

Abstract

The server logs play an essential role in website maintenance. When the Internet server is running, the service maintained may encounter various problems, and the logs generated by the server are the key data to locate the problems. However, the existing log storage systems are prone to a single point of failure, which may cause logs to be lost, and the problem cannot be rectified. Meanwhile, external attackers can tamper with logs to evade tracking, and the person in charge of an accident may also tamper with the log to evade responsibility. The advantages of blockchain, such as immutability, decentralization, and security, can effectively overcome these shortcomings. In this paper, we proposed a log data storage scheme based on blockchain and Interplanetary File System (IPFS). The lack of scalability of blockchain for storing large files is addressed by storing the log data stored in IPFS. Firstly, our model uses IPFS to store extensive file log data and combines blockchain technology to realize log data storage decentralization and difficulty to tamper. Secondly, to improve the query efficiency of log data, we propose two improved Merkle tree methods to provide fast queries of log data with a timestamp as query keywords. Finally, the experimental results show that the two methods have better log retrieval efficiency.

Keywords: blockchain, log storge, IPFS, Merkle tee

1 Introduction

Log data can help administrators discover vulnerabilities in the system (Sur, 2019). The server downtime or malicious invasion of the server will cause the log data to be lost, and it is difficult to analyze the reasons or investigate the responsibility after the problem occurs. Sending log data to trusted third-party management or a distributed log system to aggregate these files is now the mainstream log storage method (He et al, 2010; Son et al, 2017). However, these servers are vulnerable to accidental data damage or malicious attacks, not guarantee log data security. On the one hand, the management of the log database is usually performed by users with permissions, and semi-honest

users with permissions can tamper with the log data in the log database, which cannot guarantee the security of the log data. On the other hand, external attackers can illegally manipulate the system, and log data may be tampered with, causing economic losses (Faradzhullaev, 2008).

The emergence of blockchain technology provides a new idea for log data storage security (Kumar et al, 2018; Nakamoto, 2008). Blockchain technology is now applied to various fields, such as electronic health record Uddin et al (2021), food traceability Jiuliang Liu (2020), etc. The decentralized distributed architecture of blockchain technology makes it possible to establish a log storage system independent of trusted third parties. However, if the log data is directly stored in the blockchain, it may limit its large-scale application since the traditional blockchain has limitations in capacity and scalability (Yue et al, 2020). Recently, (Shekhtman and Waisbard, 2021) considered the storage capacity of one blockchain, uploading ample file storage to the cloud, only saving the hash value of the file to the blockchain, ensuring the integrity of the file. However, they did not guarantee the authenticity of files when they were compromised or maliciously tampered with within a cloud server. Fortunately, Interplanetary File System (IPFS) provides a new solution for storing large file log data (Benet, 2014), which provides storage scalability and high availability through replication through generating a unique hash value for each file. Users can obtain the file's content through the hash value, and the uploaded data are almost impossible to modify and delete.

In this paper, we propose a blockchain-based log storage scheme, which uses IPFS to store log data and utilizes an improved Merkle tree to improve log query efficiency. In full generality, we outline our contributions as follows:

Firstly, we propose a decentralized log storage scheme, which eliminates the traditional centralized log storage scheme. This scheme utilizes IPFS to address the need to store a large number of logs. By storing log data into the IPFS server, the storage pressure of blockchain alleviates.

Secondly, we improve the traditional Merkle tree and propose two enhanced schemes: the baseline and the enhanced method. Both of them support finding data by log timestamp on the blockchain.

Thirdly, we finished the simulation of the proposed two methods in the implementation.

The main structure is as follows: In section 2, we discuss work related to log storage. We design a log storage scheme based on blockchain and IPFS in section 3. Further, in section 4, we describe two methods of the Merkle tree. Then, we compare the query performance of the two schemes through comprehensive simulation in section 5. Finally, in section 6, we summarize the full text.

2 Related Works

In order to avoid the problem that centralized log storage is vulnerable to attack, researchers have proposed some blockchain-based log storage schemes. The security of log data can be protected by using the immutability property of blockchain (Ahmad et al, 2018; Putz et al, 2019; Xu et al, 2022). (Gürsoy et al, 2020) realized the storage and query scheme of genomic access log files based on the blockchain using a multi-chain platform. In their scheme, inquirers need to download all data from the chain and save them in local memory for fast queries. To solve this problem, they designed another 'bigmem' solution, which uses index instead of local storage for fast queries. (Shekhtman and Waisbard, 2021) added an additional encryption layer to the blockchain, encrypting data before data transmission to the blockchain and protecting the privacy and security of data between authorized parties. (Pourmajidi and Miranskyy, 2018) designed a super blockchain with two layers of hierarchical structure. Log data is stored in a circled blockchain, including a genesis block and a terminal block where each superblock contains a circled blockchain. Obviously, most existing blockchain-based log data storage solutions mainly solve the shortage of centralized log storage. In this article, we will focus on the scalability of blockchain storage and query.

Merkle tree is a data structure, which can realize fast data verification. Changing the value of any leaf node causes the root to change (Buchmann et al, 2008). Merkle tree is the main component of the blockchain and is used to store and verify transactions, ensuring that the blocks are consistent at each node. Most mainstream blockchain frameworks have improved the Merkle tree. Bitcoin uses the traditional Merkle tree (Nakamoto, 2008); Ethereum improved Bitcoin and proposed

Merkle Patricia tree based on the characteristics of efficient retrieval of Patricia tree (Bonneau, 2016); to reduce the cost of adding data, the Hyperledger combines Merkle trees and hash buckets (Androulaki et al, 2018). (Xu et al, 2017) proposed an educational certificate blockchain based on the Merkle Patricia tree, which not only can you provide efficient queries but also support historical transaction queries. (He et al, 2020) proposed a blockchain-based verification scheme for large-scale cloud data and designed a T-tree structure, which supports binary search on the blockchain. (Yue et al, 2018) proposed a general data integrity verification framework for blockchain-based P2P cloud storage. Some multi-branch Merkle trees based on data shards are constructed, and their performance is compared.

3 System Design

3.1 System model

In the model proposed in this paper, log data are stored in IPFS, and the hash tags corresponding to each log data are stored in the chain, which solves the shortage of blockchain in storing extensive file log data. Based on this model, combined with the improved Merkle tree, the log data with timestamp query keywords can be quickly retrieved. The concrete implementation of the improved Merkle tree is in section 4.

Our blockchain-based log storage system model elaborates in Figure 1. The model contains three different entities: client, IPFS server, and blockchain nodes. The clients have many log data that need to be stored in the system safely and can send query requests to blockchain nodes to obtain the query results. IPFS servers store log files received from the clients and generate a unique hash for each log file. Clients and other users can send download requests to servers or nodes according to unique hash tags to get the contents of the log files. Blockchain stores hash values generated by the IPFS server for data integrity verification.

Our proposed scheme includes two stages: the storage stage and the query stage.

Storage Stage: (1) The client encrypts the log data using symmetric encryption and signs them, and uploads the processed log data to the IPFS server. (2) The IPFS server verifies the processed log data to ensure the integrity of log data. The

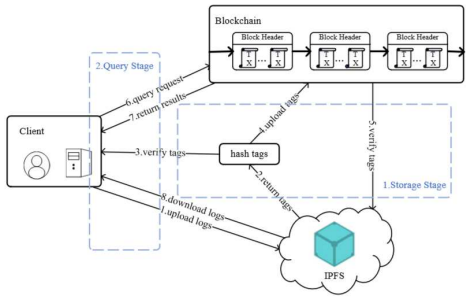

Fig. 1 Blockchain-based log storage system model

log data is then stored, and the generated hash tags are returned to the client. (3) The client verifies the correctness of hash tags with other IPFS servers. (4) The client signs the hash tags and sends the data to the blockchain node. (5) The blockchain node sends a comparison tags verification request to the IPFS node to verify whether the sender of tags is the same as the log data holder in IPFS.

Query Stage: (6) The client sends log query requests to a blockchain node. (7) The blockchain node query the corresponding log hash tag according to the request, then return the data proof and query results to the client. (8) The client verifies the correctness of the data proof to other blockchain nodes. If the data proof proves that the hash tag is intact, the client downloads data from the IPFS server node using the hash tag; otherwise, it indicates that the data tag is incomplete and may be tampered with.

In data transmission, a cryptography algorithm is used to ensure the authenticity, reliability and integrity of information transmission. The sender of the data has to sign the hash value of the data. If the data is tampered with during transmission, the verification signature will not pass, and the data will not be trusted. In the storage stage, the Practical Byzantine Fault Tolerance (PBFT) algorithm is used to complete the consensus process of data blocks (Castro and Liskov, 2002). Suppose there are blockchain nodes to compose a consensus system, the PBFT algorithm provides the fault-tolerant capability of $f = (n-1)/3$, that is, the number of failed or malicious nodes cannot exceed 1/3 of all nodes. IPFS guarantees the integrity of log data.

3.2 The Improved Blockchain Storage Structure

In our scheme, hash tags generated from log data are packaged into blocks by blockchain nodes. In order to adapt to the application scenario where log data needs to be queried by timestamp, this paper extends the block structure, as shown in Figure 2.

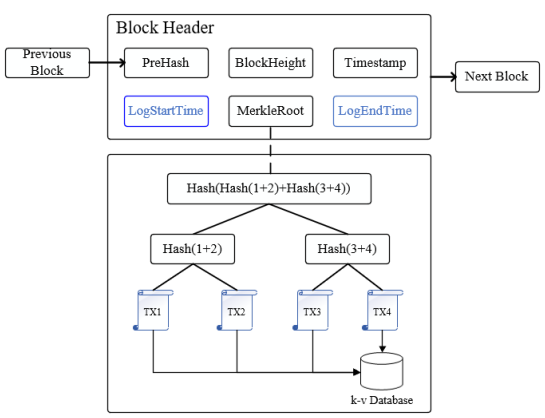

Fig. 2 The improved blockchain architecture

Each block is composed of a block head and a block body (Nofer et al, 2017). The block header contains the hash value of the previous block, the timestamp when generating the block, the Merkle root, block height, and the log start and end time of the transaction list in the block. The data in the transaction list is sorted by log generation time, all of which are stored in the k-v database.

We added the start time and end time of the transaction in the transaction list of the blockchain to the original block header data to improve the search efficiency of the timestamp keyword on the chain. Due to network delay and upload speed, the start time of logs in the current block may be shorter than the end time of the previous block. Given a timestamp, the block containing the timestamp is determined by scanning all the block headers on the blockchain.

4 The Improved Merkle Tree

The transaction data for each block in the blockchain is stored as a Merkle tree. One feature of the Merkle tree is that given a Merkle path, it can quickly verify whether a transaction is in the transaction list. However, when a client wants to query a transaction by keyword, the blockchain node needs to traverse all the transaction data stored in the Merkle tree. As the transaction data in the block increases, the time it takes to query a transaction record on the blockchain increases dramatically. In order to reduce memory and improve query efficiency, the nodes of the Merkle tree are improved in our method. Our method contains the Merkle tree validation features (the client can verify that a transaction is included in the transaction list without downloading all transactions in all blocks) but also improve the timestamp keyword query efficiency on the blockchain and the methods support the timestamp range query. In this section, we describe in detail the structure of the two improved Merkle trees and give the timestamp point query algorithm and timestamp range query algorithm.

4.1 The Baseline Method

The baseline method combines the characteristics of binary search and Merkle tree, and its node structure is shown in Figure 3. There are two types of nodes in this method: branch nodes and leaf nodes. A branch node $Bnode$ of the baseline method consists of the minimum log timestamp value Min, maximum log timestamp value Max, the hash values of the left subtrees $LeftChild$ and right subtrees $RightChild$. A leaf node $Lnode$ contains the transaction value TX and the generation timestamp $LogTime$ of the log data in the transaction. In this method, all nodes are stored in the $k-v$ database, where v is all data connections of nodes and k is obtained by the hash algorithm Hash(v).

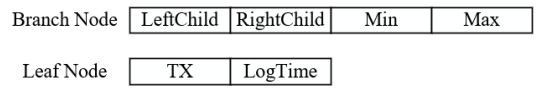

Fig. 3 Basic baseline method data types

In tree constructions, leaf nodes are first constructed according to the transaction list and saved in the database. Then, starting at the bottom of the tree, a new branch node is constructed layer by layer by two nodes of the tree, and if there is an odd number of branch nodes in a layer, the last branch node is copied until the last is the root

node. In a branch node, if its child node is a branch node, Min is the left child's Min, and Max is the right child's Max; if its child node is the leaf node, and its Min and Max are the $LogTime$ of the leaf node.

In the query stage of the blockchain-based log storage model, when a client requests a timestamp single point query of a blockchain node, the blockchain node invokes the query algorithm to search for transaction data. The client sends a timestamp T to the blockchain node. If T is not within the time range of the blockchain, the algorithm returns a null value. Otherwise, find the blocks where T is located by the start and end times in the block header. Then, based on the hash value of the root of the Merkle tree in the block header, the search algorithm is called to obtain TX. In the search process, the traversed nodes are Merkle path, and the blockchain nodes return Merkle path and search results to the client. The Merkle path allows the client to determine whether the transaction is on the blockchain quickly. The timestamp point query algorithm for the baseline method is as follows:

Algorithm 1 Timestamp Point Query Algorithm

Require: timestamp T, Merkle root $Root$

```
nodes=stack()
nodes.append(GetNode(Root))
while nodes is not empty do
    Plist.append(nodes[-1])
    node= nodes.pop()
    if node is leaf node and node.Min==T then
        Hlist.Add(GetValue(node.leftChild))
    else if T ≥ GetNode(node.leftChild).Max then
        nodes.append(GetNode(node.rightChild))
    else if T ≤ GetNode(node.rightChild).Min then
        nodes.append(GetNode(node.leftChild))
    end if
end while
return Hlist, Plist
```

According to the timestamp keyword query feature of log data, when the client sends the start and ends time of the log to the blockchain node, the blockchain node returns the corresponding query results. Gives a start timestamp Ts and an end timestamp Te, timestamp range query algorithm returns the hash tag list within that timestamp range. Assuming that the timestamp ranges of each block have no intersection, the blocks containing Ts and Te are found from the blockchain. From these blocks, we can call the query algorithm to get hash tag list, according to the following conditions:

Case1: Ts and Te are in the same block, call query algorithm.

Case2: Ts and Te are in different blocks. The block transaction list in the middle of the two blocks is added to the hash tag list. The query algorithm is then called on both blocks.

The following query algorithm is given to search for transactions within a given timestamp range in a block:

Algorithm 2 Timestamp Range Query Algorithm

Require: start timestamp Ts, end timestamp Te, Merkle root $Root$

```
nodes=queue()
nodes.append(Root)
while nodes is not empty do
    node = nodes.get()
    if node is leaf node then
        Hlist.Add(GetValue(node.leftChild))
        Plist.append(nodes[-1])
    else
        leftChild = GetNode(node.leftChild)
        rightChild = GetNode(node.rightChild)
        if Ts or Te ≤ leftChild.Max then
            nodes.append(GetNode(node.leftChild))
            Plist.append(nodes[-1])
        end if
        if Ts or Te ≥ leftChild.Max then
            nodes.append (GetNode(node.rightChild)
            Plist.append(nodes[-1])
        end if
    end if
end while
return Hlist, Plist
```

4.2 The Enhanced Method

In the enhanced method, we combine the prefix tree and the Merkle tree to improve the baseline scheme. In this method, we take the log generation time to build a prefix tree. The maximum height of the improved tree is the length of the Unix timestamp under decimal. The basic nodes of the enhanced method are composed of two nodes: branch node, leaf node as shown in Figure 4. A branch node $Bnode$ is a node with a child set $child_i, i \in [0, 9]$ and a prefix value $Prefix$. A leaf node Lnode consists of a prefix value and a transaction list $TX_j, j \in [0, m]$, transactions in each leaf node have the same log generation time.

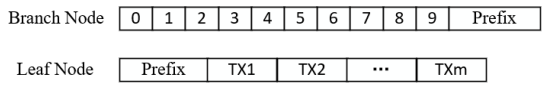

Fig. 4 Basic enhanced method data types

Similar to the baseline, the input of the tree constructions algorithm is a list, and the output is the hash value of the root node, except that the list element UT is A Unix timestamp. Transactions with the same log timestamp are stored in the same leaf node, which greatly reduces the number of nodes that need to be traversed based on the timestamp keyword.

Algorithm 3 Timestamp Point Query Algorithm

Require: timestamp UT, Merkle root $Root$

$node$=GetNode($Root$)
while $node$ is not leaf node **do**
  if UT[len($node.prefix$)] in $node.children$ **then**
    $Plist$.append($node$)
    $node$= $node$.getChild(UT[len($node.prefix$)])
  else
    return $null$
  end if
end while
if $node.prefix$==UT **then**
  for tx in $node.txes$ **do**
    $Hlist$.append (tx)
  end for
end if
return $Hlist$, $Plist$

In the timestamp range query algorithm, whether the prefix of the node is within the given time range is determined. The algorithm is described in detail as follows:

Algorithm 4 Timestamp Range Query Algorithm

Require: start timestamp Ts, end timestamp Te, the Merkle root $Root$

$nodes$=queue()
$nodes$.append($Root$)
while $nodes$ is not empty **do**
  for $temp$ in range($nodes$.size()) **do**
    $node$ = $nodes$.popleft()
    for $childNode$ in $node$.childrenNode **do**
      if Ts [:len($prefix$)] $\leq$ $childNode.prefix \leq Te$[:len(prefix)] **then**
        if $childNode$ is leaf node **then**
          for tx in childNode.txes **do**
            $Hlist$.append(tx)
          end for
        else
          $nodes$.append($childNode$)
        end if
        $Plist$.append($childNode$)
      end if
    end for
  end for
end while
return $Hlist$, $Plist$

5 Experiments

The experiment was performed on Windows 10. The main hardware configuration is as follows: CPU: Intel(R) Core (TM) i7-4720HQ 2.60GHz; RAM:16GB. This paper mainly studies how to improve the blockchain structure and Merkle tree to improve the search efficiency on the blockchain. Consequently, we have not conducted experiments on clusters.

In this paper, we implemented the two improved methods using the python language and compared them with the Merkle tree of Yue's scheme [22]. Yue's scheme [22] constructs some multi-branched Merkle trees based on data shards, each tree needs to download all the transactions in

the block to be queried, so we only design a comprehensive comparison experiment between the Binary Branching tree (BBT) and our methods. We compare the storage structures of blockchain in terms of tree construction complexity, timestamp point query, and range query overhead. We randomly generated one hundred thousand log data, uploaded it to the IPFS server in the private network, and uploaded the hash tags of these logs to blockchain for an experiment. Each block contains the Merkle root of BBT, the baseline, and the enhanced method.

Firstly, we compare the tree generation efficiency versus the number of the size of the transaction list. Assuming that the block's transaction list is from 256 to 8192 (256, 512, 1024, 2048, 4096, 8192). From Figure 5, we can see that the two improved methods' tree construction efficiency is lower than BBT. Due to the two methods, we proposed storing all nodes in the database during the construction process. Furthermore, in the enhanced method, the prefix value of log generation time for all transactions is different in the worst case, resulting in a much larger number of nodes than the other methods.

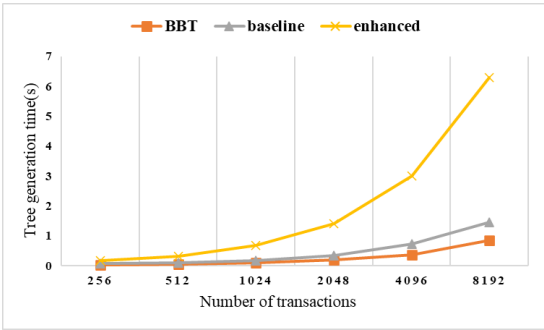

Fig. 5 The comparison of methods and BBT build cost

Secondly, we compare the query efficiency. Considering that timestamps are often used as keywords when searching for logs, we compare the time consumption of point query and range query. We experimented on the blockchain generated by the tree construction experiment, randomly querying 10,000 times and recording the average query time. As shown in Figure 6, the baseline and enhanced methods do not require all transactions to be read in, so timestamp point query times are much lower than BBT. As shown in Figure 7, two methods also have higher query efficiency in timestamp range queries. Fewer tree nodes are generated in the enhanced approach, so it takes less time to query than the baseline method. Therefore, it can be observed that both methods have better query performance than BBT.

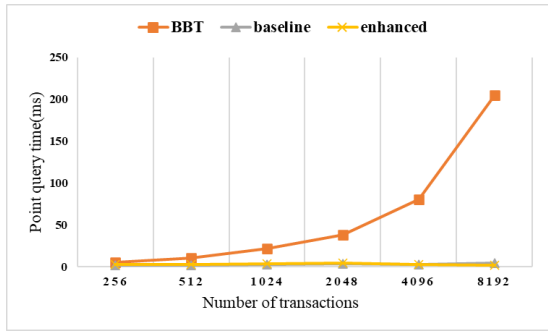

Fig. 6 The comparison of methods and BBT point query cost

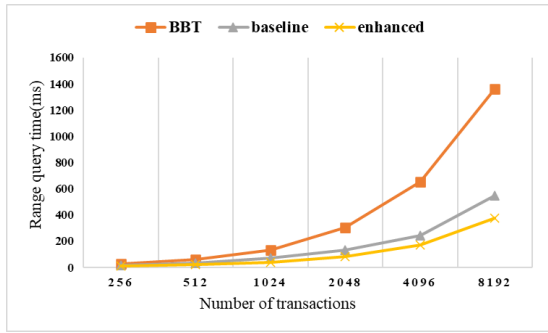

Fig. 7 The comparison of methods and BBT range query cost

In our methods, we build a tree with the timestamp as the keyword and store each node in the database, so the build time of the tree is more protracted than BBT. However, the query overhead is lower than BBT because we have added fields to each node to enable efficient searching.

6 Conclusions

This paper proposes a log data storage scheme based on blockchain and IPFS. This scheme effectively solves the problem of insufficient scalability of blockchain by storing log data in IPFS and hash tags in the blockchain. The structure of the model is introduced and analyzed. Aiming at the

requirement that logs need to be queried by timestamp, we design two improved methods based on the Merkle tree to improve the query rate of logs by timestamp. Experiments show that the two improved methods proposed in this paper can effectively improve the search performance of the query log according to the timestamp.

Author Contributions Conceptualization, Methodology, Validation, Investigation, Writing, Funding acquisition, Formal analysis by G. Xu and F. Yun. Collected data, Data analysis, Review by Y.Y. Yu and S.Y. Xu. Funding acquisition, Visualization, Writing -Review & Editing, Supervision by X. B. Chen and M. X. Dong. All authors read and approved the final manuscript.

Funding This work was supported by the Open Fund of Advanced Cryptography and System Security Key Laboratory of Sichuan Province (Grant No. SKLACSS-202101), NSFC (Grant Nos. 62176273, 61962009, U1936216), the Fundamental Research Funds for Beijing Municipal Commission of Education, Beijing Urban Governance Research Base of North China University of Technology, the Natural Science Foundation of Inner Mongolia (2021MS06006), Baotou Kundulun District Science and technology plan project (YF2020013), and Inner Mongolia discipline inspection and supervision big data laboratory open project fund (IMDBD2020020).

Declarations

Conflict of interest All authors declare that they have no conflict of interest.

Ethical approval This article does not contain any studies with human participants or animals performed by any of the authors.

Informed consent Informed consent was obtained from all individual participants included in the study.

References

Ahmad A, Saad M, Bassiouni MA, et al (2018) Towards blockchain-driven, secure and transparent audit logs. In: Schulzrinne H, Li P (eds) Proceedings of the 15th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services, MobiQuitous 2018, 5-7 November 2018, New York City, NY, USA. ACM, pp 443–448, https://doi.org/10.1145/3286978.3286985

Androulaki E, Barger A, Bortnikov V, et al (2018) Hyperledger fabric: a distributed operating system for permissioned blockchains. In: Oliveira R, Felber P, Hu YC (eds) Proceedings of the Thirteenth EuroSys Conference, EuroSys 2018, Porto, Portugal, April 23-26, 2018. ACM, pp 30:1–30:15, https://doi.org/10.1145/3190508.3190538

Benet J (2014) IPFS - content addressed, versioned, P2P file system. CoRR abs/1407.3561

Bonneau J (2016) Ethiks: Using ethereum to audit a CONIKS key transparency log. In: Clark J, Meiklejohn S, Ryan PYA, et al (eds) Financial Cryptography and Data Security - FC 2016 International Workshops, BITCOIN, VOTING, and WAHC, Christ Church, Barbados, February 26, 2016, Revised Selected Papers, Lecture Notes in Computer Science, vol 9604. Springer, pp 95–105, https://doi.org/10.1007/978-3-662-53357-4_7

Buchmann J, Dahmen E, Schneider M (2008) Merkle tree traversal revisited. In: Buchmann J, Ding J (eds) Post-Quantum Cryptography, Second International Workshop, PQCrypto 2008, Cincinnati, OH, USA, October 17-19, 2008, Proceedings, Lecture Notes in Computer Science, vol 5299. Springer, pp 63–78, https://doi.org/10.1007/978-3-540-88403-3_5

Castro M, Liskov B (2002) Practical byzantine fault tolerance and proactive recovery. ACM Trans Comput Syst 20(4):398–461. https://doi.org/10.1145/571637.571640

Faradzhullaev R (2008) Analysis of web server log files and attack detection. Autom Control Comput Sci 42(1):50–54. https://doi.org/10.1007/s11950-008-1008-y

Gürsoy G, Bjornson R, Green ME, et al (2020) Using blockchain to log genome dataset access: efficient storage and query. BMC Medical

Genomics 13(Suppl 7)

He K, Shi J, Huang C, et al (2020) Blockchain based data integrity verification for cloud storage with t-merkle tree. In: Qiu M (ed) Algorithms and Architectures for Parallel Processing - 20th International Conference, ICA3PP 2020, New York City, NY, USA, October 2-4, 2020, Proceedings, Part III, Lecture Notes in Computer Science, vol 12454. Springer, pp 65–80, https://doi.org/10.1007/978-3-030-60248-2_5

He S, Liu G, He Z, et al (2010) Design and implementation of log management module in three-dimensional spatial database management system. In: The 18th International Conference on Geoinformatics: GIScience in Change, Geoinformatics 2010, Peking University, Beijing, China, June, 18-20, 2010. IEEE, pp 1–5, https://doi.org/10.1109/GEOINFORMATICS.2010.5567648

Jiuliang Liu KSXingming Sun (2020) A food traceability framework based on permissioned blockchain. Journal of Cyber Security 2(2):107–113. https://doi.org/10.32604/jcs.2020.011222

Kumar M, Singh AK, Kumar TVS (2018) Secure log storage using blockchain and cloud infrastructure. In: 9th International Conference on Computing, Communication and Networking Technologies, ICCCNT 2018, Bengaluru, India, July 10-12, 2018. IEEE, pp 1–4, https://doi.org/10.1109/ICCCNT.2018.8494085

Nakamoto S (2008) Bitcoin: A peer-to-peer electronic cash system. consulted

Nofer M, Gomber P, Hinz O, et al (2017) Blockchain. Bus Inf Syst Eng 59(3):183–187. https://doi.org/10.1007/s12599-017-0467-3

Pourmajidi W, Miranskyy AV (2018) Logchain: Blockchain-assisted log storage. In: 11th IEEE International Conference on Cloud Computing, CLOUD 2018, San Francisco, CA, USA, July 2-7, 2018. IEEE Computer Society, pp 978–982, https://doi.org/10.1109/CLOUD.2018.00150

Putz B, Menges F, Pernul G (2019) A secure and auditable logging infrastructure based on a permissioned blockchain. Comput Secur 87. https://doi.org/10.1016/j.cose.2019.101602

Shekhtman LM, Waisbard E (2021) Engravechain: A blockchain-based tamper-proof distributed log system. Future Internet 13(6):143. https://doi.org/10.3390/fi13060143

Son S, Gil M, Moon Y (2017) Anomaly detection for big log data using a hadoop ecosystem. In: 2017 IEEE International Conference on Big Data and Smart Computing, BigComp 2017, Jeju Island, South Korea, February 13-16, 2017. IEEE, pp 377–380, https://doi.org/10.1109/BIGCOMP.2017.7881697

Sur C (2019) Deepseq: learning browsing log data based personalized security vulnerabilities and counter intelligent measures. J Ambient Intell Humaniz Comput 10(9):3573–3602. https://doi.org/10.1007/s12652-018-1084-9

Uddin M, Memon MS, Memon I, et al (2021) Hyperledger fabric blockchain: Secure and efficient solution for electronic health records. Computers, Materials and Continua 68(2):2377–2397

Xu G, Cao Y, Xu S, et al (2022) A novel post-quantum blind signature for log system in blockchain. Computer Systems Science and Engineering 41(3):945–958. https://doi.org/10.32604/csse.2022.022100

Xu Y, Zhao S, Kong L, et al (2017) ECBC: A high performance educational certificate blockchain with efficient query. In: Hung DV, Kapur D (eds) Theoretical Aspects of Computing - ICTAC 2017 - 14th International Colloquium, Hanoi, Vietnam, October 23-27, 2017, Proceedings, Lecture Notes in Computer Science, vol 10580. Springer, pp 288–304, https://doi.org/10.1007/978-3-319-67729-3_17

Yue C, Xie Z, Zhang M, et al (2020) Analysis of indexing structures for immutable data. In: Maier D, Pottinger R, Doan A, et al (eds) Proceedings of the 2020 International Conference on Management of Data, SIGMOD Conference 2020, online conference [Portland, OR, USA], June 14-19, 2020. ACM, pp 925–935, https://doi.org/10.1145/3318464.3389773

Yue D, Li R, Zhang Y, et al (2018) Blockchain based data integrity verification in P2P cloud storage. In: 24th IEEE International Conference on Parallel and Distributed Systems, ICPADS 2018, Singapore, December 11-13, 2018. IEEE, pp 561–568, https://doi.org/10.1109/PADSW.2018.8644863