An Efficient Blockchain-Based Framework For File Sharing

Wanzong Peng
Harbin Institute of Technology
Tongliang Lu (azqsx098@qq.com)
PLA78156
Zhongpan Wang
Chongqing University

Article

Keywords:

Posted Date: April 24th, 2023

DOI: https://doi.org/10.21203/rs.3.rs-2815114/v1

Additional Declarations: No competing interests reported.

An Efficient Blockchain-Based Framework For File Sharing

Wanzong Peng[1], Tongliang Lu[2,*], and Zhongpan Wang[3]

[1]Harbin Institute of Technology, Harbin, 150001, China
[2]PLA78156, Chongqing, 400000, China
[3]Chongqing University, Chongqing, 400044, China
[*]azqsx098@qq.com

ABSTRACT

File sharing is the foundation of the Internet. But the traditional centralized service architecture will result in huge infrastructure costs and maintenance costs. Due to the lack of effective file management system, a lot of sensitive information is out of control and loss of confidentiality document has occur from time to time. In order to address the difficulty of tamper detection and the lack of supervision in the entire process of file transmission in the current Internet environment, this paper designs a block-chain-based system architecture for secure sharing of electronic documents. An efficient Blockchain model is used in our framework, and with the help of distributed storage system and asymmetric encryption technology, file sharing can be controlled, reliable and traceable in the transmission process.Referring to existing consensus mechanism, e.g., Delegated Proof of Stake (DPoS) and Practical Byzantine Fault Tolerance (PBFT), we propose a new consensus for efficient and secure file sharing.

Introduction

In recent years, the security of network files has been paid more and more attention. The traditional electronic file transfer process mostly uses centralized mode, data security depends entirely on the capabilities of large companies. The traditional file transfer system mainly uses C/S mode to provide services, and uploads all electronic files to the server for centralized management. This mode can cause problems, e.g., heavy load on the central server, high overhead of system resources, high cost of deployment and maintenance. As for P2P file sharing system, which is applied on the premise of trust between users, but is vulnerable to illegal access and malicious attacks, resulting in the disclosure of sensitive information. Therefore, it is essential to provide a high-performance and secure file transfer system.

With the popularity of cryptocurrency all over the world, blockchain technology has attracted tremendous interest from both academia and industry[1,2] and applied in a host of fields,including healthcare, Internet of Things (IoT), and cloud storage[3]. Blockchain technology provides us with new ideas on how to reliably transfer files. It has both decentralized characteristics and reliable security. With the idea of blockchain, we can achieve file transfer more efficiently, save a lot of server resources,reliably track the source and monitor the lifecycle of file.

Related Work

After the emergence of blockchain, people considered using it for cloud storage[4].The most application method was simple and centralized.Then Benet created IPFS[5].It is is a peer-to-peer distributed file system, and more and more people begin to use blockchain technology for file transfer by it[6–9].But they generally use existing blockchain systems such as Ethereum or Hyperledger Fabric for implementation, which is often not efficient enough.

In IoT, There has been increasing interest in high-performance information sharing blockchain. Dorri et al.[10] propose a lightweight blockchain architecture for IoT. Xu et al.[11] propose DIoTA, a decentralized ledger-based framework to authenticate IoT devices and data generated from them. And people are also starting to use the next generation blockchain for data sharing: Directed Acyclic Graph (DAG) Distributed Ledgers, e.g., IOTA[12], Nano[13].

Methods

Function

The main aim of our system is to share file safely. So we design two main functions to validate blockchain effects.

- **Upload**

  The user encrypts the local file with a randomly generated symmetric encryption key (as "the file key") and uploads it to IPFS to obtain the file hash (i.e. the IPFS content identifier, which is used to obtain the file). Of course, we can also use other storage platform(e.g., cloud storage platform). Each user need generate a "blockchain wallet", simplified as an asymmetric key here. The file key is encrypted with the user's public key and stored in the blockchain transaction together with the file hash and file related information. The process is depicted in Algorithm 1.

Algorithm 1: Upload file

input : $fileInfo$,$file$

```
// Fernet is a symmetric-key algorithm
```

$fileKey \leftarrow$ Fernet.generatekey()
$eFile \leftarrow$ Fernet.encrypt($file$,$fileKey$)
$eFileKey \leftarrow$ RSA.encrypt($fileKey$,$PublickKey_{user}$)
$fileHash \leftarrow$ IPFS.add($eFile$)
$transaction \leftarrow$ Transaction($fileInfo$,$eFileKey$,$fileHash$)
Sign($transaction$,$PrivateKey_{user}$)
$receiveNode \leftarrow$ GetNearbyNode()
Send($transaction$,$receiveNode$)

- **Transfer**

  File Hash is equivalent to a file on the blockchain. During download, the file owner retrieves the file hash and key from the blockchain, downloads the file from IPFS through the file hash, and decrypts the key with a private key to decrypt the file. During circulation, first decrypt the file key using the owner's private key, then encrypt the file key by the recipient's public key, and then store the encrypted file key and file hash in the transaction. The process is depicted in Algorithm 2.

Algorithm 2: Transfer file

input : $fileHash$,$eFileKey$,$PublickKey_{receiver}$

$fileKey \leftarrow$ RSA.decrypt($eFileKey$,$PrivateKey_{user}$)
if *download file* **then**
    $file \leftarrow$ IPFS.get($fileHash$)
    $file \leftarrow$ Fernet.decrypt($file$,$fileKey$)
else
    $newFileKey \leftarrow$ RSA.encrypt($fileKey$,$PublickKey_{receiver}$)
end
$transaction \leftarrow$ Transaction($PublickKey_{user}$,$PublickKey_{receiver}$,$fileHash$,$newFileKey$ or $eFileKey$)
Sign($transaction$,$PrivateKey_{user}$)
$receiveNode \leftarrow$ GetNearbyNode()
Send($transaction$,$receiveNode$)

Framework

For efficient transfer, we need a novel blockchain that can achieve high concurrency and security. Therefore, we borrowed from the general chain block storage structure and consensus algorithms (PoW, DPoS, etc.) widely used in cryptocurrency systems for sequential generation of single blocks and proposed our framework.

File Transaction Chain

This chain treats transactions as direct processing objects for file flow. When a transaction is validated, it is stored in the following structure: To facilitate traceability, the file is used as a root block to form an array; The user group is attached to the corresponding file, where each user points to the user who shares the file to him to form a chain structure; Transactions are sorted chronologically and attached to the corresponding users as an array. The overall structure is shown in Fig.1.Information stored in each section:

- File:file name and file hash.

- User:user's public key and the encrypted file key.
- Transaction:Address of both parties, transaction type, timestamp, transaction information, signature of validation node group, signature of transaction creator and transaction hash value.

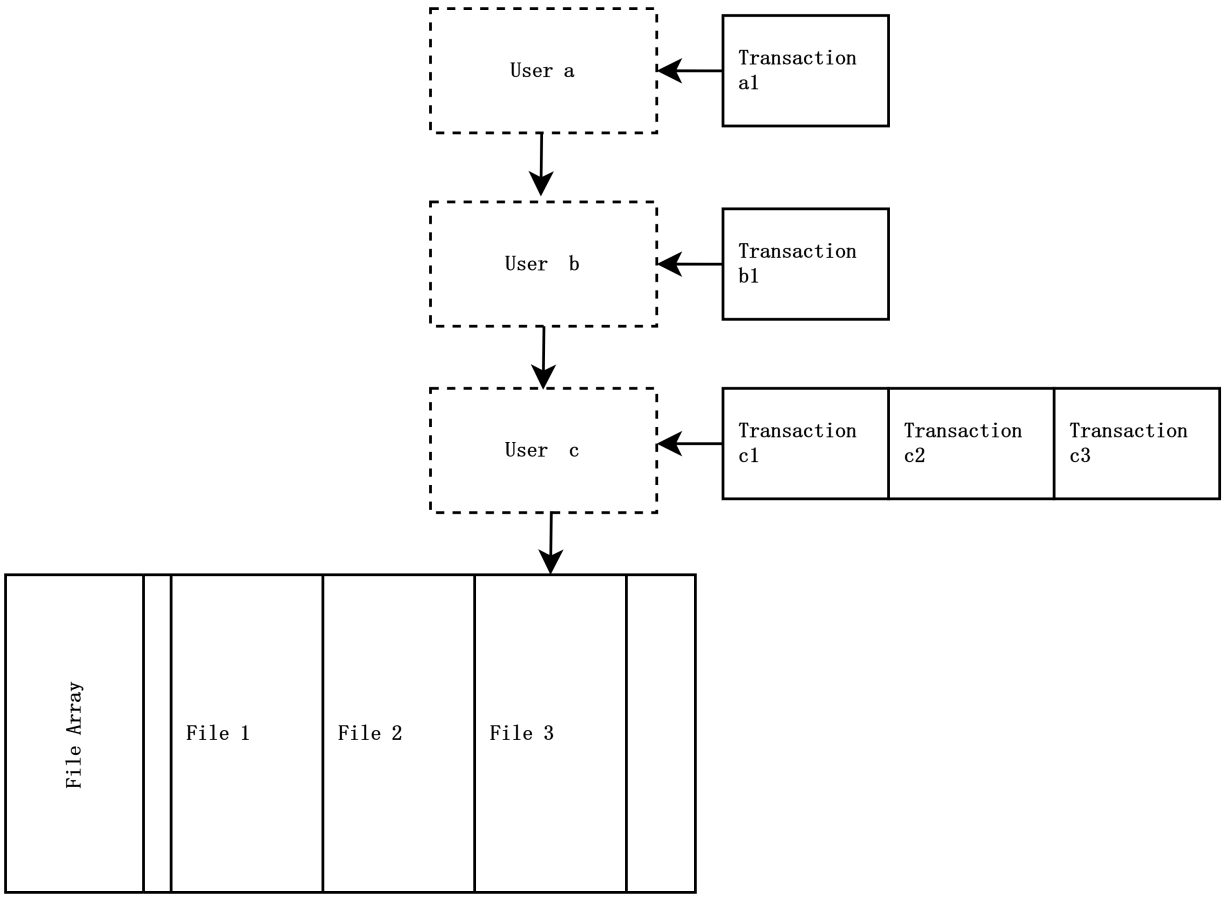

Figure 1. The data structure of File Transaction Chain.

File Info Chain

This is a traditional blockchain like the Ethereum blockchain, which is used to store system information such as voting, node reputation, and efficiency.

Normal-Case Operation

There are 5 main steps for whole system's lifecycle. The lifecycle is shown in Algorithm 3.

1) Vote to select the leadership group. Nodes vote based on efficiency and reputation of each node. The number of votes owned by the node is determined by the rating of the nodes on FIC(File Info Chain). Nodes broadcast the voting results as a transaction to all nodes, and after obtaining all voting results, calculate the node ranking. The top 1/5 nodes are selected as the leadership group.

2) Divide node groups by the leadership group. The members of the leadership group rotate as chair according to the ranking order. Based on the information blocks on FIC, each node is rated and divided into 6 groups with almost the same total score (the number of groups is adjusted reasonably according to the node size and transaction volume).

3) Create block on FIC. According to the PBFT algorithm[14], the chair packages and broadcasts the voting results, node grouping information, and other system information(e.g., efficiency information, reputation information).When 2/3 of the leadership group nodes confirm the result, blocks are created on FIC. If more than one-third of the members do not agree with the grouping, they need to go back to the second step until a grouping is formed.

4) Conduct transactions. Each group forms an independent P2P network, and adjacent groups establish peer-to-peer network channels to form a ring network structure. One user of the previous group of nodes request transactions through a random node in the next group.After receiving the transaction, the node (receiving node) in the transaction processing group broadcasts the transaction within the group. After more than 2/3 of the nodes in the group validate their signatures, the receiving node attaches all signature authentication to the transaction and broadcasts it to the group and the leadership group. The leadership group forms a P2P network channel with each group. After receiving it, the leadership group nodes broadcast the transactions to each group, and each node inserts the transactions into the transaction array of the initiating user in chronological order based on the corresponding files on File Transaction Chain. Any transaction that have failed validation in one-third of the validation nodes will be discarded and notified to the transaction initiator and leadership group.

5) Supervise transactions. The leadership group evaluates the efficiency of each node based on the speed of transactions and initiates transactions containing efficiency information; Randomly select transactions for validation, label transactions based on the results, and initiate transactions containing node reputation information. These transactions will be validated by the leadership group in the next time of grouping and the production blocks will be added to FIC. Loop steps 4-5, and repeat steps 2-3 every 10 minutes. After rotating as the chair at all leadership group nodes, start from step 1 again.

Evaluation

To validate the progressiveness of this framework, we need to analyze the time consumption. The total process time between two votes is T, which can be analyzed in two aspects: communication consumption and computational consumption.

Communication Consumption

Assuming that when a large number of nodes are evenly distributed in a network and there is no congestion caused by broadcasting, the average communication time RTT is considered as a fixed value.There are two types of Communication consumption.

1 Vote and divide. Each node needs to broadcast its own voting results to all nodes, and the chair will divide nodes and broadcast once, taking a total of 2 RTT.

2 Process and supervise transactions, create block on FIC. When each group conducts transaction processing, the transaction is initiated, and the receiving node receives it and broadcasts it to all nodes within the group for signature. Then, each node sends result to the receiving node for integration, and the receiving node broadcasts signed transaction to the group and leadership group, taking a total of 4 RTT. While other groups conduct the transaction, the leadership group conducts transaction supervision. In extreme cases, all leadership group nodes record, validate, and broadcast the results to all nodes, with a broadcast time of RTT. FIC block generation adopts the PBFT algorithm, which takes 5 RTT in 5 stages. The leadership group took a total of 6 RTT, which is longer than transaction processing.

Overall, in the lifecycle of a leadership group, each node mainly spends time processing transactions. In part 1 some single communications consume less time and have fewer occurrences, and communication consumption is mainly considered in part 2.

Computational Consumption

If we have a total of n nodes, γ leadership group nodes, α transactions and β groups, we can get algorithm complexity of the entire system.

1 Leadership group management. Dividing node requires traversing FIC and scoring each node, with a complexity of $\mathrm{O}(n)$. The time consumption of the transaction supervision part is linearly related to the number of transactions, and in the extreme case, γ leadership group nodes record and validate α transactions with a complexity of $\mathrm{O}\left(\frac{\alpha}{\gamma}\right)$. Each transaction can be completed by traversing the transaction chain once per node, and the time consumption can be ignored. The block generation adopts the PBFT algorithm with a complexity of $\mathrm{O}\left(\gamma^2\right)$. We set average coefficient as C_1, the leadership group calculates consumption is:

$$T_1 = C_1 \times \left(\frac{\alpha}{\gamma} + n + \gamma^2\right)$$

Algorithm 3: Nodes lifecycle

```
nodesInfo ← GetNodesInfoFromFIC()
votes ← Vote(nodesInfo)
infoTransaction ← Transaction(votes,PublickKey_node)
Sign(infoTransaction,PrivateKey_node)
Broadcast(infoTransaction)
Wait until get all nodes votes
Calculate ranking
if rank in the top quintile then
    // do as leader
    groupInfoTrans ← GroupNodes(nodesInfo)
    Broadcast(groupInfoTrans)
    transInfoTrans ← SuperviseTransaction()
    Broadcast(transInfoTrans)
    CreateFileInfoBlock()
else
    // do as normal node
    Get group info
    net ← CreateP2PNetwork()
    // process transactions
    transaction ← ListenTransaction()
    if transaction is from user then
        receiveNode ← GetNodeByGroupInfo()
        Send(transaction,receiveNode)
    else
        validate(transaction)
        Sign(transaction,PrivateKey_node)
        if no other signature in transaction then
            // broadcast in group
            net.broadcast(transaction)
            Wait until received 2/3 nodes signature
            net.send(transaction,leaderNode)
        else
            receiveNode ← GetFirstSigner(transaction)
            net.send(transaction,receiveNode)
        end
        // add transactions to FTC
        transaction ← ListenLeaderGroup()
        FTC.add(transaction)
    end
end
```

2 Transaction processing. β groups are conducted simultaneously, and the transaction is validated and signed by all nodes within the group after being broadcasted by the receiving node. Then, each node sends it back to the receiving node for integration, with a complexity of $O\left(\frac{\alpha}{\beta}\frac{n-\gamma}{\beta}\right)$. We set average coefficient as C_2 and the transaction processing time of each group is:

$$T_2 = C_2 \times \left(\frac{\alpha}{\beta}\frac{n-\gamma}{\beta}\right)$$

Generally, α is much greater than γ. By the above two equations, we can find that the more the number of nodes in the leadership group, the less transaction each node in the leadership group handle, the smaller the time consumption, and the greater the degree of decentralization. However γ Increasing will result the number of group nodes $\frac{n-\gamma}{\beta}$ in a too small number, which cannot guarantee the credibility of the transaction; Moreover, the leadership group nodes have lost their transaction ability, and users can only create transactions through other non leading nodes. Due to the limited processing capacity of non leading nodes, excessive number of leadership group nodes can lead to transaction congestion. So we need to select the appropriate number of groups and leader nodes based on the total number of nodes to ensure the overall system is reliable and efficient. Compare the main parts of T_1 and T_2, $\frac{\alpha}{\gamma}$ and $\frac{\alpha(n-\gamma)}{\beta^2}$.Beacuse $\gamma < n$ and β can be ignored compared to $n\gamma$,so $\beta^2 + \gamma^2 < n\gamma$, $\frac{\alpha}{\gamma} < \frac{\alpha(n-\gamma)}{\beta^2}$.

And C_1 is mainly caused by the program for transaction validation and recording, C_2 is mainly caused by the program for transaction validation and signature, and the consumption of signature algorithms is much greater than that of recording algorithms. So $C_2 > C_1$ and $T_2 > T_1$.

Experiment

Focus on analyzing T_2 through system simulation experiments. Our experiment environment is that each server runs multiple nodes within the local LAN, and the nodes communicate point-to-point through TCP to achieve "0 latency" communication. We use six computers, which is Intel core i5 at 3.6 GHz, 16GB of RAM running Microsoft Windows 11 64-bit version on 500GB hard drive. On each server, there are 30 nodes running through different ports, totaling 300 nodes. The leadership group takes 60 nodes. We request fixed number of transactions in each normal node. After leader group nodes validate all transactions, we collect and calculate the average time spent on each transaction. The results are shown in Figure 2.

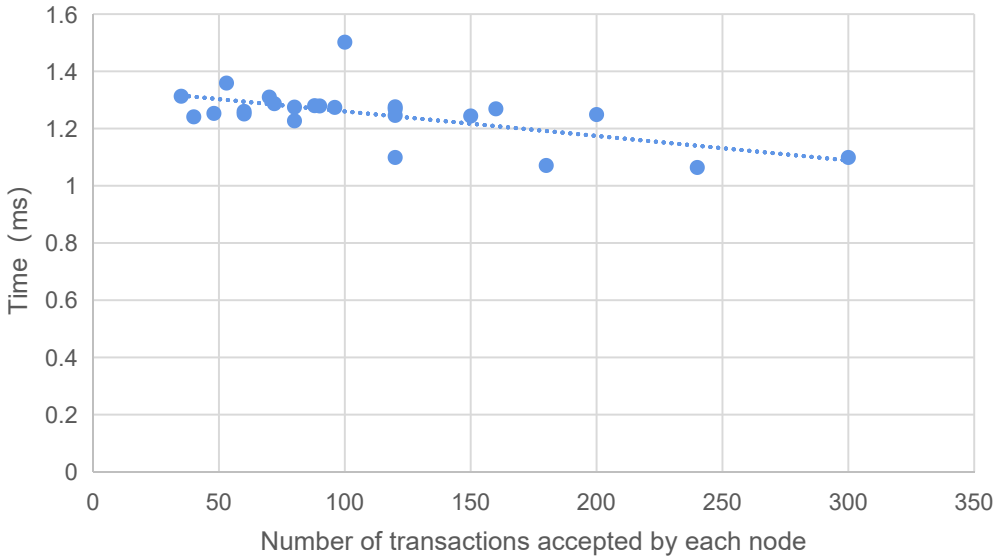

Figure 2. Verification and signature time consumption for each transaction.

The number of nodes within a group has a significant impact on transaction efficiency. The average number of transactions initiated by each node per second is TPS. We set different TPS for evaluate processing capacity of groups of different sizes, we collect the time of each group finish requests of one second. Blockage occurs when the total time consumed exceeds 1 s. The result is shown in Table 1.

TPS	120 nodes	80 nodes	60 nodes	48 nodes	40 nodes	35 nodes
10	1.394	1.046	0.800	0.637	0.526	0.489
15	2.055	1.592	1.217	0.986	0.794	0.758
20	2.721	2.161	1.628	1.301	1.087	0.974
25	3.526	2.641	1.975	1.619	1.587	1.184

Table 1. Transaction computational time consumption(s).

To compare with the performance of other directed acyclic graph blockchains(e.g., IOTA), we refer to the experimental settings in other test[15], the number of groups was increased to 30, and there are 8 nodes every group. We test average processing speed of each group from 5 to 150 TPS per second in 300 seconds.The result is shown in Figure 3.

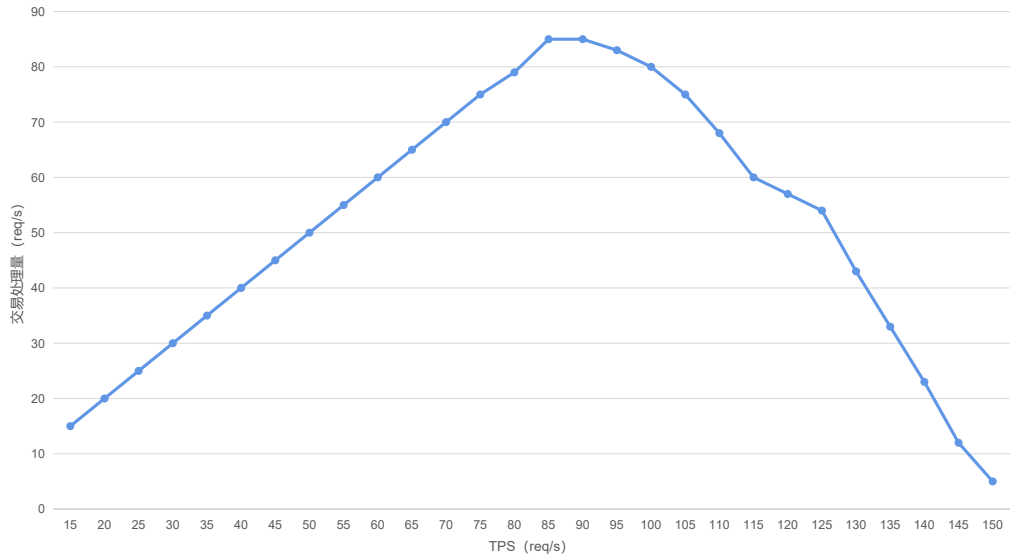

Figure 3. Verification and signature time consumption for each transaction.

Among the three implementations of IOTA, NANO, and Byteball in the paper[15], NANO can achieve a maximum throughput of 60 transactions per second. Our system can achieve 85 transactions per second for each group, and our global throughput needs to be multiplied by the number of groups. Compared to the state-of-the-art method, our method has made some progress.

Results

In this paper, to address the fundamental issue of file transfer, we proposes a new blockchain-based framework for file transfer. Firstly, we have proposed the core functions of the entire system based on security requirements. Then, in order to efficiently complete the task, we designed a dual-chain blockchain structure and designed the entire system lifecycle based on the PBFT algorithm and sharding concept[16]. Finally, we verify the feasibility and efficiency of the framework through analysis and quantitative experiments of the system. The difference between the proposed framework and existing solutions lies in two aspects. Firstly, we adopt a relatively centralized consensus approach through the leadership group to ensure efficient operation

of the system while ensuring security. The second is that transaction processing is highly parallel in each group, which solves the problem of low efficiency in existing blockchain file transfer solutions.

Data availability

The datasets generated during and analyzed during the current study are available from the corresponding author on reasonable request.

References

1. Wylde, V. *et al.* Cybersecurity, data privacy and blockchain: A review. *SN Comput. Sci.* **3**, 127 (2022).
2. Taylor, P. J., Dargahi, T., Dehghantanha, A., Parizi, R. M. & Choo, K.-K. R. A systematic literature review of blockchain cyber security. *Digit. Commun. Networks* **6**, 147–156, DOI: https://doi.org/10.1016/j.dcan.2019.01.005 (2020).
3. Salman, T., Zolanvari, M., Erbad, A., Jain, R. & Samaka, M. Security services using blockchains: A state of the art survey. *IEEE Commun. Surv. & Tutorials* **21**, 858–880, DOI: 10.1109/COMST.2018.2863956 (2019).
4. Sharma, P., Jindal, R. & Borah, M. D. Blockchain technology for cloud storage: A systematic literature review. *ACM Comput. Surv.* **53**, DOI: 10.1145/3403954 (2020).
5. Benet, J. IPFS - Content Addressed, Versioned, P2P File System. *arXiv e-prints* arXiv:1407.3561, DOI: 10.48550/arXiv.1407.3561 (2014). 1407.3561.
6. Chen, Y., Li, H., Li, K. & Zhang, J. An improved p2p file system scheme based on ipfs and blockchain. In *2017 IEEE International Conference on Big Data (Big Data)*, 2652–2657, DOI: 10.1109/BigData.2017.8258226 (2017).
7. Vimal, S. & Srivatsa, S. K. A new cluster p2p file sharing system based on ipfs and blockchain technology. *J. Ambient Intell. Humaniz. Comput.* DOI: 10.1007/s12652-019-01453-5 (2019).
8. Nyaletey, E., Parizi, R. M., Zhang, Q. & Choo, K.-K. R. Blockipfs - blockchain-enabled interplanetary file system for forensic and trusted data traceability. In *2019 IEEE International Conference on Blockchain (Blockchain)*, 18–25, DOI: 10.1109/Blockchain.2019.00012 (2019).
9. Liu, M., Palaoag, T. & Zhang, W. An e-resource sharing solution based on blockchain technology. In *Proceedings of the 2021 4th International Conference on Blockchain Technology and Applications*, ICBTA '21, 101–106, DOI: 10.1145/3510487.3510502 (Association for Computing Machinery, New York, NY, USA, 2022).
10. Dorri, A., Kanhere, S. S. & Jurdak, R. Towards an optimized blockchain for iot. In *2017 IEEE/ACM Second International Conference on Internet-of-Things Design and Implementation (IoTDI)*, 173–178 (2017).
11. Xu, L. *et al.* Diota: Decentralized-ledger-based framework for data authenticity protection in iot systems. *IEEE Netw.* **34**, 38–46, DOI: 10.1109/MNET.001.1900136 (2020).
12. Müller, S. *et al.* Tangle 2.0 leaderless nakamoto consensus on the heaviest dag. *IEEE Access* **10**, 105807–105842, DOI: 10.1109/ACCESS.2022.3211422 (2022).
13. Le Mahieu, C. Nano: A feeless distributed cryptocurrency network. https://content.nano.org/whitepaper/Nano_Whitepaper_en.pdf.
14. Castro, M. & Liskov, B. Practical byzantine fault tolerance. In *Proceedings of the Third Symposium on Operating Systems Design and Implementation*, OSDI '99, 173–186 (USENIX Association, USA, 1999).
15. Dong, Z., Zheng, E., Choon, Y. & Zomaya, A. Y. Dagbench: A performance evaluation framework for dag distributed ledgers. In *2019 IEEE 12th International Conference on Cloud Computing (CLOUD)*, 264–271, DOI: 10.1109/CLOUD.2019.00053 (2019).
16. Liu, Y. *et al.* Building blocks of sharding blockchain systems: Concepts, approaches, and open problems. *Comput. Sci. Rev.* **46**, DOI: 10.1016/j.cosrev.2022.100513 (2022).

Author contributions statement

Wanzong Peng conceived the framework and experiment, Tongliang Lu conducted the experiment, Zhongpan Wang analysed the results. All authors reviewed the manuscript.

Additional information

Competing interests

The authors declare no competing interests.