SMDSB: Efficient Off-Chain Storage Model for Data Sharing in Blockchain Environment

Randhir Kumar, Ningrinla Marchang, and Rakesh Tripathi

Abstract Blockchain technology has been gaining great attention in recent years. Owing to its feature of immutability, the data volume of the blockchain network is continuously growing in size. As of now, the size of Bitcoin distributed ledger has reached about 200 GB. Consequently, the increasing size of the blockchain network ledger prevents many peers from joining the network. Hence, the growing size not only limits the expansion of network but also the development of the blockchain ledger. This calls for development of efficient mechanisms for storage and bandwidth synchronization. As a step toward this goal, we propose an IPFS-based decentralized off-chain storage called storage model for data sharing in blockchain (SMDSB), to store the data volume of the blockchain network. In the proposed model, the miners validate and deposit the transactions into an IPFS-based decentralized storage, whereas they store the hash of the transactions in a blockchain network. Thus, by utilizing the characteristics of IPFS-based off-chain storage and its feature of hash creation, the blockchain ledger size can be significantly reduced. Implementation of SMDSB results in reduction of Bitcoin storage by 81.54%, Ethereum by 53.86%, and Hyperledger by 62.59%. Additionally, the experimental results also highlight the impact on storage cost for 1 transaction per second (TPS) in a year.

Keywords Blockchain · Off-chain storage · Interplanetary file system (IPFS) · IPFS hash · Peer-to-peer data sharing model

R. Kumar (✉) · R. Tripathi
Department of Information Technology, National Institute of Technology Raipur, Raipur, Chhattisgarh 492010, India
e-mail: rkumar.phd2018.it@nitrr.ac.in

R. Tripathi
e-mail: rtripathi.it@nitrr.ac.in

R. Kumar · N. Marchang · R. Tripathi
Department of Computer Science, North Eastern Regional Institute of Science and Technology (NERIST), Nirjuli, Arunachal Pradesh 791109, India
e-mail: ningrinla@gmail.com

D. Swain et al. (eds.), *Machine Learning and Information Processing*,
Advances in Intelligent Systems and Computing 1311,
https://doi.org/10.1007/978-981-33-4859-2_24

1 Introduction

The blockchain technology is known to be decentralized, immutable, and cryptographically secure with high byzantine fault tolerance. It was first implemented in the Bitcoin cryptocurrency [1]. Ever since the blockchain technology operation started in the year 2009, the data volume of the Bitcoin ledger has been growing in size every year leading to storage problems [3]. As of now, the ledger size of the Bitcoin network has grown up to 200 GB, and this size is said to be increasing daily by 0.1 GB. Hence, data storage [4, 7, 19] and data synchronization [5] have emerged as challenging issues. With regard to Bitcoin, the transaction blocks only consist of the transfer records which contain the addresses of sender and receiver and consequently are relatively very small data records. However, in future, application of the blockchain technology will not only be limited to Bitcoin. For example, the IoT-based blockchain system [6, 18, 22] may consist of off-chain data like audio, video, and text files. Therefore, in the foreseeable future, a high demand for off-chain data in blockchain network is perceived. Consequently, an efficient off-chain data storage model is required to deal with large volumes of data.

In the decentralized structure of blockchain technology, every node keeps a copy of the ledger. The copying of a ledger among all the peers in the network ensures security and availability of data. However, repeated storage of the ledger by every node increases data redundancy in the network. Therefore, a suitable storage model is required to reduce data redundancy in the network.

Off-chain data are non-transactional data that are too large to be stored in the blockchain efficiently, or, that require the ability to be changed into a different format in order to reduce the size of the original data. The objective of off-chain data storage is to tackle the limitation of growing size of blockchain networks by storing the off-chain data elsewhere off the blockchain. However, in doing so, its fundamental properties like availability, privacy, integrity, and immutability network may be compromised. Various models have been already proposed [3, 7, 8, 10] for off-chain storage and reduction of data size. The downside is that use of these approaches may result in the original state of data being potentially violated. Thus, the need of the hour is an efficient off-chain data storage model which can maintain the state of data and their integrity.

With the help of distributed storage systems like interplanetary file system (IPFS), large volumes of data can be stored off-chain and the hash of the off-chain data within the blockchain network, thereby maintaining immutability. The files in IPFS are stored in the distributed hash table (DHT) with their hash mapping which ensures data integrity. The stored files in IPFS are content-addressed and temper-proof due to the version control feature [2]. IPFS removes redundant off-chain data and maintains version control for non-transactional data. The edit history is easily recorded and traced out in the IPFS distributed peer-to-peers storage system. Off-chain data are retrieved from IPFS with the help of their hash values. Moreover, IPFS provides availability of data and maintains synchronization among the peers in the network [24].

The rest of the paper is organized as follows: Sect. 2 provides an overview of the related work. In Sect. 3, we present the proposed model for off-chain data storage in order to reduce the size of blockchain network. In Sect. 4, we present the methodology used in SMDSB model. Section 5 shows the implemention of off-chain data storage. Section 6 gives the result analysis, and Sect. 7 concludes the paper.

2 Related Work

This section discusses related existing work. In [3], a coding-based distributed framework is introduced to reduce the ledger size, where each block is divided into sub-blocks. A coding scheme is then used to encode these sub-blocks into more sub-blocks, which are then disseminated to all the peers in network. The encoding and decoding process increases the complexity of the blockchain network. Moreover, a large number of blocks get distributed over the network. Although the coding (and decoding) scheme helps in reducing the size of data to some extent, it leads to data inconsistency problems and decreases the system efficiency.

In [7], a technique that records the transactions by using block change approach is proposed. In this approach, a set of transactions which is older moves into summary block, and the current transactions are stored in the network chain. The summary block approach reduces the size of the main chain of the network up to some extent. However, at the same time, size of the summary block increases owing to the collection of older transactions of different blocks. Moreover, storage space for the summary block is again needed. In another similar work [10], a summary block approach is used to store the details of blockchain ledger is presented. It utilizes the file system to store the state of the summary block. This approach suffers from the issue of historical traceability. Besides, the entire system depends on the traditional file systems because of which if the file gets corrupted, then the complete summary block state may vanish.

The account tree approach is employed to store the account details of non-balance accounts in [8]. In the proposed work, the expired transaction blocks are deleted, and only the block header details such as merkle root, proof-of-work, and previous hash of the block are kept [5]. This approach of holding current transactions helps to reduce the size of the blockchain ledger to the maximum extent. On the downside, historical records of transactions cannot be traced out.

The above techniques have various limitations. Keeping this in mind, in this paper, we present an off-chain storage model for the blockchain network with an aim to overcome the problem of storage. The proposed IPFS-based off-chain storage scheme maintains the IPFS hash of the transactions which is unique for each transaction. The hash of a transaction consists of only 46 bits irrespective of the original size of the transaction. Furthermore, the IPFS model maintains the versioning control system. In our proposed storage model, the same IPFS hash gets disseminated among all the peers of network. This ensures consistency in the network. Additionally, our technique eliminates the drawback of third party dependency for storage. Moreover,

synchronization among the peers is maintained due to the reduced size of transactions. Consequently, the problem of adding new peers is possibly resolved. Furthermore, our proposed model supports all types of transaction data, maintains the traceability of the blockchain history, and ensures the availability of transactions among the peers.

Motivation
The work in this paper is motivated by limitations in existing works where size of distributed ledger is reduced by keeping the old transactions record in the chain of summary block. These approaches which use summary block storage suffer when summary block data volume increases along with time. Another limitation of the approach is that it fails to maintain the history of transactions. In this paper, we propose a storage model which can store structured as well as unstructured files by using off-chain storage.

Novelty
Whereas existing literature mostly addressed the problem on how to reduce the size of blockchain storage, to the best of our knowledge, there is no work as yet on storage and access of unstructured files such as CSV, Excel, PPT and Word. Hence, to the best of our knowledge, we are the first to provide a storage model for unstructured files by using blockchain and IPFS distributed file storage system. The proposed model will be very much useful in real-time applications including supply chain management, health care, finance, identity management, insurance claiming, and other applications in which government agencies need to store reports, audio, video, and other transactions in a blockchain network.

The contribution of the paper is as follows:

1. We have proposed a storage model for data sharing in blockchain (SMDSB) to reduce the size of blockchain network using IPFS distributed file storage system.
2. We have implemented proof-of-work consensus approach to validate the transactions with their IPFS hash values. The mining process is applied for block creation in the blockchain network.
3. The ledger which is relatively small in size is disseminated among all the peers of blockchain network which ensures synchronization among the peers.
4. We have given transaction size representation in bytes of Bitcoin, Ethereum and Hyperledger using SMDSB and found that the model is efficient in reducing storage size and in providing synchronization and availability of transactions among the peers in the network.

3 Storage Model for Data Sharing in Blockchain (SMDSB)

This section discusses SMDSB, the proposed storage model for data sharing in blockchain. The model can efficiently store both structured and unstructured files such as pdf, text, audio, video, csv, and excel sheets. Figure 1 illustrates how the model works. It consists of four different components, viz. user, miner, off-chain storage and blockchain storage.

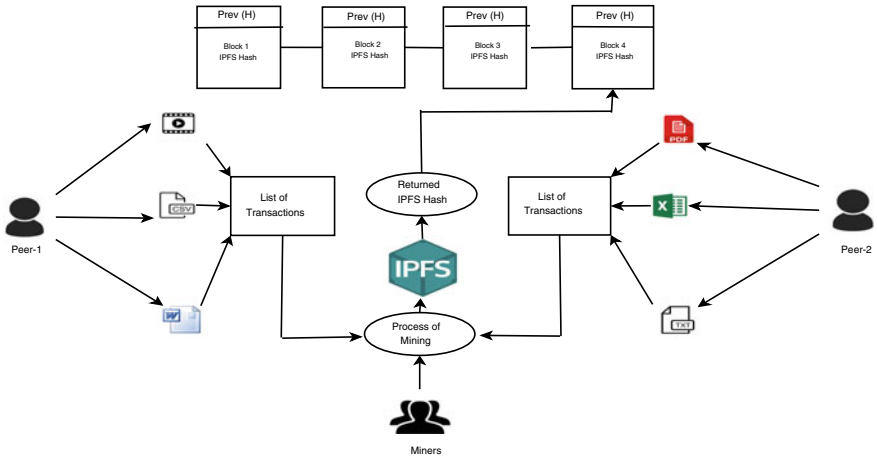

Fig. 1 SMDSB model for data sharing using blockchain and IPFS-based decentralized storage

User

A peer in the blockchain network is also called a user. Different peers can share their information on the network by using the upload process. The users can share more than one file at a time in the network. The other peers (users) in the network can verify the transactions by using a consensus technique. The user must be registered to become a part of the blockchain network.

Miner

A miner validates the list of transactions and disseminates it among the peers in the network to verify the transactions in order to construct a new block in the blockchain network. In SMDSB, the miner stores the original transaction in the IPFS distributed storage during the mining process. At the same time, it generates the IPFS hash of the transaction and stores it in the blockchain network. The IPFS hash thus generated is unique for each transaction and gets modified only when the original content (corresponding transaction) is modified due to the two inherent IPFS features, viz. version control system and distributed hash table (DHT) storage.

Off-chain storage

IPFS is used for off-chain storage in SMDSB. IPFS is distributed in nature. Moreover, it provides facilities to generate hash of a file which is much smaller in size than that of the file. The IPFS hash consists of only 46 bytes, thereby needing little storage in the blockchain network. SMDSB supports storage of both structured and unstructured files. Examples of structured files include pdf, audio, video, text, and image files, whereas those of unstructured files include excel, csv, and word files. Therefore, the SMDSB model can be used in real-time applications to share large volumes of information among peers.

Blockchain storage
As mentioned earlier, only the IPFS hash of a transaction is stored in the blockchain network in order to reduce the size of blockchain, while the transaction itself is stored in IPFS. Moreover, to maintain the history of transactions, we keep the timestamp of each transaction, and similar copies of the chain get disseminated among the peers to provide the transaction details. Furthermore, the proof-of-work consensus is applied to ensure consistency in the network. To identify the files in the blockchain network, we also include the file extension of a file.

The following steps are involved in the working of the SMDSB model:

1. The user can upload both types of files, i.e., structured and unstructured as a transaction in blockchain network. Moreover, a list of files can also be uploaded at a time by a peer in blockchain network as a transaction.
2. The list of transactions gets disseminated to the local miners for validation. Once the transactions get validated by the miners, then they are verified by the peers in network with the purpose of constructing a new block in the blockchain network.
3. The transaction hash is generated during the mining process, and the IPFS hash gets stored into the blockchain network.
4. The synchronization of peers is easily maintained owing to the IPFS hash which is unique for each transaction, and the same IPFS hash gets accessed by other peers in the network.

4 Methodology

In this section, we present the methodology used in the SMDSB model. We first present the existing storage architecture and then the proposed storage architecture.

In the existing blockchain architecture, which we call *On-chain Storage*, files and their hashes are stored on the blockchain network as transactions. The problem with this method is that as more and more files get uploaded, the storage size of blockchain increases rapidly since all the content is stored on the blockchain network. Whenever a new peer arrives, it has to download the entire chain containing all the files. This solution is definitely not scalable, especially since file sizes may run into many Megabytes.

The proposed storage architecture which overcomes the above drawbacks uses the IPFS. IPFS is a distributed P2P hypermedia distribution protocol. IPFS [13] uses content-addressable approach and generates cryptographic hashes to give unique fingerprint to files, removes redundancy across the network. Moreover, these unique hashes are used for information retrieval. In the proposed model, the file hashes are stored in the blockchain network and the actual files in IPFS. Thus, we make large savings in size of the blockchain network while maintaining privacy of the transactions (files). The content-addressable hashes are linked into the blockchain network to reduce the overhead of blockchain scalability. Each hash is mapped using map_user function in our implementation in order to provide auditability of transactions.

The proposed approach also uses proof-of-identity (PoI), which provides peer security by ensuring that one peer has only one account within a system, without sacrificing the advantages of anonymity or pseudonymity. Each peer gets its identity after the registration process. Proof-of-work mining is also incorporated in the model. We use consortium blockchain structure and local mining strategy to validate the transactions. The local mining approach improves the scalability of chain and can be used to do away with centralized mining [25].

4.1 Transaction and Peer Security

This subsection discusses some algorithms used in SMDSB for data storage and peer security.

```
Input: PoI
Output: verification of document upload
// Authorization of peers gets validated while upload of document //
if (document.sender.PoI is not authorized) then
    return false
else
    //upload the document to the IPFS network and related metadata to the
    blockchain network.//
    return true
end if
```

Algorithm 1: Algorithm for authentication of the peer

```
Input: PoI, file
Output: integrity of off-chain and on-chain storage
// Computing hash of files by adding it to IPFS //
Hash = file | ipfs add
// Add the IPFS hash to blockchain network//
Block_index = add_metadata(Hash)
//Mapping the PoI and computed hash for off-chain and on-chain Integrity
of uploaded file //
map_user((PoI, Block_index, Hash, timestamp, PoW, Hash_of_previous_
Block))
```

Algorithm 2: Algorithm for Integrity of off-chain and on-chain storage

```
Input: Registraion-Id, BlockId
Output: true, false
if (Registration-Id, Block_Id)==valid then
    S = get_hash_of_Block_Id (Block_Id)
    T = get_value_from_IPFS(IPFS_Hash)
    if (S != T) then
        return false
    else
        return true;
    end if
else
    return false
end if
```

Algorithm 3: Algorithm for auditability verification during transaction access

Algorithm 1 is suggested during the exchange and upload of documents to verify the peers in the blockchain network. The algorithm limits access of the shared docu-

ments to malicious peers. The proof-of-identity guarantees model security. If the peers are registered in the network, unique proof-of-identity (http:/<url:port>/nodes/register) is given to them. This proof-of-identity gets verified against the peers authorization.

Data Storage in blockchain network: When a peer begins uploading a file, the file is added to the IPFS, and IPFS Hash (content-addressed hash) is added to the blockchain network. However, the mapping is done to define the peer (PoI) and its corresponding Block_Id as shown in Algorithm 2.

Auditability of Data in Blockchain network: As shown in Algorithm 3, the auditability of the transactions is checked. The auditability process is done, when a peer accesses the content-addressed hash of a transaction from the blockchain network.

5 Implementation

The implementation of SMDSB was carried out in the IPFS distributed file sharing system where each transaction is represented in the blockchain network by its IPFS hash value instead of the transaction. The experimental setup consists of Python Anaconda and Python Flask. Experiments were performed on Intel(R) Xeon(R) W-2175 CPU @ 2.50 GHZ running Window x64-based processor with 128 GB of RAM and 1 TB of local storage. We implemented four different modules, viz. user, miner, off-chain storage, and blockchain storage, which have been discussed earlier.

As shown in Fig. 2, the peers can upload the structured and unstructured files as a transaction into the blockchain network. We have tested with a maximum size of 100 MB video upload as a transaction. The uploaded original file gets stored into

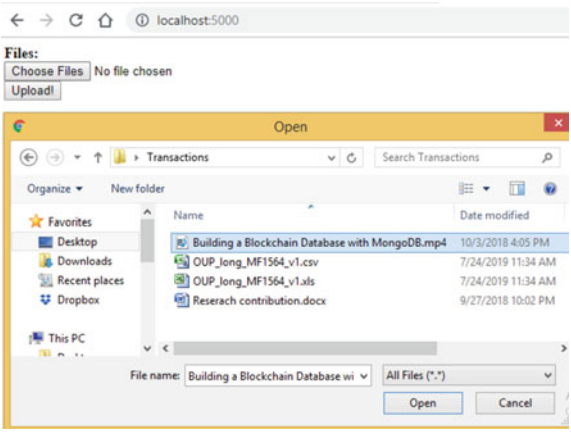

Fig. 2 Upload of files on blockchain network as a transaction

Fig. 3 Process of mining for transaction (Mp4 file) validation and block creation

IPFS distributed peer-to-peer off-chain storage initially. Furthermore, the generated IPFS hash also gets stored in the blockchain network.

Figure 3 illustrates the mining process for two different files (mp4 and csv files, respectively, as a transactions) in order to validate the transactions and creation of a new block into the blockchain network. We generated the IPFS hash during the mining process and disseminated the same hash among the peers of blockchain network for verification. We also stored the file names and file extensions during the mining process so as to be able to identify the structured and unstructured files in the blockchain network.

To access the files from the blockchain network, the peers have to use a content-addressed scheme:

1. To access a structured file, the following url must be followed http://127.0.0.1:8080/ipfs/<ipfs hash value>
2. To access an unstructured file, the following URL must be followed http://127.0.0.1:8080/ipfs/<ipfs hash value> > filename with extension

As shown in Fig. 4, we have used a content-addressed scheme (i.e., the original content gets accessed by their hash value) to access the video file with the format provided above. Similarly, the unstructured files can be accessed by using the above format. The content-addressed scheme is very efficient for transaction access and synchronization among peers (owing to the unique IPFS hash value). In the content-addressed scheme, more than one peer can access the same transaction at the same time with the help of the distributed hash table (DHT) [2]. The IPFS hash gets modified once the transaction is updated by the peers. The modified IPFS hash is stored into the DTH. The versoning control system of IPFS framework ensures availability of the modified transaction using different timestamps. Figure 5 shows a list of transactions stored with their unique IPFS hash value, file extension, and timestamp. The IPFS hash value uniquely identifies a transaction in the blockchain network. The use of file extension is to identify the structured and unstructured files in

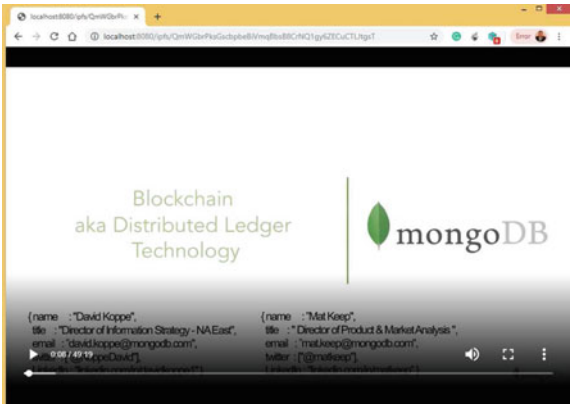

Fig. 4 Access of structured files by using content-addressed scheme

Fig. 5 List of transactions with their IPFS hash on blockchain network (port 5000 main chain)

http://localhost:5000/chain


```
{
"chain" : [
{
"filext" : "1",
"index" : 1,
"ipfsh": "1",
"Previous_Hash": "1",
"Proof" : 100,
"timestamp" : 1564165740.0441937
},
{
"filext" : "Building_a_Blockchain_Database_with_MongoDB.mp4",
"index" : 2,
"ipfsh": "{
"Hash" : "QmWGnrPksGscbpbeBiVmqBbsB8CrNQ1gy6ZECuCTLJtgsT",
"Name" : "file",
"Size" : 67407542
}",
"Previous_Hash": "b3cd8fb8d172e3f357ca347220d4b7bf679649d4a4ef05a801b",
"Proof" : 67813,
"timestamp" : 17141645670.5687912
}
"length" : 2
]
```


the network, and finally, the timestamp is recorded in order to maintain the history of transactions. Furthermore, we have used proof-of-work (PoW) consensus approach to maintain consistency of blocks in the network, and previous hash is used for the immutability in blockchain network. In our working model, we have added almost all types of unstructured files as transactions such as csv, excel, docx, and ppt files. The main objective is to store the unstructured files in a blockchain network to maintain the records of reports, presentations, and important documents. Moreover, the chain provides synchronization during addition of new peers into the network due to the reduced size of transactions (IPFS hashes of transactions).

Figure 6 shows how we have added a peer into SMDSB model by using json structure. The peer gets registered into the main chain of blockchain network. The main chain is currently running on port 5000, and we have added the peer into main chain to access the transactions of blockchain network. To add the peers, we have used postman browser. In SMDSB, an anonymous peer cannot access the chain until that peer is registered into blockchain network.

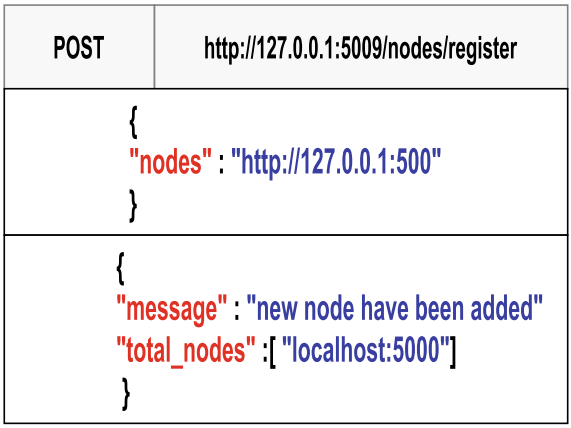

Fig. 6 Process of adding peer in SMDSB model

Fig. 7 List of transactions accessed by a peer (port 5009 peer chain) in blockchain network

As shown in Fig. 7, the peer with port 5009 is allowed to access the main chain of the blockchain network, which was initially created on port 5000, after its registration. The peers are allowed to access the main chain only after acceptance of blockchain network consensus (PoW). In SMDSB, we have employed proof-of-work (PoW) consensus to ensure the confirmation of transactions and creation of a new block in the blockchain network. As it can be seen in Fig. 7, the same chain is copied by the peers http://127.0.0.1:5009/nodes/resolve in the network.

6 Transaction Size Analysis in Terms of Storage Cost

In this section, we report the results based on implementation of the SMDSB model. The metrics used to compare the blockchain models are given by the formulas in Eqs. (1) and (2) [26]. The meaning of each symbol in Eq. (1) is as follows: Hash

(size of all block headers in blockchain), kHash (IPFS hash size), Tn (number of all transactions in blockchain), and Tx (original data size of each transaction). To compare the storage space requirement, we have studied different storage models like Bitcoin, Ethereum, and Hyperledger.

$$\text{Compression Ratio} = \frac{\text{Hash} + (\text{kHash} \times \text{Tn})\text{Hash} + \sum_{k=1}^{\text{Tn}} T x_k}{} \tag{1}$$

$$\text{Storage Space Saving Rate} = 1 - \text{Compression Ratio} \tag{2}$$

The Bitcoin network contains a header size of 80 bytes, 500 on average transactions in one block, and each transaction is represented by a 250-byte hash value [1, 11, 12, 17]. Similarly, the Ethereum network contains a 508-byte header size, 1196 average transaction per block, and 100 bytes representation for each transaction [9, 13]. The Hyperledger fabric contains 72 bytes block header, 3500 transaction per block, and 123 bytes of representation for each transaction [14–16, 20, 21]. We have compared the SMDSB with the existing blockchain storage models (Bitcoin, Ethereum, Hyperledger). Figure 9 shows difference in hash size of a transaction in different blockchain storage models. In SMDSB, we have represented each transaction with 46 bytes that thereby reducing the transaction storage size by 80.60% as compared to that of Bitcoin (Fig. 8).

In this experiment, we studied the cost of different storage models. The cost per transaction in Hyperledger is USD 3.86 for 5 KB storage. Similarly, the cost per transaction for Bitcoin is USD 1.30 for 5 KB of storage. However, this might change based on the underlying Bitcoin cryptocurrency. The cost per transaction in Ethereum is USD 0.25 for 5 KB of storage, and again, it depends on the underlying Ether cryptocurrency [23]. We have computed the storage and their costing per transaction based on the IBM off-chain storage. The company works for 8 h in a day, 240 days

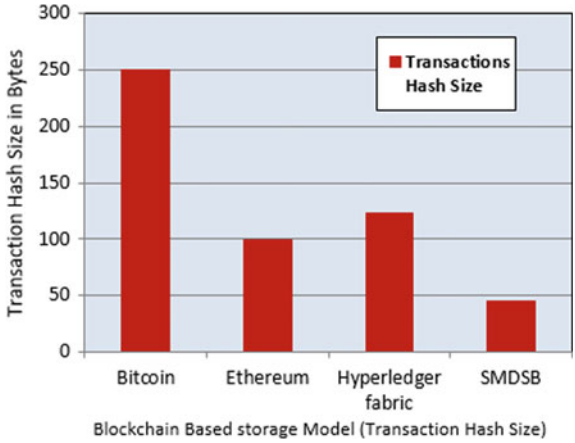

Fig. 8 Comparison of transaction size in bytes

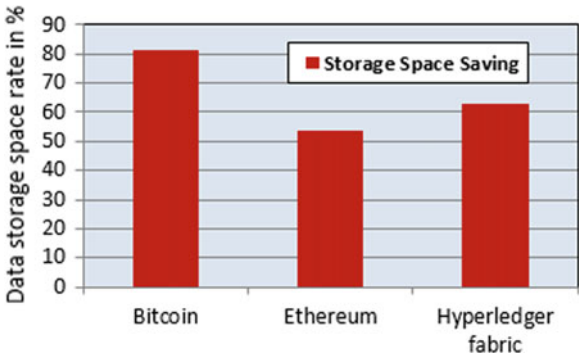

Fig. 9 Storage size reduction by using SMDSB

in a year (standard time taken from IBM white paper off-chain storage) in order to manage the storage in Hyperledger for each transaction.

We evaluated the existing storage models shown in Fig. 8 to compute the transactions size and their per year cost. The representation of a Bitcoin transaction takes 250 bytes = (250/1024)= 0.245 KB for each transaction (1 transaction per second). Similarly, Ethereum, Hyperleger, and IFPS take 0.098 KB, 0.121 KB, and 0.0449 KB for representation of each transaction, respectively. We have computed the storage size required in a year for 1 transaction per second (TPS) and their cost (shown in Table 1).

Table 1 Storage size and cost evaluation of different blockchain model

Storage size and cost evaluation	
Bitcoin	Storage size (in Bitcoin)= 1 TPS * 0.245 * 3600 s/h * 8 /per day * 240 per year = 296,900 KB = 289.945 GB= 0.2832 TB/year/transaction Storage cost in bitcoin for 1 TPS (in one year) = 296,900 * USD 0.26 = USD 77,194
Ethereum	Storage size (in Ethereum) = 1 TPS * 0.098 * 3600 s/h * 8 /per day * 240 per year = 661.5 KB = 0.646 GB= 0.000631 TB/ year/transaction Storage cost in ethereum for 1 TPS (in one year) = 661.5 * USD 0.05 = USD 33.07
Hyperledger	Storage size (in Hyperledger) = 1 TPS * 0.121 * 3600 s/h * 8 /per day * 240 per year = 810.791 KB = 0.792 GB= 0.000773 TB/year/transaction Storage cost in Hyperledger for 1 TPS (in one year)} = 810.791 * USD 0.77 = USD 624.309
SMDSB	Storage size (in SMDSB)} = 1 TPS * 0.0449 * 3600 s/h * 8/per day * 240 per year = 303.223 KB = 0.2961 GB= 0.000289 TB/year/transaction

Thus, the proposed model (SMDSB) can minimize the storage cost of existing blockchain models by 99.89% in Bitcoin, 54.15% in Ethereum, and 64.19% in Hyperledger.

As shown in Fig. 9, we have applied SMDSB to reduce the storage size of existing blockchain storage models. We can observe that SMDSB achieves improvement in the storage by 81.54% in Bitcoin, 53.86% in Ethereum, and 62.59% in Hyperledger. We conclude that SMDSB is advantageous when sizes of original transactions vary significantly, which appears to be the case in most of the real-time scenario. Thus, it can be observed that, by utilizing SMDSB model in Bitcoin, Ethereum, and Hyperldeger, the storage size of blockchain network can be significantly reduced.

7 Conclusion and Future Enhancement

In this paper, we propose a storage model for data sharing in blockchain (SMDSB) using IPFS-based off-chain storage. In SMDSB, we can store structured and unstructured files as a transaction in the blockchain network. To reduce the size of transactions in the blockchain ledger, we store original files in IPFS off-chain storage, and the returned IPFS hash is mapped into the blockchain network. The proposed model is not only effective for Bitcoin transaction types but also for other types of transactions. We have compared our model with the storage model used in Bitcoin, Ethereum, and Hyperledger composer. It can be seen from experimental results that our storage model is optimal in terms of transaction storage size, availability (distributed off-chain storage by using IPFS), and synchronization (adding new nodes becomes easy owing to reduced size of transaction) of the nodes. Moreover, in case of file storage as a transaction, only files of size up to 100 KB are allowed to be uploaded with other models, whereas in SMDSB, we have successfully tested with upload of file of size up to 100 MB size into the blockchain network with the help of content-addressed hash mapping. By using SMDSB, the transaction size of Bitcoin can be reduced by 81.54%, Ethereum by 53.86%, and Hyperledger by 62.59%. In addition, the SMDSB model reduces the cost of per transaction storage (transaction per year) in Bitcoin by 99.89%, Ethereum by 54.15%, and Hyperledger by 64.19%. Furthermore, the SMDSB can also be used for a real-time application. In the future, we wish to use SMDSB model to store different types of bulky data generated by IoT devices and cyber-physical systems.

References

1. A. Tenorio-Fornés, S. Hassan, J. Pavón, Open peer-to-peer systems over blockchain and ipfs: an agent oriented framework, in *Proceedings of the 1st Workshop on Cryptocurrencies and Blockchains for Distributed Systems* (ACM, 2018), pp. 19–24

2. E. Androulaki, A. Barger, V. Bortnikov, C. Cachin, K. Christidis, A. De Caro, S. Muralidharan, Hyperledger fabric: a distributed operating system for permissioned blockchains, in *Proceedings of the Thirteenth EuroSys Conference* (ACM, 2018), p. 30
3. J. Benet, Ipfs-content addressed, versioned, p2p file system (2014). arXiv:1407.3561
4. J. Bonneau, A. Miller, J. Clark, A. Narayanan, J.A. Kroll, E.W. Felten, Research perspectives and challenges for bitcoin and cryptocurrencies (extended version). Cryptology ePrint Archive, Report 2015/452 (2015)
5. M. Brandenburger, C. Cachin, R. Kapitza, A. Sorniotti, Blockchain and trusted computing: problems, pitfalls, and a solution for hyperledger fabric (2018). arXiv:1805.08541
6. J.D. Bruce, The mini-blockchain scheme. White paper (2014)
7. M. Dai, S. Zhang, H. Wang, S. Jin, A low storage room requirement framework for distributed ledger in blockchain. IEEE Access **6**, 22970–22975 (2018)
8. A. Dorri, S.S. Kanhere, R. Jurdak, P. Gauravaram, Blockchain for IoT security and privacy: the case study of a smart home, in *2017 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom workshops)* (IEEE, 2017), pp. 618–623
9. W.G. Ethereum, A secure decentralised generalised transaction ledger. Ethereum Project Yellow Paper **151**, 1–32 (2014)
10. J. Gao, B. Li, Z. Li, Blockchain storage analysis and optimization of bitcoin miner node, in *International Conference in Communications, Signal Processing, and Systems* (Springer, Singapore, 2018), pp. 922–932
11. J. Göbel, A.E. Krzesinski, Increased block size and Bitcoin blockchain dynamics, in *2017 27th International Telecommunication Networks and Applications Conference (ITNAC)* (IEEE, 2017), pp. 1–6
12. https://www.ibm.com/downloads/cas/RXOVXAPM
13. J. Jogenfors, Quantum bitcoin: an anonymous, distributed, and secure currency secured by the no-cloning theorem of quantum mechanics, in *2019 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)* (IEEE, 2019), pp. 245–252
14. A. Kumar, A review on implementation of digital image watermarking techniques using LSB and DWT, in *Information and Communication Technology for Sustainable Development* (Springer, Singapore, 2020), pp. 595–602
15. R. Kumar, N. Marchang, R. Tripathi, Distributed off-chain storage of patient diagnostic reports in healthcare system using IPFS and blockchain, in *2020 International Conference on COMmunication Systems & NETworkS (COMSNETS)* (IEEE, 2020), pp. 1–5
16. R. Kumar, R. Tripathi, Implementation of distributed file storage and access framework using IPFS and blockchain, in *2019 Fifth International Conference on Image Information Processing (ICIIP)* (IEEE, 2019), pp. 246–251
17. A. Kumar, Design of secure image fusion technique using cloud for privacy-preserving and copyright protection. Int. J. Cloud Appl. Comput. (IJCAC) **9**(3), 22–36 (2019)
18. Q. Lin, H. Wang, X. Pei, J. Wang, Food safety traceability system based on blockchain and epics. IEEE Access **7**, 20698–20707 (2019)
19. S. Matetic, K. Wüst, M. Schneider, K. Kostiainen, G. Karame, S. Capkun, BITE: bitcoin lightweight client privacy using trusted execution. IACR Cryptol. ePrint Archive **2018**, 803 (2018)
20. A. Palai, M. Vora, A. Shah, Empowering light nodes in blockchains with block summarization, in *2018 9th IFIP International Conference on New Technologies, Mobility and Security (NTMS)* (IEEE, 2018), pp. 1–5
21. E. Palm, O. Schelén, U. Bodin, Selective blockchain transaction pruning and state derivability, in *2018 Crypto Valley Conference on Blockchain Technology (CVCBT)* (IEEE, 2018), pp. 31–40
22. P. Thakkar, S. Nathan, B. Viswanathan, Performance benchmarking and optimizing hyperledger fabric blockchain platform, in *2018 IEEE 26th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)* (IEEE, 2018), pp. 264–276
23. P. Yuan, X. Xiong, L. Lei, K. Zheng, Design and implementation on hyperledger-based emission trading system. IEEE Access **7**, 6109–6116 (2018)

24. S. Zhang, E. Zhou, B. Pi, J. Sun, K. Yamashita, Y. Nomura, A solution for the risk of non-deterministic transactions in hyperledger fabric, in *2019 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)* (IEEE, 2019), pp. 253–261
25. Q. Zheng, Y. Li, P. Chen, X. Dong, An innovative IPFS-based storage model for blockchain, in *2018 IEEE/WIC/ACM International Conference on Web Intelligence (WI)* (IEEE, 2018), pp. 704–708
26. Z. Zheng, S. Xie, H. Dai, X. Chen, H. Wang, An overview of blockchain technology: architecture, consensus, and future trends, in *2017 IEEE International Congress on Big Data (BigData Congress)* (IEEE, 2017), pp. 557–564