Received August 6, 2019, accepted August 26, 2019, date of publication August 30, 2019, date of current version September 12, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2938578

Blockchain-Based Outsourced Storage Schema in Untrusted Environment

KUN HAO[1], JUNCHANG XIN[1,2], ZHIQIONG WANG[3,4], KEYAN CAO[5], AND GUOREN WANG[6]
[1]School of Computer Science and Engineering, Northeastern University, Shenyang 110169, China
[2]Key Laboratory of Big Data Management and Analytics, Northeastern University, Shenyang 110169, China
[3]College of Medicine and Biological Information Engineering, Northeastern University, Shenyang 110169, China
[4]Neusoft Research of Intelligent Healthcare Technology Company Ltd., Shenyang 110167, China
[5]Information and Control Engineering Faculty, Shenyang Jianzhu University, Shenyang 110168, China
[6]School of Computer Science and Technology, Beijing Institute of Technology, Beijing 100081, China

Corresponding author: Junchang Xin (xinjunchang@mail.neu.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61472069, Grant 61402089, Grant 61602323, and Grant U1401256, in part by the China Postdoctoral Science Foundation under Grant 2019T120216 and Grant 2018M641705, in part by the Fundamental Research Funds for the Central Universities under Grant N161602003, Grant N180408019, and Grant N180101028, and in part by the Open Program of Neusoft Institute of Intelligent Healthcare Technology Company Ltd. under Grant NRIHTOP1802.

ABSTRACT Outsourced data, as significant service offered by the cloud service provider (CSP), can effectively facilitate the data owner (DO) overcoming the storage limitations on massive data. To ensure the availability of data, DO usually outsources the data replications to multiple CSPs (multi-CSPs) and utilizes a third party metadata management (TPMM) to dominate the metadata of the corresponding replications. However, during the outsourced procedures, DO can hardly confirm the confidence of the TPMM who may take some malicious behaviors to affect the reliability of data. Thus, DO inevitably faces data security issues caused by the over-reliance on the semi-trusted TPMM to manage the metadata of replications. In this paper, we focus on the problem of reliable outsourced data service among multi-CSPs in untrusted environment, that is, how to reliably store and verify the metadata of the data replications in untrusted multi-CSPs environment. To address the problem, we use the novel blockchain technology as a medium to build a trusted outsourced service platform. Moreover, we fully consider the innovative characteristics of blockchain including decentralized architecture, redundancy storage, collective maintenance, and tamper resistant to ensure the data cannot be changed maliciously. We first design a blockchain-based outsourced service framework for storing data replications in untrusted environment, which contains three key layers, that is, *storage layer*, *verification layer*, and *blockchain layer*. Then, we devise a novel concept of verification peer (VP) for maintaining metadata stored by a form of blockchain, and each of which holds the entire blockchain locally to prevent metadata from being maliciously tampered with. Finally, based on the proposed model, we introduce a collaborative algorithm invoked by VPs to store and verify the metadata of replications. We present a completed analysis and conduct extensive experiments on multi-CSPs scenario. The evaluation results demonstrate that our proposed approach achieves superior performance.

INDEX TERMS Blockchain, collaborative, outsourced data, reliable storage, untrusted environment.

I. INTRODUCTION

Outsourced data, as significant service offered by the cloud service provider (CSP), is increasingly gaining considerable attention in industry and academia, due to its convenience, low overhead, and high flexibility. The powerful success of outsourced data service is that CSP can provide distributed strategies to offer "infinite" storage capacity for data owner (DO) to overcome the limitations of storing massive data, thus reducing physical and economic costs. Based on outsourced data services, a mushrooming number of applications have been presented, such as Google App Engine [1], Microsoft Azure Platform [2], Amazon S3 [3] and Baidu Yun [4]. However, managing the data through a single CSP may be unreliable since DO cannot ensure the reliability of the CSP during utilizing the outsourced data services.

The associate editor coordinating the review of this article and approving it for publication was Vlad Diaconita.

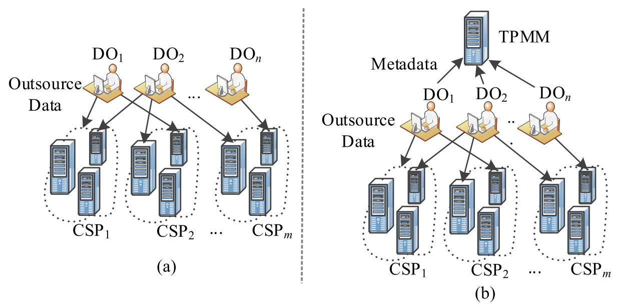


FIGURE 1. **Two architectures of mutli-CSPs outsourced data service models. (a)Traditional model where *DO_i* is the data owners and *CSP_j* are the Cloud Service Providers. (b) Third Party Metadata Management (TPMM) based model in multi-CSPs environment.**

Specifically, DO will encounter some complicated issues such as data migration, data error, and privacy disclosure if the untrusted CSP unilaterally shuts down or interrupts the services. For example, in October 2016, *Qihoo 360 Technology Co.* announced that it would close its *net-disk* services and notified all their users to migrate the data as soon as possible, which had brought great inconvenience. Furthermore, in February 2017, *Amazon* briefly suspended its outsourced data services, which led to high error rates in thousands of online applications. More seriously, in March 2018, *Facebook* divulged privacy profiles of nearly 50 million users to some analysis companies.

An intuitive way to improve the reliability of outsourced data service is to outsource some replications of the same data to multiple CSPs (multi-CSPs), and each of which independently maintains these data replications. Moreover, DO can get the metadata (e.g., physical locations) generated by multi-CSPs to locate and download all the raw replications [5]. However, as shown in Figure 1(a), this naive solution is inefficient and insecure due to the following reasons. First, DO must keep separate communications with each CSP to ensure the consistencies of all the replications. Second, DO need to manage the metadata of each replication that can be leveraged to access the physical address of the outsourced data. Finally, DO cannot discover whether the metadata of replications produced by the multi-CSPs have been maliciously tampered with or leaked.

A. MOTIVATION

To overcome the above shortcomings, DO usually utilizes a third party metadata management (TPMM) to dominate the metadata of the corresponding replications when finishing the outsourced procedures [6], [7]. As shown in Figure 1(b), DO first gets the metadata of all the replications by requesting multi-CSPs, and then sends entire metadata to a TPMM which is responsible for providing store and verification services. Obviously, the TPMM-based approach for metadata management can improve the reliability and availability of outsourced data at some extent. Nevertheless, during the outsourced procedures, DO can hardly confirm the confidence of the TPMM, and the semi-trusted TPMM may take some malicious behaviors to affect the reliability of data. Thus, DO inevitably faces data security issues caused by the over-reliance on the untrusted TPMM to manage the metadata of replications. First, the metadata of replications can be maliciously leaked or tampered with, and the semi-trusted TPMM may collude with multi-CSPs to cheat DOs by returning arbitrary results. Second, as increasing the concurrent requests, it is difficult to guarantee efficient throughput since all requests must be processed by the TPMM. Finally, DO will face the risk of losing the metadata when the TPMM occurs the single point of failure problem.

B. CHALLENGES

In this paper, we focus on the problem of reliable outsourced data service among multi-CSPs in untrusted environment, that is, how to reliably store and verify the metadata of the data replications maintained by multi-CSPs in untrusted environment. However, designing a solution to address the problem is non-trivial and need to tackle the following challenges. First, in order to avoid excessive reliance on the centralized party (i.e., TPMM), how to construct a decentralized storage schema to provide better secure and efficient outsourced data service is a great challenge. Moreover, to ensure that the metadata have not been maliciously tampered with, the second challenge is how to provide a trusted integrity verification service which can be invoked at any time to complete the integrity verification while ensuring high throughput. As mentioned above, the metadata generated by multi-CSPs will be confronted with security risk in the untrusted environment, e.g., malicious attackers can modify the raw data through the corresponding metadata. Thus, the final challenge is to provide necessary privacy preserving mechanism for metadata in the untrusted multi-CSPs environment. It should be noted that the privacy preserving here is different from the traditional concept, that is, the user can encrypt the metadata to ensure the privacy, and can verify the integrity of the metadata.

C. CONTRIBUTIONS

To deal with the above challenges, we use the novel blockchain technology as a medium to build a trusted outsourced service platform. Moreover, we fully consider the innovative characteristics of blockchain including decentralized architecture, redundancy storage, collective maintenance, and tamper resistant to ensure the data cannot be changed maliciously. We first design a blockchain-based outsourced service framework, namely **R**eliable **CO**llaborative **M**odel (RCOM), for storing data replications in untrusted environment, which contains three key layers, that is, *storage layer*, *verification layer*, and *blockchain layer*. Then, we devise a novel concept of verification peer (VP) constituting a collaborative network [5], [8], [9] for maintaining metadata stored by a form of blockchain, and each VP holds the entire blockchain locally to prevent metadata from being maliciously tampered with. Finally, based on the RCOM, we introduce a collaborative algorithm invoked by VPs to store and verify the metadata of outsourced replications.

To the best of our knowledge, we are the first to formally consider the reliable problem caused by adopting the centralized party for managing the metadata of replications and take advantage of the remarkable features of blockchain to solve the problem.

In summary, the specific contributions of this paper can be outlined as follows.

- We formally define the problem of reliable outsourced data service among multi-CSPs in untrusted environment, and the description of the problem consists of three parts including system model, threat model, and design goals.
- To address the problem, we first design a blockchain-based outsourced service framework for storing data replications in untrusted environment, namely RCOM, which contains three key layers, i.e., storage layer, verification layer, and blockchain layer. Then, we devise a novel concept of verification peer (VP) for maintaining metadata stored by a form of blockchain, and each of which holds the entire blockchain locally to prevent metadata from being maliciously tampered with.
- Based on the RCOM, we propose a collaborative algorithm which consists of two key phases, that is, metadata store and metadata verification. In the metadata store, the signed metadata are written into the blockchain, and the VPs synchronize the local state of the blockchain. In the metadata verification, the VPs return the corresponding verification results by retrieving the local metadata blockchain.
- We present a completed security analysis and conduct extensive experiments on multi-CSPs scenario. The evaluation results demonstrate that our proposed approach achieves superior performance.

The remainder of this paper is organized as follows. Section II gives the preliminaries of this work, and Section III introduces the problem statement. In Section IV, we introduce the architecture and data model of RCOM. In Section V, we propose a collaborative algorithm based on RCOM model. We show the experimental evaluations in Section VI. We provide an overview of related work in Section VII and conclude this paper in Section VIII.

II. PRELIMINARY

In this section, we give some preliminaries of this work including outsourced service and blockchain technology.

A. OUTSOURCED SERVICE

As we described before, this paper focuses on the problem of how to provide reliable data outsourced service in untrusted multi-CSPs environment. As shown in Figure 1, the traditional model of multi-CSPs environment contains three key roles, that is, data owner, cloud service provider, and third party metadata management.

- **Data owner (DO).** The DO has some sets of data and wants to transmit the massive data set d to the cloud service providers. To avoid the secure problems, DO usually outsourced replications of d denoted by $S_r = \{d_1, d_2, \cdots, d_k\}$ to some providers. Note that all data replications d_i in S_r are identical.
- **Cloud service provider (CSP).** The CSP has distributed strategies to manage the massive data set. According to the storage mechanism, we argue that once the data d_i is successfully stored, the corresponding metadata $metadata_i$ is generated.
- **Third party metadata management (TPMM).** The TPMM can manage the metadata $metadata_i$ of data replication d_i generated by the CSPs. Moreover, the TPMM provides an interface for the DO to query and verify $metadata_i$.

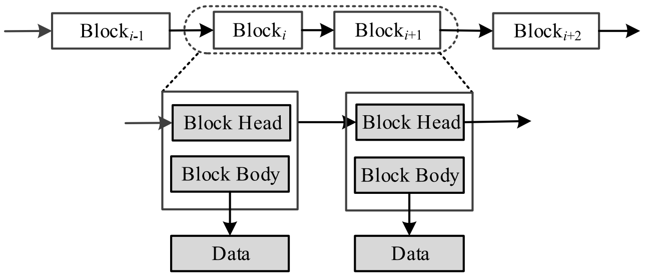


FIGURE 2. The blockchain structure just like a linked list where each *Block$_i$* contains two components, i.e., Block Head and Block Body. The Block Head store the digest information and the Block Body store the raw data.

B. BLOCKCHAIN TECHNOLOGY

Blockchain, as the foundation of cryptocurrency such as Bitcoin [10] and Ethereum [11], is considered to be a subversive innovational field of computer science. The features of blockchain include decentralization, redundancy storage, collective maintenance, and tamper resistant, and it can be viewed as an *append-only* distributed database maintained by a set of peers who do not fully trust each other [12]. Therefore, blockchain can be used to store and share information without a trusted central party [13], and it is expected to transform the Internet from an information model to a trust model [14]. Specifically, the core of the blockchain can be summarized into two parts, that is, the data structure and the consensus mechanism.

As shown in Figure 2, the data structure of blockchain consists of some time sequential blocks linked by hash computations, and each block includes two elements, i.e., block header and block body. The block header holds (i) the hash value of the previous block $PreHash$, (ii) the digest of the data stored in the current block (e.g., Merkle Root) $Digest$, (iii) the proof which is constructed by the miner $Proof$, and (iv) the timestamp when the specific block was constructed TS. The entire data in a given block head is shown as follow:

$$blockhead := (PreHash|Digest|Proof\ |TS)$$

On the other hand, the block body is responsible for storing raw data (e.g., financial transactions or account information). Particularly, all the block in a specific blockchain only depend on their previous one just like a linked list, and each blockchain has a special genesis block which has no predecessor block.

The other significant part of blockchain is the specific consensus protocols, e.g., POW [10] and POS [11], which maintained by a set of semi-trusted consensus nodes. In order to understand the basic technical concepts, we take the POW consensus protocol to describe the workflow of the blockchain. First, there are many nodes (i.e., miners) that don't trust each other in the blockchain network, and each of which has own computation power to perform hash calculation. Then all miners get the transactional data issued by users within TS. To write the valid data into the blockchain, these miners create a validation tree (e.g., Merkle Tree) and save the root value in the block header. Next, all miners simultaneously solve a "hash puzzle" and construct a candidate block with all validate data. The puzzle is to find a *nonce* to satisfy the following inequality:

$$Hash(blockhead|nonce) \leq Z$$

where Z denotes the mining difficulty which can be dynamically adjusted based on the computation power of the miners. Finally, the new block B^* can be linked by the $miner^*$ who first finishes the above computations, and $miner^*$ gets the write permission to link B^* at the end of the blockchain. Note that, for a given time, the network allows only one miner to update the blockchain, and other miners need to synchronize the local blockchain after the update is completed.

According to the above characteristics, the blockchain can be used to solve the reliable issues in the untrusted environment due to the following reasons. First, all nodes in the blockchain have identical obligations of storing, transmitting and verifying the data, which are entirely based on decentralized network architecture. Second, blockchain can be viewed as a reliable database by redundantly storing data locally, that is, each node has a complete replication of blockchain and maintains the consistency with each other at any time. Thus, malicious nodes must control more than half of the computation power of the entire blockchain network to modify the data, which is greatly difficult to achieve. Finally, the blockchain can preserve privacy and prevent data from being maliciously tampered with, that is, the data stored in the blockchain can be 1) preprocessed by cryptographic encryption, 2) correlated by hash calculations, and 3) validated using an effective mechanism by all nodes.

III. PROBLEM STATEMENT

In this section, we formally define the problem of reliable outsourced data service among multi-CSPs in untrusted environment, and the description of the problem consists of three significant parts including system model, threat model, and design goals.

A. SYSTEM MODEL

As shown in Figure 3, the system model of reliable outsourced data service in the untrusted environment contains three parties: (i) DO, (ii) multi-CSPs, and (iii) TPMM. We assume that DO outsource massive data by adopting outsourced data services of multi-CSPs to overcome storage limitations and desire for ensuring the reliability and availability of the raw data at any time. Specifically, in order to improve the reliability of data, DO outsource some replications of the data to multi-CSPs, and the metadata of all replications guarantee the raw data can be downloaded. It worth noting that, in this paper, the data replication refers to the DO outsourcing multiple replications of the same data to multi-CSPs, which is different from the concept in the distributed storage system. The traditional concept of data replication means that the CSP will store the data redundantly in order to ensure the high availability of the raw data, and the quantity of the replications depends on the specific distributed storage system. The essential difference is that the former one is DO-driven and the latter one is CSP-driven.

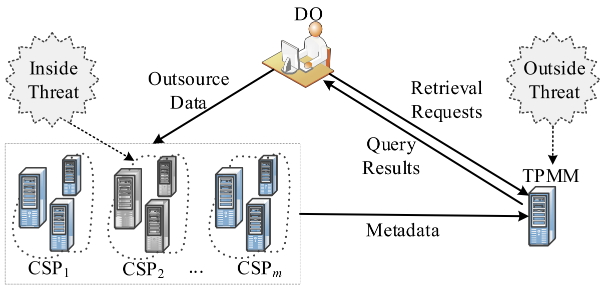


FIGURE 3. Outsourced model in the untrusted environment. The DO outsource the data replications to multi-CSPs, and the metadata of all replications are transmitted to the TPMM. The system includes two threat models., i.e., Inside Threat and Outside treat.

Additionally, all the data replications have the corresponding metadata to describe attribute information of the raw data, e.g., the physical locations. The metadata are generated by the multi-CSPs and used to guarantee that the original data can be downloaded by DO at any suitable time. Moreover, to ensure the reliability of the metadata of data replications, DO will adopt the centralized TPMM that provides the management services for entire metadata including storage, verification, and retrieve. As DO cannot determine the reliability of CSPs and TPMM, in this paper, we assume that the CSPs and TPMM are not fully trusted, which means that they can perform malicious operations on metadata without notifying DO. It should be highlighted that we assume that the raw data can be safely stored by multi-CSPs and will not be maliciously tampered with. Specifically, this paper only considers the security of metadata of outsourced replications and proposes storage and verification approaches in multi-CSPs environment.

B. THREAT MODEL

In this paper, we assume that the CSPs and TPMM are "semi-honest-but-curious", which means that they may execute misbehaved operations, such as returning fake results, performing collusion attacks, or maliciously tampering with, which lead to reducing the security of the outsourced services [15]. Note that we also assume the communication channels among DO and all CSPs are fully trusted, which indicates that the data (including metadata and raw data) can be transmitted securely and integrally. Therefore, since losing the supervision of the metadata of data replications by adopting the TPMM, we consider two types of threat model as shown in Figure 3:

- **Inside threat.** First, as the untrusted CSPs can arbitrarily modify metadata, they can return incorrect results to DO. Moreover, the untrusted CSPs can delete or forge metadata of data replications and return fake results. As a result, DO cannot download the raw data from incorrect or fake metadata.
- **Outside threat.** Second, as mentioned above, DO can utilize a TPMM which provides store and verification services for metadata of replications. However, DOs cannot fully confirm the confidence of TPMM, e.g., untrusted TPMM and CSPs can collude with each other to cheat DO. Clearly, the main reason is that DO, in order to reduce the computational workload, relies on the centralized untrusted TPMM excessively to hands over the integrity verification.

C. DESIGN GOALS

To avoid the above threats, our proposed approach (i.e., RCOM) has the following primary design goals with respect to full decentralization, collaborative maintenance, store reliably, result verifiable, and scalability.

- **Full decentralization.** RCOM avoids metadata reliability problems caused by excessive reliance on untrusted third parties so that avoiding single points of failure.
- **Collaborative maintenance.** To improve the availability, RCOM maintains metadata of replications in a reliably collaborative manner.
- **Store reliably.** RCOM grantees that the metadata of replications should be securely stored, which means that the metadata cannot be leaked or tampered with from untrusted CSPs in a decentralized scenario.
- **Result verifiable.** RCOM allows DO to detect the misbehavior (e.g., modification or forgery) when the untrusted CSPs intentionally return incorrect or fake search results. It should be noted that, unlike TPMM, RCOM can verify the metadata in a collaborative manner which significantly improves the integrity.
- **Scalability.** While fully guaranteeing the reliability of metadata of outsourced replications, RCOM provides high scalability for storage and verification.

In the following sections, we give details of our reliable mode and the collaborative approach that achieves above design goals while avoiding the threat models.

IV. RELIABLE MODEL BASED ON BLOCKCHAIN

In this section, we first outline the architecture of RCOM which contains a set of collaborative verification peers (VP) to maintain the metadata securely. And then, we design a novel data model for storing the metadata of the outsourced replications in a blockchain form.

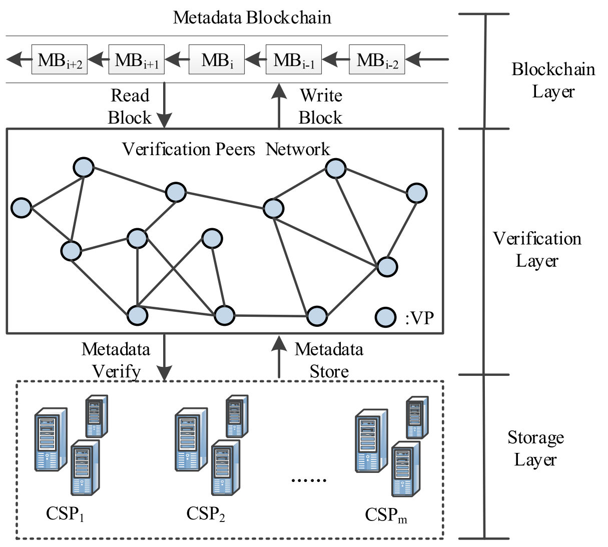


FIGURE 4. The architecture overview of RCOM that contains three key layers, i.e., Storage layer, verification layer, and blockchain layer.

A. ARCHITECTURE OVERVIEW

The core idea of RCOM is inspired by the blockchain. Similarly to the traditional blockchain that we elaborated before, RCOM can also be divided into two noteworthy parts namely the data model and the verification network. Additionally, RCOM has an extra part to represent the untrusted multi-CSPs who are responsible for generating the metadata of the outsourced replications. The overview of RCOM is shown as Figure 4, which consists of three key layers including storage layer, verification layer, and blockchain layer. In the following of this subsection, we describe design details and specific objective of each layer of the RCOM.

- **Storage Layer.** The bottom layer of RCOM consists of some CSPs. Each CSP has its own distributed storage system, such as HDFS [16] and Ceph [17], to manage massive data and generate metadata for replications. Note that since we only consider the reliability of metadata of outsourced replications, we omit the specific mechanisms that all CSPs used to store raw data. Moreover, we assume that after DO successfully outsourcing the raw data to the CSPs, it will generate the corresponding metadata including the physical locations of all replications, the number of replications, etc. Obviously, we can also use traditional databases to store raw data. However, as we have described, this paper only considers how to ensure the reliability of data replications after the user outsourcing the replications to multiple CSPs in order to avoid excessive reliance on the central node. This means that we ignore how the CSP stores data replications.
- **Verification Layer.** In order to address the reliable problem of outsourced data service among the untrusted multi-CSPs, we design a set of VPs constituting a collaborative network, and each VP stores and verifies metadata for DO in a blockchain form. In addition,

all VPs maintain a complete replication of blockchain locally and synchronize its state with global blockchain in real time. This layer is the core item of our proposed approach, and it provides two methods for managing metadata including store phase and verification phase. The specifications of these two phases will discuss in the following section.

- **Blockchain Layer.** This layer maintains the global state of metadata blockchain. All VPs in the collaborative network of RCOM can read this layer to synchronize the local state. Additionally, the global metadata blockchain is Read-Only, i.e., if any metadata changed, it must reconstruct new metadata blocks and rewrite the new metadata blocks into the global metadata blockchain. Note that the global blockchain here is a "virtual concept" that is not maintained by any VP. Moreover, the state of global blockchain can be approved by most of the VPs in the entire collaborative network at a given time t. The advantage of setting up a global blockchain is that all VPs do not rely on the central party to verify the local state of data.

Note that, since **Verification Layer** and **Blockchain Layer** are the key components to our proposed approach, we clarify the details of both parts in the followings.

On the one hand, in this paper, VPs replace the traditional blockchain nodes to verify and store metadata and maintain metadata blockchain locally. Functionally, VPs are similar to the traditional blockchain nodes. However, we have simplified the implementation details to make it more suitable for our defined scenarios (multi-CSPs). In addition, DO does not run any VP. The roles of the two nodes (DO and VP) are different. The former is mainly responsible for providing data to multi-CSPs, while the latter is mainly responsible for ensuring the integrity of the metadata of outsourced replications. In a nutshell, VP and its maintained metadata blockchain are part of our proposed trusted cloud service in multi-CSPs scenario.

On the other hand, we use a private model to build the **Blockchain Layer** for the following reasons: First, all VPs maintain communications with others forming a whole entity to collaboratively store and validate metadata. Second, to avoid security problems, our approach does not allow external nodes to join the system. Finally, since we can control the number of VPs in the system, it can improve the efficiency of verification of metadata of outsourced replications.

B. DATA MODEL

In this subsection, we give details of the data model of RCOM to reliably maintain metadata of outsourced replications. Compare to the basic definitions, in our proposed approach, we reshape the data unit, the block format, and the blockchain structure. Specifically, we design novel data models including signature metadata, metadata block, and metadata blockchain to organize the metadata of the outsourced replications. Recall that, in this paper, we consider the integrity of metadata of the outsourced replications which include information of raw data, e.g., physical locations and the quantity of replications. In the following, we discuss the definitions and relative theorems. The data model of RCOM is shown in Figure 5.

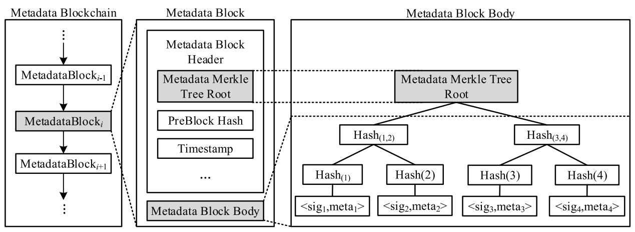


FIGURE 5. **Example of data model of RCOM. The left side is the entire Metadata Blockchain, the middle is detailed structure of a specific *MetadataBlock$_i$*, and the right side is the block body including raw data formed by a merkle tree of *MetadataBlock$_i$*.**

Definition 1 (Signature Metadata): Consider the DO d, sig_d represents the signature of d, and $meta_d$ denotes corresponding metadata of d. As shown in right side of Figure 5, signature metadata consists of two-tuples which represents by $< sig_d, meta_d >$.

Note that we omit the details of the specific cryptograph algorithms for signing the metadata of outsourced replications since this is not the main concern of this work. Moreover, we assume that the signing algorithms invoked by DO are absolutely security such that the signature metadata can be viewed as a fully trusted entity. Thus, according to **Definition 1**, we can generate the signature metadata for any metadata of outsourced replications. Next, we define the metadata block to store the signature metadata of DO.

Definition 2 (Metadata Block): Like the traditional definition of block, the metadata block (MB) also includes block body and block header. Specifically, in the metadata block body, all signature metadata of DO are constructed a Merkle Tree. And the metadata block header consists of hash digest computed by the head of the previous MB, the Merkle Tree Root of all signature metadata, the timestamp and other optional components. The middle of Figure 5 shows an example of MB.

After building the MB of all the given signature metadata, we now define the metadata blockchain that forms all MBs as a sequential structure that is linked by hash calculations.

Definition 3 (Metadata Blockchain): The metadata blockchain (MBC) is consisted of several MBs indicated by $\{MB_0, MB_1, \cdots, MB_m\}$, according to **Definition 2.**, each header of MB denoted by MB_{i+1}*->header* $(0 \leq i \leq m)$ which is calculated by hash computation with its previous MB_i*->header*, i.e., MB_{i+1}*->header* $=$ Hash $(MB_i.header)$, where $Hash(*)$ is a security hash algorithms (e.g., SHA256), such that it only depends on its previous one. Therefore, all MBs can be organized in series just like a linked list structure. Particularly, MB_0 is a pre-defined genesis metadata block which has no previous MB.

Below we give some theorems about metadata blockchain to demonstrate that it can satisfactorily guarantee the security and reliability of the metadata of outsourced replications.

Theorem 1: Given a set of signature metadata $S_{sigmeta} = \{< sig_1, meta_1 >, < sig_2, meta_2 >, \cdots, < sig_m, meta_m >\}$, the Merkle Tree Root generated by $S_{sigmeta}$ is unique.

proof: As the definition of Merkle Tree [18], nodes except the leaf are made up of the hash values of their child nodes. All nodes from the penultimate layer to the root are calculated by their child nodes through hash calculation. For example, in the right side of Figure 5, $Hash(1, 2) = Hash(Hash(1) \cup Hash(2))$, where $Hash(*)$ is a secure hash algorithm. According to the uniqueness of hash algorithm, the root node changes along with variation of any leaf node. □

Theorem 2: Given a MBC $= \{MB_0, MB_1, \cdots, MB_m\}$, if we modify any metadata block MB_i in MBC, where $i \in [0, m]$, all previous and successor blocks of MB_i must be modified simultaneously.

proof: According to **Definition 3.**, the block MB_i is calculated by its previous block, and the successor blocks are constructed by MB_i. Since the uniqueness of the hash algorithm, we need to recalculate all the previous and successor blocks MB_j, where $j \in [0, i-1]$, if modified any block in the MBC. □

Theorem 3: The block header of each block is calculated by hashing the stored data. If the block headers of the blocks in the two blockchains are the same (i.e., have same hash values), the states of the two blockchains is consistent.

proof: According to **Definition 2.**, given two metadata blockchain, $MBC_1 = \{MB_{11}, MB_{12}, \cdots, MB_{1m}\}$ and $MBC_2 = \{MB_{21}, MB_{22}, \cdots, MB_{2m}\}$. MBC_1 and MBC_2 have the same state only if $MBC_1.MB_{1i}\text{->}Header = MBC_2.MB_{2i}\text{->}Header$, where $i \in [0, m]$. □

Now, we take some examples to illustrate the data models in RCOM. In a specific time period t, given a DO set $S_{DO} = \{DO_1, DO_2, DO_3, DO_4\}$, $S_{sig} = \{sig_1, sig_2, sig_3, sig_4\}$ represents the corresponding signatures of S_{DO}, and $S_{meta} = \{meta_1, meta_2, meta_3, meta_4\}$ are metadata of all replications. We can get the signature metadata tuples for each DO, that is, $S_{sigmeta} = \{< sig_1, meta_1 >, < sig_2, meta_2 >, < sig_3, meta_3 >, < sig_4, meta_4 >\}$. To make a MB, we can compute hash value for each element in $S_{sigmeta}$. The right side of Figure.5 shows the process for making a MB. From bottom to top, each two elements in $S_{sigmeta}$ compute hash value recursively to build a Merkle Tree. Then we can get unique Merkle Tree Root for $S_{sigmeta}$. According to **Theorem 1.**, if the value of any leaf node is modified, the Merkle Root also changes. The middle of Figure 5 shows a example of MB.

As shown in the left side of Figure.5, according to **Theorem 2.**, if we modify any metadata block, we must recalculate the previous and successor blocks of MB_i. In addition, assuming that there are two nodes which denoted by v_1 and v_2, and each node holds a replication of MBC. The replications of MBC which store locally by v_1 and v_2 are denoted by $v_1.MBC_L$ and $v_2.MBC_L$. According to the **Theorem 3.**, if the header information of metadata block stored in $v_1.MBC_L$ and $v_2.MBC_L$ are equal, which means that $v_1.MBC_L\text{->}Header = v_2.MBC_L\text{->}Header$, we can argue that v_1 and v_2 have the same state of the MBC.

TABLE 1. Parameters and key words of algorithms.

Parameters	Descriptions
DO	data owner
CSP	cloud service provider
VP	verification peer
LBC_i	local blockchain of VP_i
GBC	global blockchain
F	outsourced file
F^*	processed file of F
sig_j	signature of DO_j
$metadata$	the metadata of the replication of given F
pub_k/pri_k	cryptographic key pairs

In the following section, we introduce the collaborative algorithm adopted by a set of VPs to store and verify the signature metadata maintained in the metadata blockchain.

V. COLLABORATIVE ALGORITHM

Based on RCOM model, we design a collaborative algorithm which includes two key phases, that is, metadata store and metadata verification. In the metadata store phase, each VP collects signature metadata for constructing metadata block and writes to local metadata blockchain. In the metadata verification phase, all VPs check whether local metadata blockchain state is consistent with the global state and retrieve corresponding metadata by signatures. In the following, we give details and examples of both phases. Table 1 gives the parameters and key words of all the algorithms.

A. SETUP

We first define some prerequisite functions for setting up our proposed algorithm including $KeyGen()$, $Sign()$, $Transmit()$, and $MetadataGen()$. The specific definition of each function is shown as follows.

- $(pub_k, pri_k) \leftarrow KeyGen()$. Each DO needs to execute the key generation function for receiving public key pub_k and private key pri_k. All DOs distribute their pub_k to the collaborative network and keep pri_k secretively. We assume that each VP can get the public keys of all DOs.
- $F^* \leftarrow Sign(F, pri_k)$. The function signs the raw file F by utilizing the privative key pri_k of DOs and returns the signed file F^*.
- $Transmit(F^*, CSPs)$. The function allows DOs outsource signed file F^* to multi-CSPs. We assume that the replications of data received by each CSP should be consistent. The procedures for managing these data replications are transparent to the DOs. However, DOs can get the physical location information and other parameters by utilizing the metadata.
- $Metadata_d \leftarrow MetadataGen()$. The function gets metadata of replications for DO d which are stored in multi-CSPs and sends all metadata to the collaborative

network. Especially, the metadata can be utilized by DOs to check the physical locations of outsourced data.

After invoking the required functions, the DOs' signatures and corresponding metadata of data replications can be transmitted to the collaborative network. Then VPs can adopt store algorithm to maintain the metadata locally and invoke the verify algorithm to check the integrity of the metadata collaboratively. We give the details of the algorithms in the following subsections.

B. METADATA STORE

In this subsection, we discuss metadata store algorithm based on RCOM. The detailed procedure is shown as follows.

Consider a DO set $S_{DO} = \{DO_1, DO_2, \cdots, DO_m\}$, $S_F = \{F_1, F_2, \cdots, F_m\}$ represents the raw data of each DO respectively. $S_{VP} = \{VP_1, VP_2, \cdots, VP_s\}$ denotes a set of verification peers constituting a collaborative network. $S_{CSP} = \{CSP_1, CSP_2, \cdots, CSP_n\}$ denotes multiple CSPs, and each of which keeps communications with others in the collaborative network.

We suppose that all DOs in S_{DO} generate pub_k and pri_k by invoking $KeyGen()$ and sign their own data by using $Sign()$. Then, DOs transmit signed data to multi-CSPs by utilizing $Transmit(S_F, S_{CSP})$. Each VP in S_{VP} gets the metadata for DOs' from CSPs by invoking $MetadataGen()$. We denote the metadata set of all data replications is $S_{Meta} = \{meta_1, meta_2, \cdots, meta_m\}$. Next, each VP maintains a set denoted by $S_{sigmeta} = \{< sig_1, meta_1 >, < sig_2, meta_2 >, \cdots, < sig_m, meta_m >\}$. And then, all VPs validate all signatures in $S_{sigmeta}$ by using corresponding pub_k of each *DO*. The VP who finishes the validation procedure broadcast message_{succ} to others.

After receiving quorum amount of messages (more than half of S_{VP}, for simplicity), the VPs terminates the current validation process and waits for the next request. Specially, we suppose that S_{succ} represents the peers who finish the validation procedure. Randomly choosing a VP in S_{succ}, namely VP_{succ}. Then, VP_{succ} utilizes $S_{sigmeta}$ to construct a Merkle Tree, and generate a metadata block *MB*. Finally, VP_{succ} writes the *MB* into the local metadata blockchain denoted by $VP_{succ}.MBC_L$. The VP_{succ} gets the *write permission* for linking the *MB* to global blockchain MBC_G and broadcasts message $message_{sync}$ for notifying other VPs. When received $message_{sync}$, all VPs synchronize their local metadata blockchains to match the global blockchain state. Clearly, when a specific DO_i requires to outsource new data $F^*_{DO_i}$ to CSPs, all *CSP*s in S_{CSP} generate corresponding new metadata $meta_{new}$ of $F^*_{DO_i}$ and broadcast it to S_{VP}. Then, all *VP*s verify and store the $meta_{new}$ into last metadata blockchain MB_{last}. The pseudo code is shown as **Algorithm 1**.

It worth noting that the Metadata Store algorithm can be seen as a consensus protocol. The reasons why we design a new consensus protocol to replace the existing protocols (e.g., POW [10] or POS [19]) are as follows. First, POW and POS are mainly for financial scenarios. Differently, our proposed protocol fully considers the storage and verification of metadata of outsourced replications in multi-CSPs scenario. Second, the reliability of the POW depends on the number of nodes in the network, that is, the more nodes, the higher the security. And the reliability of POS protocol depends on the credibility of the highest-stake node. Unlike these two protocols, in our approach, VPs obtain "write-permission" through competition verification. In addition, as we have described before, we use a private blockchain model to manage these VPs to maximize the internal security of the system. Finally, the POW and POS protocols are more complex to implement, that is, the former requires a lot of hash calculations, while the latter requires periodic calculation of the nodes' stakes in the network. In contrast, our approach only requires VPs to ensure communication with each other to achieve agreement. All VPs collaboratively collect the signatures S_{sig} and S_{meta} simultaneously for a period of time t.

Algorithm 1 Metadata Store

```
Input: DO set $S_{DO} = \{DO_1, DO_2, \cdots, DO_m\}$;
       outsourced data set $S_F = \{F_1, F_2, \cdots, F_m\}$;
       verification peers set
       $S_{VP} = \{VP_1, VP_2, \cdots, VP_n\}$
1  for Each DO in $S_{DO}$ do
2      Generate keys by using $(pub_k, pri_k) \leftarrow KeyGen()$
       and sign data with $Sign(F_i, pri_k)$;
3      Outsource all signed file to $CSPs$ using
       $Transmit(F, CSPs)$;
4  for Each VP in $S_{VP}$ do
5      Get metadata from CSPs using
       $Metadata_d \leftarrow MetadataGen()$;
6      Construct $S_{sigmeta} = \{< sig_1, meta_1 >, < sig_2,$
       $meta_2 >, \cdots, < sig_m, meta_m >\}$;
7      if not receives quorum $\text{message}_{succ}$ then
8          $result$ = Verify($S_{sigmeta}$);
9          if result then
10             Generate a Merkle Tree for $S_{sigmeta}$ and
               build a $MB$;
11         Broadcast $message_{succ}$ to other nodes and get
           write permission;
12         Writes metadata block into $MBC_G$;
13         Broadcast $message_{sync}$ to other peers;
14     else
15         if receives $\text{message}_{sync}$ then
16             Synchronize the state of local meta
               blockchain $MBC_L$;
17 if Specific $DO_i$ requires to outsource new data $F^*_{new}$ then
18     Get new metadata $metadata_{new}$ of $F^*_{new}$ and
       broadcast $metadata_{new}$ to all $VPs$
19     All $VPs$ recall the line 7 to line16 to construct $MB$
       using $metadata_{new}$
```

Complexity Analysis: The **Algorithm 1** can be divided into two part, that is, the upload part (Line 1-3) and the store part (Line 4-19). Therefore, the complexity analysis includes two aspects. Consider the first aspect, we assume the keys generation (Line 2) is linear. When getting the keys, all DOs transmit the file to all s CSPs (Line 3). The time complexity of this procedure is $O(m^2 s)$. Note that we omit the communication costs of the transitions of all data replications. For the second aspect, all VPs get the metadata and construct the $S_{sigmeta}$, and the time complexity is $O(n)$ (Line 5-6). Then, each VP verifies the $S_{sigmeta}$ and broadcast messages to others, and the communication cost is $O(n^2)$ (Line 7-16). Here, we consider the worst case for a specific VP writing the new block. Thus the entire complexity of **Algorithm 1** is $O(m^2 s + n^2)$.

Example 1: In the following, we give an example to illustrate the process of **Algorithm 1**. Consider a DO set $S_{DO} = \{DO_1, DO_2, DO_3, DO_4, DO_5\}$, corresponding signatures $S_{Sig} = \{sig_1, sig_2, sig_3, sig_4, sig_5\}$, and a verification peers set $S_{VP} = \{VP_1, VP_2, VP_3, VP_4, VP_5\}$. First, all VPs in collaborative network construct signature metadata set which denoted by $S_{sigmeta} = \{< sig_1, meta_1 >, < sig_2, meta_2 >, < sig_3, meta_3 >, < sig_4, meta_4 >, < sig_5, meta_5 >\}$. And then, each VP in S_{VP} validates all signatures in $S_{sigmeta}$, and sends $message_{succ}$ to others when it finishes the validation procedures. Without loss of generality, we assume that $S_{succ} = \{VP_1, VP_3, VP_5\}$ send $message_{succ}$ to others. Therefore, VP_2 and VP_4 stop their current validation process since they receive more than quorum (half of S_{VP}) of $message_{succ}$. Picking a VP in S_{succ}, assuming VP_3, to build a metadata block MB and write the metadata block to global metadata blockchain MBC_G. Finally, VP_3 broadcasts $message_{sync}$ to other VPs for synchronizing their local blockchain.

C. METADATA VERIFICATION

In this subsection, we introduce the algorithm for metadata verification. The algorithm allows DOs for verifying the metadata of data replications at any given time. Consider a DO d and corresponding signature sig. $S_{VP} = \{VP_1, VP_2, \cdots, VP_s\}$ denotes a VP set. The detailed procedure of metadata verification is shown as follows, and the pseudocode is shown as **Algorithm 2**.

First, DO sends sig to the collaborative network, and each VP checks whether the state of local metadata blockchain MBC_L is consistent with global state MBC_G. If the local metadata blockchain is inconsistent, the VPs must retrieve the blockchain layer of RCOM to synchronize the local metadata blockchain. Otherwise, all VPs with consistent state retrieve the local blockchain to obtain the corresponding metadata. When getting all metadata for DO, the VPs broadcast the $message_{succ}$ and $digest$ which are computed by $metadata$ to other VPs. When receiving quorum amount of $messages_{succ}$ and the same $digests$, VPs stop current retrieving process. Finally, returning the $metadata$ to DO. Note that, when retrieving MBC_L using sig, all VPs need to return entire $metadata_{sig}$ stored in different metadata blocks MBs since that, in the metadata store phase, it may exist multiple metadata corresponding to sig in multiple MBs.

Algorithm 2 Metadata Verification

```
Input: The signature of DO $sig$; verification peers set
       $S_{VP} = \{VP_1, VP_2, \cdots, VP_n\}$
Output: Metadata of outsourced data replications
        $resultset$
1  Do send $sig$ to all $VPs$ in $S_{VP}$;
2  for each $VP$ in $S_{VP}$ do
3      if digests and $message_{succ}$ not exceed quorum of $S_{VP}$
       then
4          checks the state of local blockchain $MBC_L$ with
           $MBC_G$;
5          if the state is inconsistent then
6              synchronizes the local state of metadata
               blockchain;
7          else
8              $resultset$ = retrieveLocalMBC($sig$);
               /*including all $metadata$ signed by $sig$*/
9              if get resultset successfully then
10                 Broadcast $digest$ and $message_{succ}$ to
                   other $VPs$;
11 return $resultset$
```

Different to traditional blockchain which guarantees the consistency of the on-chain data, **Algorithm 2** is built on top of the metadata blockchain, which provides DO with a way to verify its own metadata. During the storage phase, metadata has been successfully written to the metadata blockchain. As we mentioned earlier, due to some unavoidable reasons (such as network latency), a transient inconsistency of state can occur when synchronizing the metadata blockchain. If the DO needs to verify the state (i.e., consistency) of the metadata at this point, then **Algorithm 2** needs to be called to synchronize the metadata blockchain first. Therefore, at any time, **Algorithm 2** can verify the integrity of the metadata.

Complexity Analysis: To verify the metadata of replications, DO sends the sig to all VPs (Line 1). Similarly, we omit the communication costs of transmitting the keys to all VPs. Then, each VP checks the state of local blockchain with others and synchronizes the inconsistent state. We also consider the worst case, that is, all VPs need to synchronize the local state. The upper bound of this progress is $O(n^2)$ (Line 3-6). Finally, VPs retrieve the local metadata blockchain to get the corresponding metadata of replications, and the time complexity of retrieving is $O(1)$. The reason for the constant complexity is that all metadata stored in the local metadata blockchain is a "key-value" tuple where the key is sig and the value is the metadata. Thus the entire complexity of **Algorithm 2** is $O(n^2)$.

Example 2: We also use an example to illustrate **Algorithm 2**. Given a verification peers set denoted by $S_{VP} = \{VP_1, VP_2, VP_3, VP_4, VP_5\}$. In addition, the notation $VP_i.MBC$ represents the local blockchain of VP_i, where $i \in [1, 5]$. DO transmits request and *sig* to all VPs in S_{VP}, and each VP checks its local state of metadata blockchain. We assume that $VP_3.MBC_L$ is inconsistent with MBC_G. Therefore, VP_3 must synchronize the local metadata blockchain. Other VPs retrieve their local blockchain to obtain corresponding metadata for *sig*. Supposing that $S_{succ} = \{VP_1, VP_4, VP_5\}$ get the *metadata* of *sig* and *digest* of *metadata*. Next, $\{VP_1, VP_4, VP_5\}$ broadcast $message_{succ}$ and *digest* to VP_2. Then, VP_2 checks whether all receiving *digests* for *metadata* are consistent. Supposing that they are equal, VP_2 skips the retrieve process for *sig* since it receives more than quorum amount (that is 3) of $message_{succ}$ and *digests* of *metadata*.

D. ANALYSIS

In this subsection, we give the analysis including data integrity, data accuracy, and data traceable, to prove that the proposed algorithm can avoid the threats and achieve the design goals, which are presented in **Section III**.

1) DATA INTEGRITY

In the metadata store phase, VPs validate the signatures collaboratively. The metadata with invalid signatures cannot be stored in the metadata blockchain. Moreover, if the signatures changed, e.g., replacing the signature algorithm or cryptographic device, the signature metadata must be regenerated by using the new signatures. According to **Definition 2.**, the data in the metadata block are constructed as a Merkle Tree consisting of $< sig_c, meta_c >$, where sig_c represents the signature and the metadata of c. By **Theorem 2**, the Merkle Tree Root changes along with sig_c or $meta_c$, which ensures the integrity of the metadata.

2) DATA ACCURACY

After validating given signatures, according to **Algorithm 1**, each VP can modify the global blockchain only if it receives more than a specific number of messages. Additionally, at any time, only one VP can update the global metadata blockchain, which avoiding the blockchain fork. Furthermore, all VPs periodically synchronize the local state of metadata blockchain in order to maintain data consistency. Such an approach ensures that the metadata stored in all VPs are consistent. Therefore, the metadata returned by VPs in the collaborative network of RCOM is accurate for any given signature.

3) DATA TRACEABLE

In the RCOM model, DOs upload the data and signatures to CSPs who provide storage services, and CSPs send metadata to the collaborative network. Therefore, the metadata can be obtained by the corresponding signature. According to **Algorithm 2**, as long as more than half of the peers in the collaborative network are working normally, we can get the corresponding metadata for replications. Given a metadata blockchain $S_{MBC} = \{MB_1, MB_2, \cdots, MB_m\}$ where MB_i $(i \in [0, m])$ represents metadata blocks, according to **Definition 1**, we can obtain the corresponding metadata for a signature, which means that the metadata in the RCOM is traceable.

TABLE 2. Experimental parameters.

Parameters	Default	Variation
Dataset (MB)	200	50, 100, 200, 500, 1000
Quantity of CSPs	5	3, 5, 7, 9
Quantity of VPs	20	5, 10, 20, 30, 50

VI. EXPERIMENTS

A. EXPERIMENTAL SETTING

We use HDFS 2.6 cluster to simulate multi-CSPs service environment to evaluate our schema. The cluster contains one master node and eight slave nodes, and each node has Intel (R) Xeon E5-2620 CPU @ 2.0GHz, 32G RAM, and Ubuntu 18.04 LTS OS. We assume that each experimental node is responsible for storing raw data for DOs. We use Java to implement RCOM model and collaborative algorithms and simulate a number of VPs by multi-thread method to constitute a collaborative network.

It worth noting that there are already more mature blockchain systems (e.g. Ethereum and Hyperledger). However, these systems cannot be directly applied or embedded in our proposed solution due to the following reasons. First, in the traditional blockchain system, the block structure is more complicated. Conversely, we define a minimized block structure while ensuring metadata integrity verification. Second, the consensus protocols in these blockchain systems are more complex, which greatly affects the storage and retrieval efficiency of on-chain data. Thus, we have designed easy-to-follow and efficient collaborative algorithms to replace complex consensus protocols. Finally, since the different application scenarios, embedding these huge systems directly into our proposed model is unrealistic and not easy to implement. Therefore, we designed and implemented our own minimal blockchain system to suit multi-CSP scenario to ensure the security of metadata of outsourced replications.

We conduct three main groups of experiments including store performance, verification performance, and scalability. Moreover, we compare our proposed solution with state-of-the-art approach. In each group of the experiment, we assume that DOs upload the same quantity of data to CSPs. The experimental dataset and parameters are shown in Table 2.

B. PERFORMANCE EVALUATION

1) STORE PERFORMANCE

In order to evaluate store performance, we use default size of dataset and quantity of CSPs to test the concurrent ability of metadata store. The results are shown in Figure 6, where *VP-X* represents the quantity of VPs (the same notations are used in the following of this section). Figure 6 shows that when using the default size of data, with the increase in

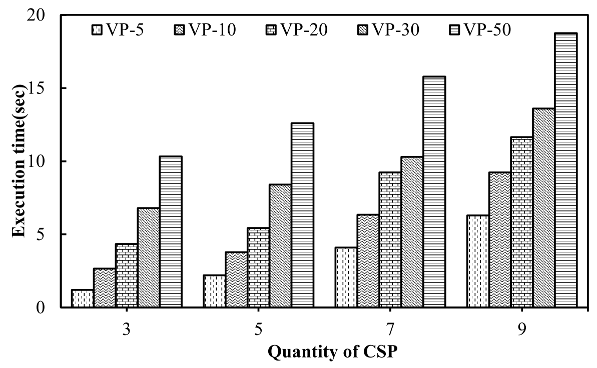


FIGURE 6. Store performance comparisons when using default data size.

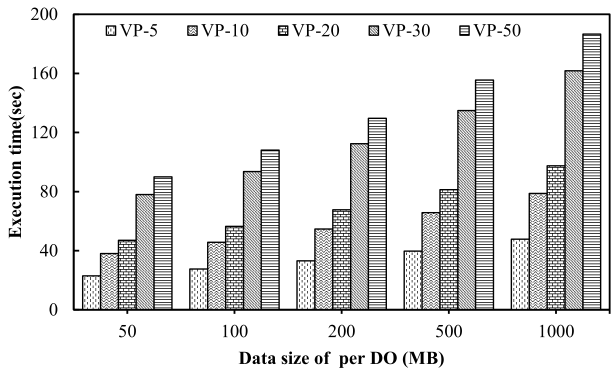


FIGURE 7. Store performance comparisons when using default CSPs quantity.

quantity VPs and CSPs, the execution time of store data grows linear. In Figure 7, when using the default quantity of CSPs, the efficiency increases linearly with the quantity of VP and the size of data.

From the above evaluations, we argue that the store performance can be affected by the data size. The reason for this result is shown as follows. In this paper, we assume that multiple users outsource data to multi-CSPs which are reasonable for storing the raw data redundantly and generating the metadata of replications. Clearly, the larger the size of data, the greater the time overhead incurred by multi-CSPs to store replications and generate corresponding metadata. Thus the efficiency of RCOM to store the metadata will be affected by data size.

2) VERIFICATION PERFORMANCE

To evaluate the verification performance, we conduct experiments with default dataset and quantity of CSPs. Figure 8 shows that, with the increase of the quantity of VPs, the verification efficiency for specific dataset increased. The reason is that VPs need to validate the corresponding signatures and retrieve the local blockchain. Figure 9 shows that the verification efficiency increase with the quantity of CSPs. When VPs is 50, the growth trend is faster since VPs need to broadcast messages to notify others. In addition, VPs will synchronize the local metadata blockchain, and it requires more time to synchronize the local metadata blockchain as the DOs increases.

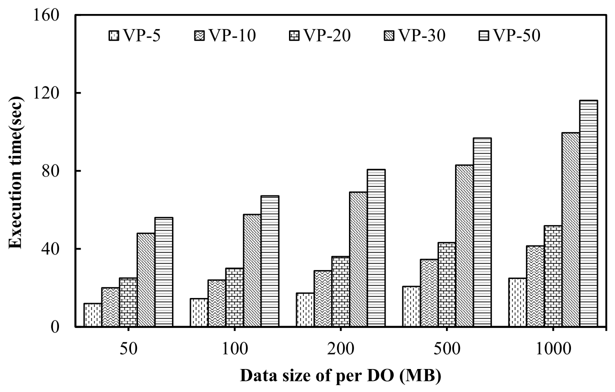


FIGURE 8. Verification performance comparisons when using default CSPs quantity.

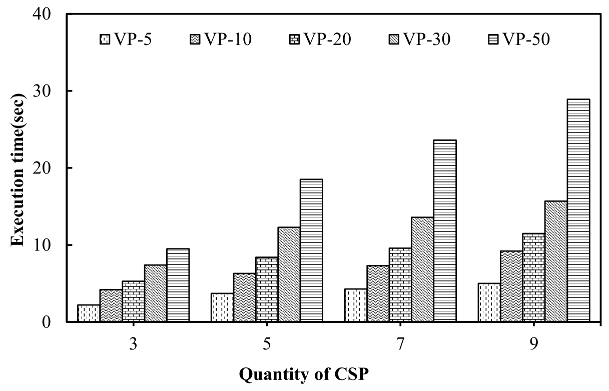


FIGURE 9. Verification performance comparisons when using default data size.

From the above experiment, it can be seen that RCOM has a better verification efficiency of metadata. When the storage node increases, the verification efficiency of RCOM is less affected. Moreover, when the number of verification peers increases, the efficiency of RCOM model is affected, but this effect is small. Finally, when DOs validate for a specific volume of data, the RCOM model can guarantee better concurrency verification efficiency. Note that the most direct factor affecting the efficiency of verification is the overhead of retrieving data on the blockchain and some unavoidable factors, e.g., network latency.

Similarly, the verification efficiency of RCOM is also affected by data size. The reason is that when larger data is stored in multi-CSPs, it is divided into some data blocks for redundant storage. The metadata records information about these data blocks including the physical locations and the block size. In RCOM model, these data are stored in the blockchain by the VPs. When the DO needs to verify, the VPs retrieve their local metadata blockchain to verify the integrity. Obviously, the larger data size, the larger of the generated metadata, which leads to affect the verification efficiency.

3) SCALABILITY

We evaluate scalability by adopting different quantity of CSPs. We use the default quantity of VPs. The results are shown in Figure 10, where *DS-Y* represents the dataset.

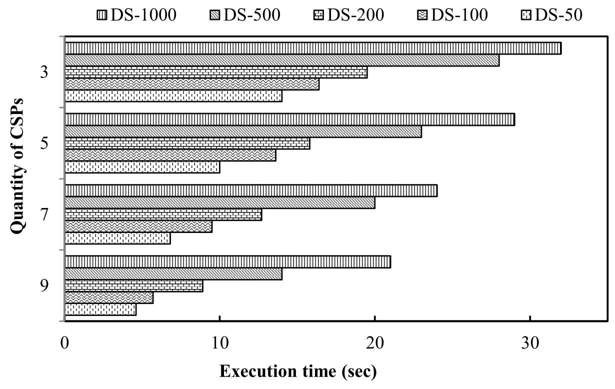


FIGURE 10. Scalability evaluations of metadata store.

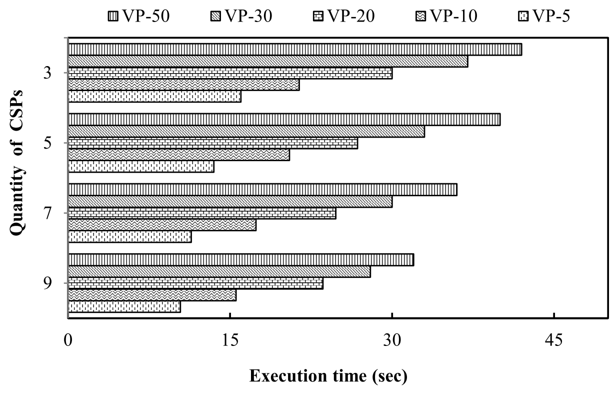


FIGURE 11. Scalability evaluations of metadata verificaion.

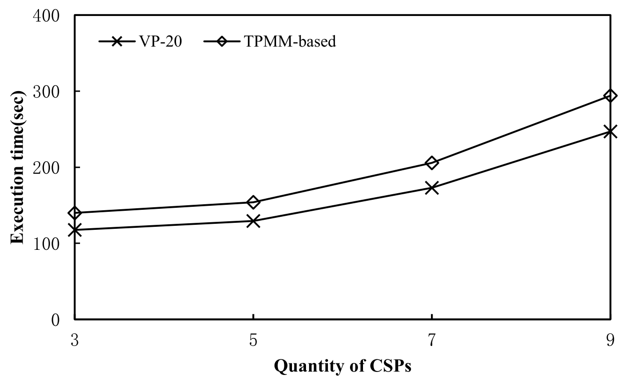


FIGURE 12. Comparisons of store efficiency with TPMM-based solution.

Figure 10 shows that the execution time of the generated metadata blocks decreases with the different quantity of dataset and CSPs. However, the overall reduction trend is great fast. When the quantity of CSPs is small, the latency of generating metadata is high. Similarly, we test the scalability of verification metadata by utilizing the different quantity of VPs. As shown in Figure 11, the verification efficiency is nearly inversely proportional to the quantity of CSPs.

4) COMPARISONS WITH STATE-OF-THE-ART

As we elaborated before, the basic solution to ensure the reliability of the replications is to involve a TPMM to store and verify the metadata of all replications. Therefore, we compare our method with the TPMM-based scheme [7].

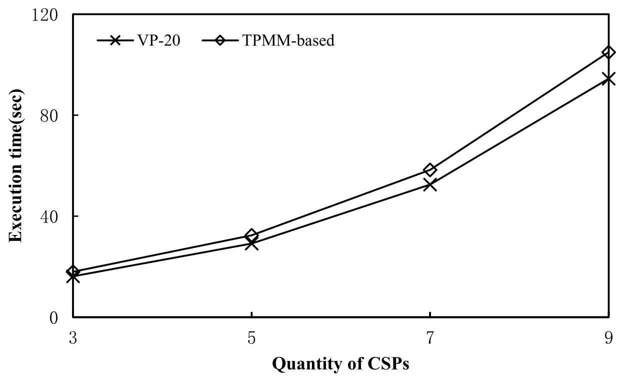


FIGURE 13. Comparisons of verification efficiency with TPMM-based solution.

In the comparison experiments, we take an additional physical machine to act as the TPMM and specify that all metadata of replications need to be stored and verified by the TPMM. Without loss of generality, we use the default dataset and quantity of VPs. The experimental results are shown as Figure 12 and Figure 13. From the experimental results, we can argue that, when the quantity of CSPs increases, the efficiency of store and verification of the two solutions grow linearly, and the trend is relatively flat. In addition, our approach is slightly better than the TPMM-based approach for storing and verifying metadata of replications. The reason for this is that in our outsourced service model, all VPs collaboratively store and verify all metadata formed by blockchain. In the TPMM-based method, both metadata store and verification are performed in the same node (i.e., TPMM), which inevitably lead to reduce the service efficiency. In contrast, our approach is to store and verify metadata in parallel, thus ensuring the efficiency of outsourced services.

Summing up, when using HDFS to simulate the multi-CSPs environment, RCOM has high scalability when changing the quantity of CSP nodes. Our solution is the first work to bring blockchain technology into data management in untrusted multi-CSPs environment, and we provide a baseline for future researches.

VII. RELATED WORK

Blockchain technology has widely attention both in academic and industry, and it is also viewed as the basis of the next generation of cloud computing [20]. The current researches focus on using the features of blockchain to solve traditional problems, such as IOT and data management. In [21] and [22], the authors summarize the challenges and solutions of the blockchain application in the Internet of Things and give the prospect of blockchain application. Blockchain-IoT [23] proposes the concept of a variation network based on blockchain, which improves the safety and efficiency of the traditional Internet of Things. In [24], a blockchain-based embedded device firmware update method has been proposed, which allows the devices to safely check the firmware version, verify the correctness of the firmware, and download the latest firmware version. Blockstack [25] adds features of blockchain to traditional DNS service. Additionally, Blockstack proposes a secure global naming and storage system to query and modify the data through Virtual Chain [26], [27]. In order to solve the integrity problem in the central management of the Internet, [28], [29] proposes a decentralized data security sharing network system, which effectively improves the efficiency and security of data. In [30], the authors propose a decentralized data storage model based on Proofs-of-Retrievability (PORs) [31], which improves the data reliability. In [32], the authors propose a decentralized data storage model for metadata store, which improves the security of cloud storage services. To ensure the data integrity, the authors in [33] provide a block-based storage network by encrypting and signing the raw data and save blocks into the P2P file system. The authors in [34] present a decentralized management system by using blockchain to preserve privacy. García-Barriocanal *et al.* [35] review and evaluate the functions of metadata in blockchain combined with other related technologies, which obtain a decentralized solution for metadata.

Similarly, in the database domain, the blockchain technology has made some significant research evolution [36]–[38]. Technically, there is a great correlation between blockchain databases and the distributed databases, that is, the mature techniques in distributed databases can be used to optimize the efficiency of blockchain databases [39]. BigchainDB [40] combines traditional distributed database with blockchain, and it improves the security of data and solves the capacity of blockchains simultaneously. In [41], the authors present a framework, namely BLOCKBENCH, to analysis the private blockchain which includes consistency algorithms, data models, execution engines, and applications on the chain. EtherQL [42] proposes a flexible efficient approach for retrieving blockchain, which includes range queries and Top-k queries. In Beihang chain [43], the authors summarize the application development method based on blockchain and give the key problems to resolve for developing blockchain applications. ForkBase [44] designs a storage engine to provide efficient support for blockchain and forkable applications. Zhang *et al.* [45] focus on the problem of authenticated range queries in blockchain database. To scale up the query services, they present a gas-efficient structure and design an authenticated data structure (ADS) that can be efficiently maintained by the blockchain. Moreover, the authors also present vChain [46] to take the first step toward investigating the problem of verifiable query processing over blockchain databases. In order to support fully semantics, SEBDB [47] leverages the existing technologies of databases which are optimized for decades to design a blockchain database. Specifically, SEBDB adds relational data semantics into blockchain platform and uses SQL-like language as the general interface, instead of code-level APIs. Dang *et al.* [48] introduce a scaling blockchain system via sharding technology. To achieve the design goals, they first enhance the basic Byzantine consensus protocols, and then design an effcient shard information protocol. Finally, they design a general distributed transaction protocol to ensure safety and liveness. Han *et al.* [49] design a crowdsourcing platform by integrating the significant features of blockchain to overcome the limitations in existing crowdsourcing, i.e., non-transparent incentive mechanism and isolated profiles of workers.

Different from above researches, our work focuses on the reliable outsourcing services within multi-CSPs. By introducing significant features of the blockchain, we address the reliability problem caused by over-reliance on third-party and improve the quality of outsourced services.

VIII. CONCLUSION

In this paper, we present a novel reliable schema for metadata management of outsourced replications stored by multi-CSPs in untrusted environment. Based on the remarkable characters of blockchain, we first design a novel framework constituting a set of VPs, and each of which maintains the metadata in a blockchain form. Then, we introduce a collaborative algorithm for storing and verifying metadata of replications. In the store phase, VPs construct metadata block by using given signatures and corresponding metadata and write metadata block into local metadata blockchain. During the verify phase, VPs retrieve local blockchain using signatures to obtain corresponding metadata. Experimental results demonstrate that our proposed schema can not only efficiently store and verify the metadata, but also provide high scalability. In terms of future work, we plan to provide the optimization strategy for the verification processes.

REFERENCES

[1] (2019). *Google*. [Online]. Available: http://www.google.com

[2] (2019). *Microsoft Azure*. [Online]. Available: http://www.microsoft.com/windowsazure

[3] (2019). *Amazon*. [Online]. Available: http://aws.amazon.com

[4] (2019). *Baidu*. [Online]. Available: http://www.baidu.com

[5] J. Mao, J. Cui, Y. Zhang, H. Ma, and J. Zhang, "Collaborative outsourced data integrity checking in multi-cloud environment," in *Proc. Int. Conf. Wireless Algorithms, Syst., Appl.* Cham, Switzerland: Springer, 2016, pp. 511–523.

[6] D. Pöhn, *Topology of Dynamic Metadata Exchange via a Trusted Third Party*. Bonn, Germany: Gesellschaft für Informatik (GI), 2015, pp. 101–113.

[7] D. Pöhn, "Risk management for dynamic metadata exchange via a trusted third party," in *Proc. 2nd Int. Conf. Inf. Syst. Secur. Privacy (ICISSP)*, Rome, Italy, Feb. 2016, pp. 227–234. doi: 10.5220/0005651702270234.

[8] Q. Lu, Y. Xiong, X. Gong, and W. Huang, "Secure collaborative outsourced data mining with multi-owner in cloud computing," in *Proc. IEEE 11th Int. Conf. Trust, Secur. Privacy Comput. Commun. (TrustCom)*, Jun. 2012, pp. 100–108.

[9] F. Wang, P. Liu, J. Pearson, F. Azar, and G. Madlmayr, "Experiment management with metadata-based integration for collaborative scientific research," in *Proc. 22nd Int. Conf. Data Eng. (ICDE)*, Apr. 2006, p. 96.

[10] S. Nakamoto. (2008). *Bitcoin: A Peer-to-Peer Electronic Cash System*. [Online]. Available: http://www.bitcoin.org/bitcoin.pdf

[11] G. Wood, "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum Project Yellow Paper*, vol. 151, pp. 1–32, Apr. 2014.

[12] T. T. A. Dinh, R. Liu, M. Zhang, G. Chen, B. C. Ooi, and J. Wang, "Untangling blockchain: A data processing view of blockchain systems," 2017, *arXiv:1708.05665*. [Online]. Available: https://arxiv.org/abs/1708.05665

[13] S. Cohen and A. Zohar, "Database perspectives on blockchains," 2018, *arXiv:1803.06015*. [Online]. Available: https://arxiv.org/abs/1803.06015

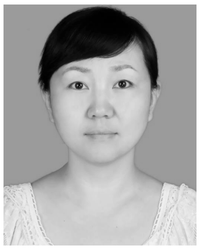
[14] Y. Yong and W. Feiyue, "The development status and prospects of blockchain technology," *Acta Automatica Sinica*, vol. 42, no. 4, pp. 481–494, 2016.

[15] J. Wang, X. Chen, J. Li, J. Zhao, and J. Shen, "Towards achieving flexible and verifiable search for outsourced database in cloud computing," *Future Gener. Comput. Syst.*, vol. 67, pp. 266–275, Feb. 2017.
[16] (2019). *Apache Hadoop*. [Online]. Available: http://hadoop.apache.org/
[17] S. A. Weil, S. A. Brandt, E. L. Miller, D. D. E. Long, and C. Maltzahn, "Ceph: A scalable, high-performance distributed file system," in *Proc. 7th Symp. Oper. Syst. Design Implement.* Berkeley, CA, USA: USENIX Association, 2006, pp. 307–320.
[18] J. Mao, Y. Zhang, P. Li, T. Li, Q. Wu, and J. Liu, "A position-aware Merkle tree for dynamic cloud data integrity verification," *Soft Comput.*, vol. 21, no. 8, pp. 2151–2164, 2017.
[19] *A Next-Generation Smart Contract and Decentralized Application Platform*. [Online]. Available: https://github.com/ethereumlwiki/wiki/WhitePaper
[20] J. H. Park and J. H. Park, "Blockchain security in cloud computing: Use cases, challenges, and solutions," *Symmetry*, vol. 9, no. 8, p. 164, 2017.
[21] A. Dorri, S. S. Kanhere, and R. Jurdak, "Blockchain in Internet of Things: Challenges and solutions," 2015, *arXiv:1608.05187*. [Online]. Available: https://arxiv.org/abs/1608.05187
[22] H. T. Vo, A. Kundu, and M. K. Mohania, "Research directions in blockchain data management and analytics," in *Proc. EDBT*, 2018, pp. 445–448.
[23] K. Christidis and M. Devetsikiotis, "Blockchains and smart contracts for the Internet of Things," *IEEE Access*, vol. 4, pp. 2292–2303, 2016.
[24] B. Lee and J.-H. Lee, "Blockchain-based secure firmware update for embedded devices in an Internet of Things environment," *J. Supercomput.*, vol. 73, no. 3, pp. 1152–1167, Mar. 2017.
[25] M. Ali, J. Nelson, R. Shea, and M. J. Freedman, "Blockstack: A global naming and storage system secured by blockchains," in *Proc. USENIX Conf. Usenix Annu. Tech. Conf.*, 2016, pp. 181–194.
[26] J. Wang, L. Gao, A. Dong, S. Y. Guo, H. Chen, and X. Wei, "Block chain based data security sharing network architecture research," *J. Comput. Res. Devices*, vol. 54, no. 4, pp. 742–749, 2017.
[27] A. Miller, A. Juels, E. Shi, B. Parno, and J. Katz, "Permacoin: Repurposing bitcoin work for data preservation," in *Proc. IEEE Symp. Secur. Privacy*, May 2014, pp. 475–490.
[28] (2014). *Filecoin: A Cryptocurrency Operated File Storage Network*. [Online]. Available: http://filecoin.io/filecoin.pdf
[29] B. Sengupta, S. Bag, S. Ruj, and K. Sakurai, "Retricoin: Bitcoin based on compact proofs of retrievability," in *Proc. Int. Conf. Distrib. Comput. Netw.*, 2016, p. 14.
[30] A. Juels and B. S. Kaliski, Jr., "Pors: Proofs of retrievability for large files," in *Proc. ACM Conf. Comput. Commun. Secur.*, 2007, pp. 584–597.
[31] H. Shacham and B. Waters, "Compact proofs of retrievability," *J. Cryptol.*, vol. 26, no. 3, pp. 442–483, Jul. 2013.
[32] S. Lowry and J. Wilkinson. (2018). *MetaDisk: Blockchain-Based Decentralized File Storage Application*. [Online]. Available: https://storj.io/metadisk.pdf
[33] S. Wilkinson and T. Boshexski. (2018). *MetaDisk: Blockchain-Based Decentralized File Storage Application*. [Online]. Available: https://storj.io/storj.pdf
[34] G. Zyskind, O. Nathan, and A. S. Pentland, "Decentralizing privacy: Using blockchain to protect personal data," in *Proc. IEEE Secur. Privacy Workshops*, May 2015, pp. 180–184.
[35] E. García-Barriocanal, S. Sánchez-Alonso, and M.-A. Sicilia, "Deploying metadata on blockchain technologies," in *Proc. Res. Conf. Metadata Semantics Res.* Cham, Switzerland: Springer, 2017, pp. 38–49.
[36] M. El-Hindi, M. Heyden, C. Binnig, R. Ramamurthy, A. Arasu, and D. Kossmann, "BlockchainDB—Towards a shared database on blockchains," in *Proc. Int. Conf. Manage. Data*, 2019, pp. 1905–1908.
[37] C. Mohan, "State of public and private blockchains: Myths and reality," in *Proc. Int. Conf. Manage. Data*, 2019, pp. 404–411.
[38] S. Maiyya, V. Zakhary, M. J. Amiri, D. Agrawal, and A. El Abbadi, "Database and distributed computing foundations of blockchains," in *Proc. Int. Conf. Manage. Data*, 2019, pp. 2036–2041.
[39] A. Sharma, F. M. Schuhknecht, D. Agrawal, and J. Dittrich, "Blurring the lines between blockchains and database systems: The case of hyperledger fabric," in *Proc. Int. Conf. Manage. Data*, 2019, pp. 105–122.
[40] T. McConaghy and R. Marques. (Jan. 2019). *BigchainDB: A Scalable Blockchain Database*. [Online]. Available: https://www.bigchaindb.com/whitepaper/bigchaindb-whitepaper.pdf
[41] T. T. A. Dinh, J. Wang, G. Chen, R. Liu, B. C. Ooi, and K.-L. Tan, "BLOCKBENCH: A framework for analyzing private blockchains," in *Proc. ACM Int. Conf. Manage. Data*, 2017, pp. 1085–1100.
[42] Y. Li, K. Zheng, Y. Yan, Q. Liu, and X. Zhou, "EtherQL: A query layer for blockchain system," in *Proc. Int. Conf. Database Syst. Adv. Appl.* Cham, Switzerland: Springer, 2017, pp. 556–567.
[43] W. T. Tsai, L. Yu, R. Wang, N. Liu, and E. Y. Deng, "Blockchain application development techniques," *J. Softw.*, vol. 28, no. 6, pp. 1474–1487, 2017.
[44] S. Wang, T. T. A. Dinh, Q. Lin, Z. Xie, M. Zhang, Q. Cai, G. Chen, W. Fu, B. C. Ooi, and P. Ruan, "ForkBase: An efficient storage engine for blockchain and forkable applications," 2018, *arXiv:1802.04949*. [Online]. Available: https://arxiv.org/abs/1802.04949
[45] C. Zhang, C. Xu, J. Xu, Y. Tang, and B. Choi, " GEM^2-tree: A gas-efficient structure for authenticated range queries in blockchain," in *Proc. IEEE 35th Int. Conf. Data Eng. (ICDE)*, Apr. 2019, pp. 842–853.
[46] C. Xu, C. Zhang, and J. Xu, "vChain: Enabling verifiable Boolean range queries over blockchain databases," in *Proc. Int. Conf. Manage. Data*, 2019, pp. 141–158.
[47] Y. Zhu, Z. Zhang, C. Jin, A. Zhou, and Y. Yan, "SEBDB: Semantics empowered blockchain DataBase," in *Proc. IEEE 35th Int. Conf. Data Eng. (ICDE)*, Apr. 2019, pp. 1820–1831.
[48] H. Dang, T. T. A. Dinh, D. Loghin, E.-C. Chang, Q. Lin, and B. C. Ooi, "Towards scaling blockchain systems via sharding," in *Proc. Int. Conf. Manage. Data*, 2019, pp. 123–140.
[49] S. Han, Z. Xu, Y. Zeng, and L. Chen, "Fluid: A blockchain based framework for crowdsourcing," in *Proc. Int. Conf. Manage. Data*, 2019, pp. 1921–1924.

KUN HAO is currently pursuing the Ph.D. degree with the College of Computer Science and Engineering, Northeastern University, China. His research interests include blockchain database, big data management, and outsourced data management.

JUNCHANG XIN received the B.Sc., M.Sc., and Ph.D. degrees in computer science and technology from Northeastern University, China, in 2002, 2005, and 2008, respectively. He visited the National University of Singapore as a Postdoctoral Visitor from 2010 to 2011. He is currently an Associate Professor with the School of Computer Science and Engineering, Northeastern University, China. He has published more than 60 research articles. He served as PIs or Co-PIs for more than ten national research grants from NSFC, the 863 Program, the Project 908 under the State Oceanic Administration, and so on. His research interests include big data, uncertain data, bioinformatics, and blockchain database.

ZHIQIONG WANG received the M.Sc. degree in computer applications technology and the Ph.D. degree in computer software and theory from Northeastern University, China, in 2008 and 2014, respectively. She visited the National University of Singapore, in 2010, and The Chinese University of Hong Kong, in 2013, as the Academic Visitor. She is currently an Associate Professor with the College of Medicine and Biological Information Engineering, Northeastern University. She has published more than 30 articles. She served as PIs or Co-PIs for more than ten national research grants from NSFC, the Natural Science Foundation of Liaoning Province, and so on. Her current research interests include biomedical, biological data processing, cloud computing, and machine learning.



KEYAN CAO is currently an Associate Professor with Shenyang Jianzhu University. Her research interests include big data management, query processing and optimization, and probabilistic database. She is a member of CCF.

GUOREN WANG received the B.Sc., M.Sc., and Ph.D. degrees from the Department of Computer Science, Northeastern University, China, in 1988, 1991, and 1996, respectively. He is currently a Professor with the Department of Computer Science, Beijing Institute of Technology, China. He has published more than 100 research articles. His research interests include XML data management, query processing and optimization, bioinformatics, high-dimensional indexing, parallel database systems, and P2P data management.