IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

SPECIAL SECTION ON INNOVATION AND APPLICATION OF INTELLIGENT PROCESSING OF DATA, INFORMATION AND KNOWLEDGE AS RESOURCES IN EDGE COMPUTING

Received January 6, 2020, accepted January 12, 2020, date of publication January 15, 2020, date of current version January 28, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2966803

Research on Integrity Check Method of Cloud Storage Multi-Copy Data Based on Multi-Agent

CHUNBO WANG[1,2,3] **AND XIAOQIANG DI**[1,3,4]

[1]School of Computer Science and Technology, Changchun University of Science and Technology, Changchun 130022, China
[2]Changchun Automobile Industry Institute, Changchun 130000, China
[3]Jilin Province Key Laboratory of Network and Information Security, Changchun 130022, China
[4]Information Center, Changchun University of Science and Technology, Changchun 130022, China

Corresponding author: Xiaoqiang Di (dixiaoqiang@cust.edu.cn)

This work was supported in part by the research grants from Science and Technology Project of Jilin Province under Grant 20190302070GX.

ABSTRACT Aimed at the huge number of files in cloud storage and the low audit efficiency of single-node audit mode, this paper proposes a multi-agent-based cloud storage multi-copy data integrity check scheme. The scheme uses a bilinear mapping method to construct the key generation process and a multi-branch authentication tree to perform multi-copy data signature to implement multi-copy authentication, signature, and verification. It also uses a directed acyclic graph (DAG) to represent the task relationship in the workflow, task and resource allocation based on QoS demand preference settings to schedule multiple tasks. Experimental results show that the proposed scheme has obvious advantages over existing schemes in terms of communication, storage overhead, and audit efficiency, with an average audit efficiency improvement of 20%.

INDEX TERMS Integrity check, multi-copy data, cloud storage, multi-agent.

I. INTRODUCTION

With the increasing development of cloud computing technology, public cloud storage services have been widely used, and the number of users of public cloud storage products such as DropBox, Google Drive, Kingsoft Express and so on has grown rapidly [1], [2]. However, cloud storage services have many risks such as data damage and data loss incidents, which has increased users' concerns about cloud data integrity. Discovering data corruption in time can not only dispel users' doubts, but also earn valuable time for data recovery, making cloud storage data integrity verification an urgent fundamental issue. The Provable Data Possession (PDP) [3], [4] is an important method for verifying the integrity of data stored in the cloud. This method uses a sampling strategy to initiate integrity verification of files in the cloud, and can timely identify the behavior of corrupting data in the cloud without downloading the entire file. However, the number of files in cloud storage is huge, and data integrity verification faces a heavy audit burden. An unreasonable audit solution will seriously affect the audit efficiency of the system. Therefore, how to effectively reduce the audit cost of data integrity verification of large-scale files and improve the audit efficiency of the system is an important issue that needs to be solved urgently.

In recent years, researchers have proposed various solutions to the problem of data integrity verification [3]–[20]. Juels and Kaliski Jr., [5] proposed a retrievable data proof model (Proofs of retrievability, POR), which required "sentinels" to return to a specified location to check for file corruption. However, the number of "sentinels" was fixed and the model only supported a limited number of audits, and rearranging "sentinels" are costly. Ateniese *et al.* [6] proposed a data-holding proof model that uses random sampling to verify documents, so that the number of audits is unlimited. And the PDP separates authentication information from the original data, maintaining the independence of the original files. PDP has become the main method to verify the integrity of cloud storage data. Researchers have conducted in-depth research on the issues of dynamic data update, public auditing, and collaborative storage of the PDP method [7]. Erway *et al.* [8] used hierarchical data skip tables to realize a PDP model that supports dynamic data update, but the intermediate nodes are complicated to calculate Wang *et al.* [9] used the merkle hash tree structure to simplify the calculation process of intermediate nodes. Zhu *et al.* [10] and Yang and

The associate editor coordinating the review of this manuscript and approving it for publication was Ying Li.

Jia [15] used a two-dimensional index-hash table structure to dynamically update data, which is suitable for occasions with fewer update requests. Zhu *et al.* [16] pointed out the problems of forgery attack and replay attack when data is updated and enhanced the security of the system by improving the audit protocol. In order to reduce the user's computational burden, the agent-based audit model [11], [16], [17] entrusted the file audit task to a third party for execution, and realized the privacy protection of the authentication process using random hidden code technology. In addition, Li *et al.* [18], [19] proposed a PDP scheme in a multi-cloud collaborative storage environment. Vijayakumar and Arun [20] proposed a PDP scheme that supports file sharing, and realized the granting and recycling of low-cost user sharing permissions. Zhou *et al.* [21] and Zhu *et al.* [22] proposed a cloud storage-oriented multi-copy data integrity auditing scheme.

For large-scale file audits, batch verification technology is mainly used to improve the audit efficiency of the system. Ateniese *et al.* [23] proposed a PDP data integrity verification model based on homomorphic authentication tags. The homomorphism of tags was used to merge multiple data block verification processes, which improved the audit efficiency of multiple data blocks in a file. Ateniese *et al.* [6] and Gao *et al.* [24] provided a dynamic version of the PDP, which enabled it to support data update operations and expanded the application scope of the PDP scheme. Jouini and Rabai [25] proposed the multi-file batch auditing scheme by using the nature of bilinear pairings. The data integrity verification of multiple files can be completed through one audit, which improves the efficiency of multi-file audits.

The above solutions provide feasible ideas for solving the problem of data integrity verification in cloud storage services, and lay a good foundation for further research. However, the existing scheme mainly considers how to efficiently handle the submitted document audit tasks, and does not consider the issue of organization and coordination of audit tasks of large-scale documents. That is, it does not consider how to arrange the order and intensity of document audits. Therefore, this paper proposes a multi-agent based cloud storage multi-copy data integrity check scheme.

II. MULTI-AGENT VERIFICATION MODEL FOR CLOUD STORAGE BASED ON MULTI-AGENT

A. MODEL-RELATED DEFINITIONS

Multi-agent based data integrity audit model consists of 8 parts: key generation algorithm, copy generation, user-generated authentication tags, audit scheme generation, audit task scheduling, challenge information generation, data holding evidence generation, and data holding verification.

(1) $KeyGen(1^{\lambda}) \rightarrow (pk, sk)$*:* This algorithm is a key generation algorithm, which is run by the client and generates public key pk and private key sk.

(2) $ReplicaGen(F) \rightarrow (F')$*:* This algorithm is a copy generation algorithm. The client processes the data and generates a specific copy of the data. The input of the algorithm is the original data file F, and the output is a set of duplicate data block F' after encryption, blocking and randomization.

(3) $SigGen\left(sk, F'\right) \rightarrow (\sigma, T)$*:* This algorithm is the root node and data block signature generation algorithm of the multi-branch authentication tree, which is run by the client. The algorithm inputs the user's private key sk and the data block set F, and the algorithm output results are the data block signature value σ and the multi-branch authentication tree root node signature value T. Finally, the client sends the data block F and the signature value (σ, T) of the root node of the authentication tree to the CSP to store.

(4) $AuditPlanGen\,(FA) \rightarrow (QA)$: This algorithm is an audit scheme generation method, which is run by a third-party auditor TPA. The input of the algorithm is a list of file attributes FA, and the output is a list of file audit schemes QA. Among them, the file attribute FA is composed of a ternary sequence pair $< d, h, \gamma >$, d is the probability of damage to the file, h is the access heat of the file, and γ is the time decay factor.

(5) $AuiTaskScheduling\,(T) \rightarrow (A)$: This algorithm is an audit task scheduling method and is run by a third-party auditor TPA. The input of the algorithm is the audit task set $T = \{QA_i | i \in W\}$ of the file, and the output list of audit tasks is $A = \{A_i | i \in N\}$. Among them, QA_i is the audit task of the file, W is the total number of tasks, A_i is the audit task queue corresponding to the audit agent, and N is the total number of agents.

(6) $GenChallenge\,(I, Q) \rightarrow chal$: This algorithm generates a challenge information generation algorithm for the audit agent, which is initiated by TPA and generates challenge information. The algorithm inputs the challenge data block set T and the random number pair $Q = (i, v_i)$, and the algorithm outputs the challenge information. Finally, the TPA sends the challenge message to the CSP primary storage server.

(7) $Gen\ Proof\ \left(F', \sigma, chal\right) \rightarrow P$: This algorithm is an evidence generation algorithm, which is run by the CSP. The CSP uses the challenge information sent by the TPA to generate corresponding challenge evidence to prove the integrity of the data block. The input of the algorithm are the data set F', the data block signature value σ and the challenge information $chal$; the output of the algorithm is the challenge evidence P. CSP obtains challenge evidence by processing homogeneous aggregation technology on the data set F' and signature value σ, and CSP returns the aggregated challenge evidence P to TPA.

(8) $Verify\ \mathrm{Pr}oof\ (P) \rightarrow \{True, False\}$: This algorithm is a verification algorithm for challenge evidence, which is run by TPA. The algorithm inputs challenge evidence P. If the challenge evidence is verified, it outputs the verification result $True$; otherwise, it outputs the verification result $False$.

B. MODEL AUDIT PROCESS

The audit process of the model is shown in Figure 1. The specific process is:

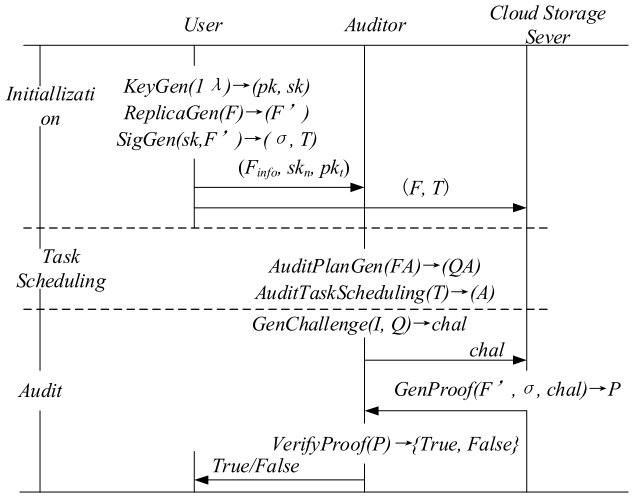

FIGURE 1. Model audit process.

(1) The user uses *KeyGen* to generate key pair (sk_t, pk_t) and encryption key sk_n, and uses *SinGen* to generate the data block authentication tag set T and file summary information $M_{inf\ o}$. The user sends $\left(M_{inf}, sk_n, pk_t\right)$ to the main audit agent and sends (M, T) to the cloud server.

(2) The main audit agent receives or organizes the file audit task, and uses *AlloTask* to complete audit task scheduling and organize the audit agent to complete the file audit; each audit agent uses *Chall* to generate challenge information C and sends it to the cloud server. The cloud server uses *GenProof* to generate data holding evidence P and returns it to the auditor. The audit agent uses *VerifyProof* to verify the received evidence information. If P passes the audit, it indicates that the file has been stored intact on the cloud server, and the local copy can be deleted.

(3) Perform the above steps cyclically to complete the periodic audit of the documents.

III. DATA INTEGRITY VERIFICATION PROCESS

In this paper, document [XX] is used for data signing. The data block signature and the signature value of the root node of the multi-branch authentication tree are generated by the client. Each node in the multi-branch authentication tree is authenticated by the corresponding parent node, and the signed data blocks are authenticated by the corresponding leaf nodes, which are the basis of evidence generation and data verification.

First, according to the constructed multi-copy and multi-branch authentication tree, retrieve and calculate the node value and the root node value of the challenge data block;

Second, generate the challenge request information and send it to the replica storage server. The replica storage server calculates the evidence of replica data holding and returns it to the CSP primary storage server to generate the signature of the challenge data block.

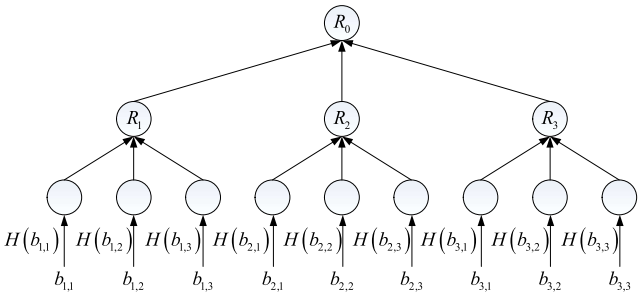

FIGURE 2. Multi-branched authentication tree replica evidence storage.

A. KEY GENERATION

The key generation algorithm is mainly the public and private keys generated by the client. Suppose that the group G_1 and group G_2 are cyclic groups with a prime order P, g_1 and g_2 are the generators of group G_1 and group G_2, bilinear mapping $e : G_1 \times G_1 \rightarrow G_2$, anti-collision *hash* function $H : M \rightarrow \{0, 1\}^s$, M is the message space. Randomly select $\alpha_0, \alpha_1, \alpha_2, \ldots, \alpha_n \leftarrow \mathbb{Z}_q$ and $H \in G_2$. Calculate $H_0 = H^{1/\alpha_0}$, $H_i \leftarrow H^{1/\alpha_I}$, $H_i \in G_2 H_i \in G_2$ and $i = \{1, 2, \ldots, n\}$. Get the public keys $pk = (H, H_0, H_1, \ldots, H_n)$ and private keys $sk = (\alpha_0, \alpha_1, \alpha_2, \ldots, \alpha_n)$.

B. COPY DATA GENERATION

The copy generation algorithm is to process the data by the client and generate a specific copy of the data. The specific process is as follows:

(1) The user uses a symmetric encryption algorithm (such as AES) to encrypt the file F to obtain the ciphertext data, and then chunks the ciphertext data in blocks to get $\tilde{F} = \{m_1, m_2, \ldots, m_n\}$.

(2) The user saves K copies of the block data $\tilde{F}$ and obtains $\tilde{F} = \left\{m_{k,1}, m_{k,2}, \ldots, m_{k,n}\right\}$ and $1 \leq k \leq K$.

(3) In order to further enhance security, use random masking technology to randomize the block data $\tilde{F}$ to obtain data $F' = (F_k)_{1 \leq k \leq K} = \left\{b_{k,i}\right\}_{1 \leq i \leq n, 1 \leq k \leq K} = \left\{b_{k,1}, b_{k,2}, \ldots, b_{k,n}\right\}$, where $b_{k,i} = m^s_{k,i} + r_s$, $m_{k,i} = \left\{m^1_{k,i}, m^2_{k,i}, \ldots, m^{a/b}_{k,i}\right\}$, $r_s = f\left(H_i || k || s\right)$, $1 \leq i, s \leq n$, $1 \leq k \leq k$ is generated by a pseudo random function.

C. EVIDENCE GENERATION

The model uses a multi-branch authentication tree to construct multiple copies of evidence, as shown in Figure 2.

The specific generation process is:

The CSP primary storage server to generate the signature of the challenge data block.

Finally, the evidence for the challenge response is generated by the aggregation of the CSP primary storage server. The specific process is as follows:

(1) After receiving the challenge message *chal*, the CSP primary storage server queries the leaf nodes of the corresponding data block of $chal = \{(i, v_i)\}_{s_1 \leq i \leq s_e}$ according to the constructed multi-copy and multi-branch authentication tree, records the authentication auxiliary

information $\{\Omega_i\}_{s_1 \le i \le s_e}$ of these leaf nodes, and calculates the corresponding first replica data node value R'_1 and root node value $\tilde{R}_0$ in the challenge information.

(2) The primary storage server generates new challenge request messages $chal_k = \{(k, i, v_i)\}_{s_1 \le i \le s_e, 2 \le k \le K}$ and sends them to the corresponding $K-1$ replica storage servers respectively.

(3) The k-th copy storage server calculates the corresponding node value R'_k and the holding evidence μ'_k and δ'_k of the challenge data block according to the challenge request message $chal_k = \{(k, i, v_i)\}$. The specific calculation results are as follows:

$$R'_k = H\left(H\left(R_{k,s_1}\right) || H\left(R_{k,s_2}\right) || \ldots || H\left(R_{k,s_e}\right)\right) \quad (1)$$

$$\mu'_k = \sum_{i=s_i}^{s_e} v_i \cdot b_{k,i} \quad (2)$$

$$\delta'_k = \prod_{i=s_i}^{s_e} \sigma_{k,i}^{v_i} \quad (3)$$

(4) Each replica storage server sends the holding evidence $\left\{R'_k, \mu'_k, \delta'_k\right\}_{2 \le k \le K}$ to the primary storage server. After the primary storage server receives the holding evidence, it uses $\left\{R'_k\right\}_{2 \le k \le K}$ to verify whether equation (4) holds or not:

$$\tilde{R}_0 = H\left(H\left(R'_1\right) || H\left(R'_2\right) || \ldots || H\left(R'_K\right)\right) \quad (4)$$

If equation (4) holds, the primary storage server calculates:

$$\mu_k = \mu_1 + \mu'_k = \sum_{i=s_i}^{s_e} v_i \cdot b_{1,i} + \mu'_k \quad (5)$$

$$\delta_k = \delta'_1 \cdot \delta'_k = \left(\prod_{i=s_i}^{s_e} \sigma_{1,i}^{v_i}\right) \cdot \delta'_k \quad (6)$$

If equation (4) does not hold, the primary storage server calculates $\left\{R''_k\right\}_{1 \le k \le K}$ and retrieves and locates the location of the incorrect replica data node value by comparing the values of R''_k and R'_k.

(5) Finally, the primary storage server calculates the holding verification evidence:

$$\mu = \sum_{k=1}^{K} \mu_k \quad (7)$$

$$\delta = \prod_{k=1}^{K} \delta_k \quad (8)$$

The primary storage server sends the generated evidence $P = \{R_0, \mu, \delta\}$ to the TPA.

D. DATA VERIFICATION

The challenge evidence verification algorithm is run by the TPA. After the TPA receives the challenge evidence P provided by the CSP, it performs the following operations:

The TPA uses R_0 to verify the validity of the data blocks and whether equation (9) holds or not:

$$e\left(T, H()\left(H\left(R_0\right), H_0\right)\right) \quad (9)$$

If the above equation holds, TPA uses μ and δ to verify whether the challenge data block is held correctly or not by equation (10):

$$e\left(\delta, H\right) = \prod_{k=1}^{K} e\left(\prod_{i=s_1^s}^{s_e} \mathscr{H}\left(b_{k,i}\right)^{v_i} \cdot H^{\mu}, H_k\right) \quad (10)$$

If the equation holds, output *True*, otherwise, *False*.

TABLE 1. Audit plan level mapping table.

Audit cycle level	Audit cycle / h	Error recognition rate level	Error recognition rate /%
1	72	1	90
2	24	2	95
3	8	3	99
4	4		
5	2		

IV. AUDIT PLAN GENERATION

The audit scheme consists of the audit cycle and recognition rate. This article sets the audit cycle level LT and recognition rate level LRR according to the service level agreement. Set the total number of levels in the audit cycle as LT_{max}, and the total number of recognition rate levels LRR_{max}.

When establishing the file audit cycle level LT_k, first calculate the audit cycle level LT_{d_k} established by the file corruption probability and the audit cycle level LT_{h_k} established by the file access heat according to the file attribute value $\langle d_k, h_k, \gamma \rangle$, and then take the minimum of the two, and calculate as follows:

$$LT_k = min\left\{LT_{d_k}, LT_{h_k}\right\} \quad (11)$$

$$T_{d_k} = LT_{max} \cdot \frac{d_k^{\gamma_k}}{d_{max}} \quad (12)$$

$$LT_{h_k} = LT_{max} \cdot \frac{h_k^{\gamma_k}}{h_{max}} \quad (13)$$

When establishing the file recognition rate level LRR_k, first calculate the recognition rate level LRR_{d_k} established by the file corruption probability and the recognition rate level LRR_{h_k} established by the file access heat according to the file attribute value $\langle d_k, h_k, \gamma \rangle$, and then take the maximum of the two, and calculate as follows:

$$\begin{aligned} LRR_k &= max\left\{LRR_{d_k}, LRR_{h_k}\right\} \\ &= max\left\{\left[LRR_{max} \cdot \frac{d_k^{\gamma^k}}{d_{max}}\right], \left[LRR_{max} \cdot \frac{h_k^{\gamma_k}}{h_{max}}\right]\right\} \end{aligned} \quad (14)$$

After obtaining the audit cycle level LT_k and recognition rate level R_k, the audit cycle t_k and recognition rate r_k of the file can be obtained from Table 1.

V. QOS-BASED AUDIT TASK SCHEDULING

A. SCHEDULING MODEL

Considering the constraint nature between tasks, it was decided to use a directed acyclic graph (DAG) to represent the task relationships in the workflow.

This paper uses $G = \{T, E, W, C\}$ to represent quads whose nodes and edges have weights. Among them, $T = \{t_i\}$ represents the set of n task nodes, $E = \left\{e_{(i,j)}\right\}$ is the set of e edges between tasks. Each edge $e(i,j) \in E$ represents the interdependence between tasks, $W = \left\{w_{(i,j)} \middle| w_{(i,j)}\right\} t_i r_j$ represents the execution cost of the task t_i on the audit agent

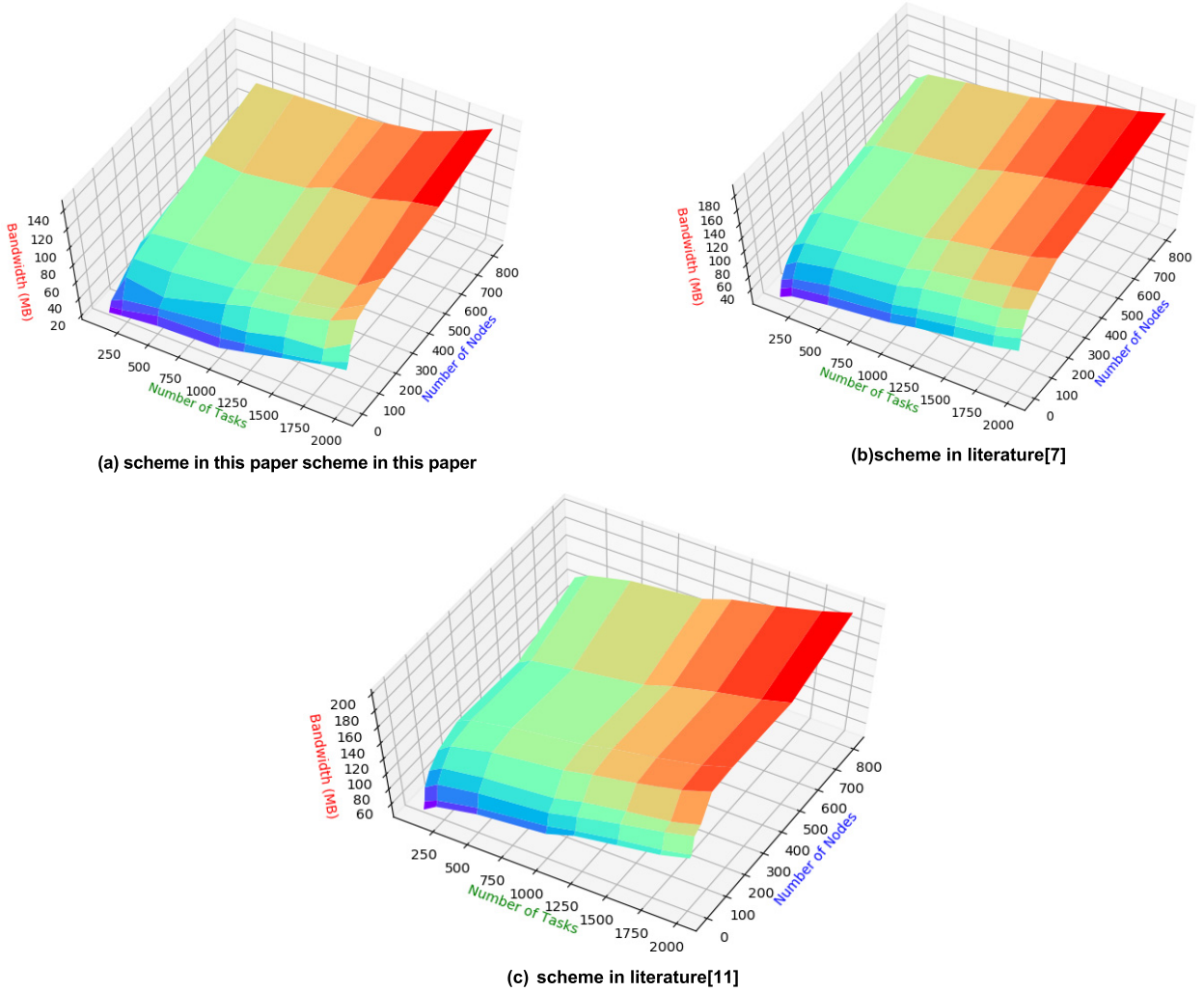

FIGURE 3. Comparison of communication overhead.

r_j, and $C = \{c_{ij}\}$ $t_i t_j$ represents the communication overhead between task t_i and task t_j.

For each task t_i, resources with many different QoS attributes are provided. Audit agents $R = \{r_1, r_2, \ldots, r_m\}$ whose number is m can provide these instance information and generate scheduling plans. Each agent r_i has different computing power and unit price, which can be expressed as $r_i = (Msr_i, Psr_i)$. Msr_i refers to the audit speed of the audit agent. Psr_i is the rental price within the audit unit time.

The earliest execution time EST of task t_i refers to the earliest start time of task t_i, which can be recursively obtained by traversing the nodes when all the predecessor nodes of the task have been executed and have been transmitted to the resource:

$$EST\ (t_i) = \begin{cases} 0 & Pred(t_i) = f \\ \max\limits_{t_j \in pred(t_i)} (c_{(j,i)} + EST(t_j) + \overline{w_j}) \end{cases} \tag{15}$$

Similarly, the latest execution time of task t_i means that when the earliest execution time of the exit task is unchanged, that is, without affecting the completion time *makespan* of the task, the latest execution time of the task t_i can also be recursively traversed each node to get LST:

$$LST(t_i) = \begin{cases} EST(t_i)\ Succ(t_i) = f \\ \min\limits_{t_k \in succ(t_i)} (LST(t_k) - c_{(k,i)}) - \overline{w_i} \end{cases} \tag{16}$$

The key factor of the task is EL, which can be obtained by the difference between EST and LST. It can be known that the smaller the EL value, the more urgent the task, and the higher the criticality:

$$EL\ (t_i) = LST\ (t_i) - EST\ (t_i) \tag{17}$$

B. QOS DEMAND PREFERENCE SETTING AND RESOURCE ALLOCATION

The earliest execution time $ST\ (t_i, r_j)$ of task t_i on the audit agent r_j can be calculated as follows:

$$ST\ (t_i, r_j) = max\ (AFT\ (t_k) + c_{(k,j)})\ avail\ (r_j)\ max_{t_j \in pred(t_i)} \tag{18}$$

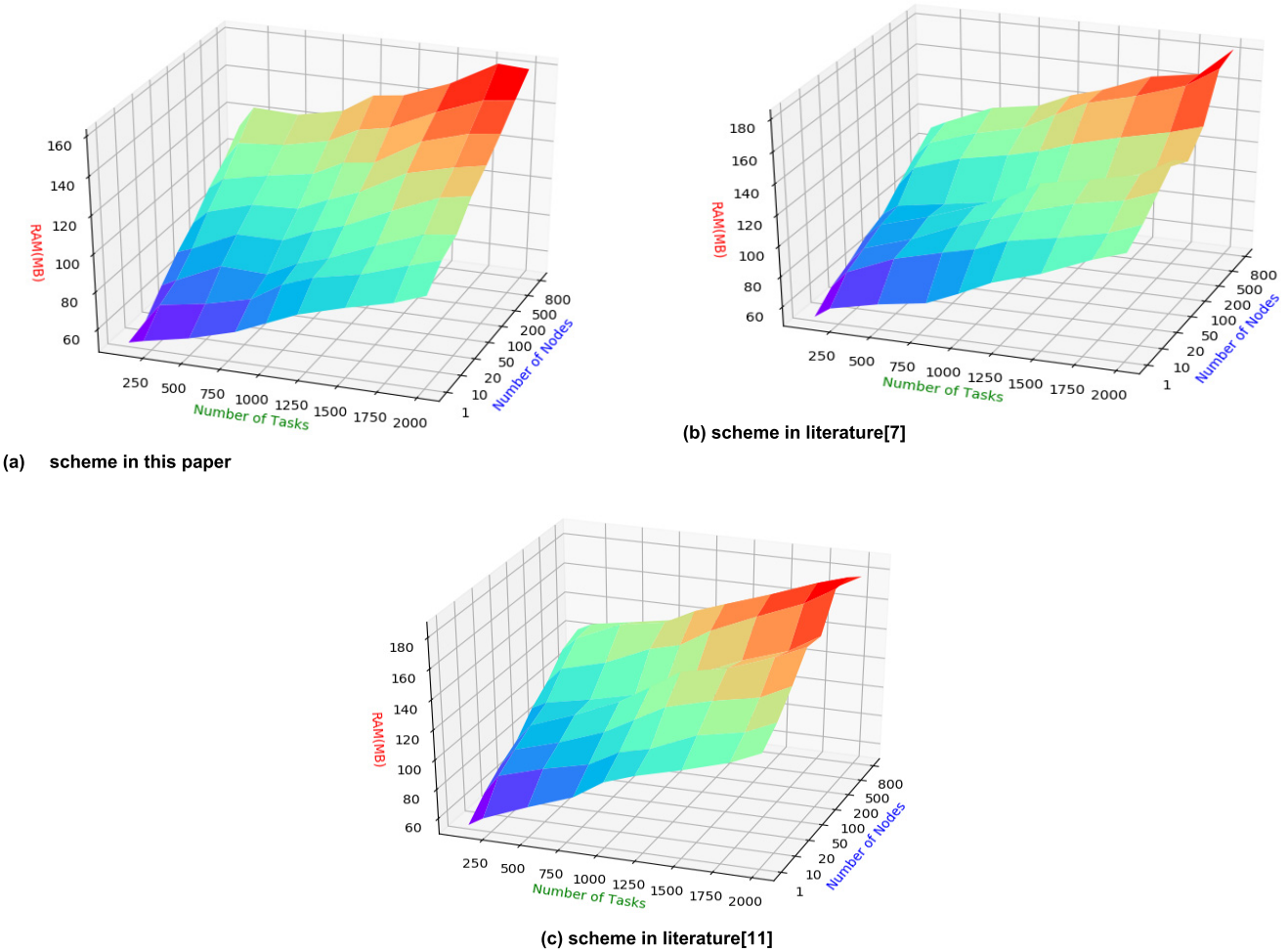

FIGURE 4. Comparison of storage overhead.

$avail\left(r_j\right)$ indicates the earliest time that the audit agent r_j can start execution. Similarly, the earliest completion time $FT\left(t_i, r_j\right)$ of task t_i on the audit agent r_j can be calculated by:

$$FT\left(t_i, r_j\right) = w_{(t_i,r_j)} + ST\left(t_i, r_j\right) \tag{19}$$

This article uses the two most common QoS indicators, which are execution time and execution cost, respectively. Tr_i and Cr_i are used to represent the time performance index and cost index of the audit agent r_i.

1) TIME PREFERENCE SETTINGS

In the task scheduling process, in order to meet the user's execution time requirements, we use the following formula:

$$Gtime = Tr_{(t_{exit}, r_{exit})} \leq Time \tag{20}$$

$Gtime$ represents the actual completion time of all task maps. $Tr(t_{exit}, r_{exit})$ represents the completion time of the export task t_{exit} on the audit agent r_{exit} assigned to it, and Time represents the completion time preset by the user for the entire task map.

During the task scheduling process, we hope to be able to select the audit agent assignment that minimizes the completion time and the completion time $Tr_{(t_i,r_j)}$ of the task t_i on the audit agent r_j under constraints, and $FT\left(t_i, r_j\right) = Tr_{(t_i,r_j)}$. Therefore, the time factor BT_{ij} of the task t_i on the audit agent r_j is defined as follows:

$$BT_{i_j} = \frac{makespan^i_{max} - Tr_{(t_i,r_j)} + 1}{makespan^i_{max} - makespan^i_{min} + 1} \tag{21}$$

Among them, $makespan^i_{max}$ refers to the actual maximum completion time of the task t_i on all audit agents, $makespan^i_{min}$ refers to the actual minimum completion time of the task t_i on all audit agents, and $Tr_{(t_i,r_j)}$ refers to the actual completion time of the task on audit agent r_j. Due to the constraint of equation (20), $0 \leq BT_{i_j} \leq 10 \leq BT_{i_j} \leq 1$, so the larger BT_{i_j} is, the less completion time of task t_i on all audit agents.

2) COST PREFERENCE SETTING

In addition to time constraints, general users will also have cost requirements. We use the following formula to express:

$$G\,cos\,t = \sum_{i=1}^{n} Cr\left(t_i, r_j\right) \leq Cos\,t \tag{22}$$

$Gtime$ represents the total cost of the audit agent after the task is executed, and $Cr(t_i, r_j) = wc_{(i,j)}$ represents the execution

cost of the export task t_i on the audit agent r_j assigned to it, while *Cost* represents the cost budget specified by the user for the entire task map.

This article uses BC_{ij} to represent the cost factor of service audit agent, which reflects the execution cost of the task t_i on the audit agent r_j. The expression is:

$$BC_{ij} = \frac{cost^i_{max} - Cr\left(t_i, r_j\right) + 1}{cost^i_{max} - cost^i_{min} + 1} \tag{23}$$

Among them, $cost^i_{max}$ represents the maximum execution cost of the task t_i on the audit agent, and $cost^i_{min}$ represents the minimum execution cost of the task t_i on the audit agent. Because it is constrained by equation (22), $0 \leq BC_{i_j} \leq 1$, it can be seen from the above that the larger BC_{i_j} is, the smaller the execution cost of the task t_i on the audit agent.

TABLE 2. BSTD scheduling algorithm.

Algorithm BSTD scheduling algorithm
Input: *DAG* graph *G*
Output: scheduling *plan*
1 Calculate the average execution time $\overline{w_i}$ of each task node in graph G and the communication time $c_{(i\ ,\ j)}$ between tasks
2 Calculate tasks EST, LST, and key factors EL according to equations (15), (16), and (17)
3 Form a task scheduling queue based on LST and EL
4 *while* task scheduling queue is not empty *do*
5 Select the first task t_i in the queue
6 *For* processor r_j in the resource list *do*
7 Calculate the completion time of the task t_i on the resource r_j, and then rank all the predecessor tasks of the task t_i according to the communication time to form a predecessor task list
8 *For* predecessor task list is not empty *do*
9 Pick the first task t_j in the list. If copying t_j on the resource r_j can reduce the completion time of task t_j on the resource r_j, then copy the task t_j on the resource r_j.
10 break
11 Remove t_j from the list if the completion time of task t_i cannot be reduced
12 End for
13 Calculate $Tr(t_i\ ,\ r_i)$ and $Cr(t_i\ ,\ r_i)$ of task t_i on resource r_j separately
14 Calculate BT_{ij} and BC_{ij} of task t_i on resource r_j according to equations (21) and (23)
15 Calculate the comprehensive evaluation factor θ_{ij} of task t_i on resource r_j according to formula (24)
16 End for
17 Send the task t_i to the resource r_j with a large comprehensive evaluation factor θ_{ij}, and delete the task t_i from the schedule list. If task t_i is not on the same resource as the copied predecessor task, delete the redundant copy task.
18 End while

TABLE 3. Host configuration.

Parameter	configuration information
CPU	i7-4790K, 3.6GHz
Memory	16G DDR3 * 2
Network	1000Mbps
Harddisk	1TB+256GB_SSD
OS	CentOS 6.5

3) COMPREHENSIVE QOS PREFERENCE SETTING

Considering the execution of workflow based on time preference and cost preference, a comprehensive evaluation factor θ_{ij} is proposed, which is calculated as follows:

$$\theta_{ij} = \alpha \times BT_{ij} + \beta \times BC_{ij} \tag{24}$$

where α and β are the preference values of the user for the constraint conditions, α and β must be greater than 0, and $\alpha + \beta = 1$, we can infer the comprehensive evaluation of the task t_i on the audit agent r_j according to the above formula. The greater θ_{ij} is, the higher the comprehensive result of the task on the audit agent will be, then the task will be assigned to the resource when scheduling.

C. BSTD ALGORITHM SCHEDULING PROCESS

According to the relationship between the tasks in the DAG graph, firs prioritize the tasks and calculate the key factors of the tasks according to the formula to form the priority scheduling queue, and then assign the tasks to the optimal resources according to the QoS demand settings. In addition, the BSTD algorithm also considers the resource idle time and combines the task replication strategy. The specific scheduling process is shown in Table 2 below:

VI. EXPERIMENTAL SIMULATION

In order to verify the rationality of the model and method in this paper, this section mainly conducts experiments from the aspects of communication overhead, storage overhead, scheme matching degree, and audit efficiency, and compares and analyzes with the schemes proposed by the literature [7] and [11].

A. EXPERIMENTAL ENVIRONMENT

Based on the PBC library, this paper designs and develops a prototype system and a comparison system for the integrity check model, and uses C++ to develop the client, audit agent, and audit management. In the verification process in this article, the characteristics of sociology and business management are produced using the Pareto principle. That is, in each audit cycle, files with different degrees of heat are generated and the files are damaged randomly according to the Pareto principle. The host configuration used in the experiment is shown in Table III.

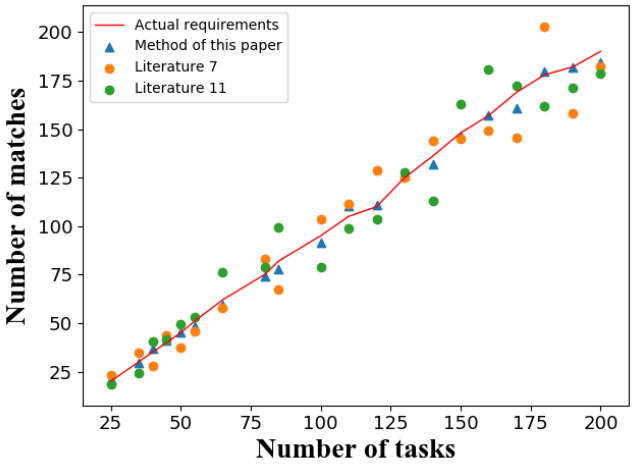

FIGURE 5. Matching degree of audit schemes.

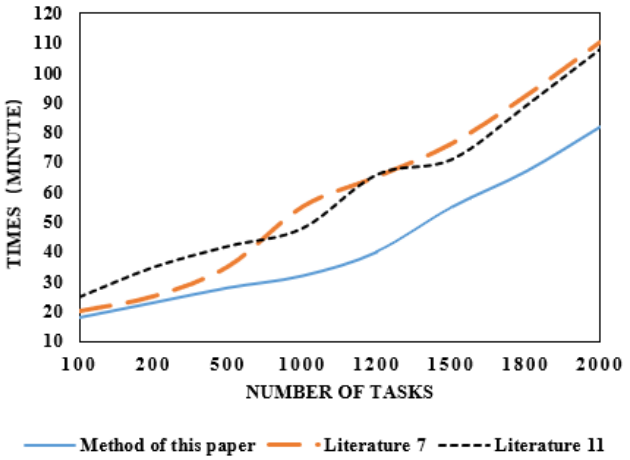

FIGURE 6. Audit efficiency.

B. EXPERIMENTAL ANALYSIS

1) COMMUNICATION AND STORAGE OVERHEAD

In terms of communication and storage costs, we mainly analyze and compare the communication costs caused by multi-node coordination. That is, communication consumption of the overall model when the number of audit agent nodes increases. In this experiment, we choose to set the number of proxy nodes as 5, 10, 20, 50, 1000, 500, 200, and 100 files to be audited, of which 20% of the files are set in the active state. Compare the bandwidth overhead of single nodes and multi-nodes under different number of tasks. The comparison is shown in Figure 3 and Figure 4:

It can be seen from Figures 3 and 4 that with the increase in the number of audit agent nodes, it does not significantly affect the communication and storage overhead of the audit model. This is mainly because the communication and storage overhead mainly comes from the ''challenge-response'' initiation and verification, as well as the consumption between the main audit agent and the audit agent. This consumption of messages based on the verification and management signals is less and can be ignored.

2) MATCHING DEGREE OF AUDIT PLAN

It can be seen from Figure 5 that the audit scheme generated by the method in this paper has a high degree of matching with the user's audit requirements and a small deviation from the actual. This is mainly because compared with the document [XX], when this article is constructing file attributes, the time decay factor γ was added, and the relationship between file storage time and file popularity was built to make the generated solution closer to the user's requirements.

3) AUDIT EFFICIENCY

It can be seen from Figure 6 that the auditing model in the multi-agent mode has significantly improved auditing efficiency compared with the single-node auditing method. This is mainly due to the multi-agent model changes the

VII. CONCLUSION

This paper proposes a multi-agent-based cloud storage multi-copy data integrity check scheme. The scheme uses a bilinear mapping method to construct the key generation process, and uses a multi-branch authentication tree to perform multi-copy data signing to implement the authentication, signature and verification of multi-copy. It also uses a directed acyclic graph (DAG) to represent the task relationship in the workflow, and allocates task and resource based on QoS demand preference setting to realize the scheduling of multiple tasks. Experimental results show that the proposed scheme has obvious advantages over existing schemes in terms of communication, storage overhead and audit efficiency, with an average audit efficiency improvement of 20%.

REFERENCES

[1] H. Wang, Z. Chen, J. Zhao, X. Di, and D. Liu, "A vulnerability assessment method in industrial Internet of Things based on attack graph and maximum flow," *IEEE Access*, vol. 6, pp. 8599–8609, 2018, doi: 10.1109/access.2018.2805690.

[2] Q. Wang, C. Wang, J. Li, K. Ren, and W. Lou, "Enabling public verifiability and data dynamics for storage security in cloud computing," in *Proc. Eur. Symp. Res. Comput. Secur.*, Saint-Malo, France: Springer, 2009, pp. 355–370, doi: 10.1007/978-3-642-04444-1_22.

[3] Q. Zhiguang, W. Shiyu, and Z. Yang, "An auditing protocol for data storage in cloud computing with data dynamics," *J. Comput. Res. Develop.*, vol. 52, no. 10, pp. 2192–2199, 2015, doi: 10.7544/issn1000-1239.2015.20150509.

[4] H. Wang, J. Gu, X. Di, D. Liu, J. Zhao, and X. Sui, "Research on classification and recognition of attacking factors based on radial basis function neural network," *Cluster Comput.*, vol. 22, no. S3, pp. 5573–5585, May 2019, doi: 10.1007/s10586-017-1371-9.

[5] A. Juels and B. S. Kaliski, Jr., "PORs: Proofs of retrievability for large files," in *Proc. 14th ACM Conf. Comput. Commun. Secur.*, 2007, pp. 584–597, doi: 10.1145/2699909.

[6] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, and D. Song, "Provable data possession at untrusted stores," in *Proc. 14th ACM Conf. Comput. Commun. Secur. (CCS)*, 2007, pp. 598–609, doi: 10.1145/1315245.1315318.

[7] H. Shacham and B. Waters, "Compact proofs of retrievability," in *Proc. Int. Conf. Theory Appl. Cryptol. Inf. Secur.* Melbourne, VIC, Australia: Springer, 2008, pp. 90–107, doi: 10.1007/978-3-540-89255-7_7.

[8] C. C. Erway, A. Küpçü, C. Papamanthou, and R. Tamassia, "Dynamic provable data possession," *ACM Trans. Inf. Syst. Secur.*, vol. 17, no. 4, p. 15, 2015.
[9] C. Wang, S. S. Chow, Q. Wang, K. Ren, and W. Lou, "Privacy-preserving public auditing for secure cloud storage," *IEEE Trans. Comput.*, vol. 62, no. 2, pp. 362–375, Feb. 2013, doi: 10.1109/TC.2011.245.
[10] Y. Zhu, H. Wang, Z. Hu, G.-J. Ahn, H. Hu, and S. S. Yau, "Efficient provable data possession for hybrid clouds," in *Proc. 17th ACM Conf. Comput. Commun. Secur. (CCS)*, 2010, pp. 756–758, doi: 10.1145/1866307.1866421.
[11] Y. Zhu, G.-J. Ahn, H. Hu, S. S. Yau, H. G. An, and C.-J. Hu, "Dynamic audit services for outsourced storages in clouds," *IEEE Trans. Services Comput.*, vol. 6, no. 2, pp. 227–238, Apr. 2013, doi: 10.1109/tsc.2011.51.
[12] B. Wang, B. Li, and H. Li, "Oruta: Privacy-preserving public auditing for shared data in the cloud," *IEEE Trans. Cloud Comput.*, vol. 2, no. 1, pp. 43–56, Jan. 2014, doi: 10.1109/tcc.2014.2299807.
[13] Y. Yu, J. Ni, M. H. Au, H. Liu, H. Wang, and C. Xu, "Improved security of a dynamic remote data possession checking protocol for cloud storage," *Expert Syst. Appl.*, vol. 41, no. 17, pp. 7789–7796, Dec. 2014, doi: 10.1016/j.eswa.2014.06.027.
[14] B. Wang, B. Li, and H. Li, "Knox: Privacy-preserving auditing for shared data with large groups in the cloud,' in *Proc. Int. Conf. Appl. Cryptogr. Netw. Secur.* Singapore: Springer, 2012, pp. 507–525, doi: 10.1007/978-3-642-31284-7_30.
[15] K. Yang and X. Jia, "An efficient and secure dynamic auditing protocol for data storage in cloud computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 9, pp. 1717–1726, Sep. 2013, doi: 10.1109/tpds.2012.278.
[16] Y. Zhu, H. Hu, G.-J. Ahn, and M. Yu, "Cooperative provable data possession for integrity verification in multicloud storage," *IEEE Trans. Parallel Distrib. Syst.*, vol. 23, no. 12, pp. 2231–2244, Dec. 2012, doi: 10.1109/tpds.2012.66.
[17] Yanyan. Fu, Min. Zhang, Kaiqu. Chen, "Proofs of data possession of multiple copies," *J. Comput. Res. Develop.*, vol. 51, no. 7, pp. 1410–1416, 2014, doi: 10.7544/issn1000-1239.2014.20131782.
[18] H. Li, R. Lu, J. Misic, and M. Mahmoud, "Security and privacy of connected vehicular cloud computing," *IEEE Netw.*, vol. 32, no. 3, pp. 4–6, May 2018, doi: 10.1109/mnet.2018.8370870.
[19] P. Li, J. Li, Z. Huang, T. Li, C.-Z. Gao, S.-M. Yiu, and K. Chen, "Multi-key privacy-preserving deep learning in cloud computing," *Future Gener. Comput. Syst.*, vol. 74, pp. 76–85, Sep. 2017, doi: 10.1016/j.future.2017.02.006.
[20] K. Vijayakumar and C. Arun, "Continuous security assessment of cloud based applications using distributed hashing algorithm in SDLC," *Cluster Comput.*, vol. 22, no. S5, pp. 10789–10800, Sep. 2019, doi: 10.1007/s10586-017-1176-x.
[21] J. Zhou, Z. Cao, X. Dong, and A. V. Vasilakos, "Security and privacy for cloud-based IoT: Challenges," *IEEE Commun. Mag.*, vol. 55, no. 1, pp. 26–33, Jan. 2017, doi: 10.1109/mcom.2017.1600363cm.
[22] L. Zhu, Y. Wu, K. Gai, and K.-K.-R. Choo, "Controllable and trustworthy blockchain-based cloud data management," *Future Gener. Comput. Syst.*, vol. 91, pp. 527–535, Feb. 2019, doi: 10.1016/j.future.2018.09.019.
[23] G. Ateniese, R. Burns, R. Curtmola, J. Herring, O. Khan, L. Kissner, Z. Peterson, and D. Song, "Remote data checking using provable data possession," *TISSECACM Trans. Inf. Syst. Secur.*, vol. 14, no. 1, pp. 1–34, May 2011, doi: 10.1145/1952982.1952994.
[24] C.-Z. Gao, Q. Cheng, X. Li, and S.-B. Xia, "Cloud-assisted privacy-preserving profile-matching scheme under multiple keys in mobile social network," *Cluster Comput.*, vol. 22, no. S1, pp. 1655–1663, Jan. 2019, doi: 10.1007/s10586-017-1649-y.
[25] M. Jouini and L. B. A. Rabai, "A security framework for secure cloud computing environments," in *Cloud Security: Concepts, Methodologies, Tools, Applications*. Hershey, PA, USA: IGI Global, pp. 249–263.

CHUNBO WANG received the B.S. and M.S. degrees in science from the Changchun University of Science and Technology, China, in 2010 and 2015, respectively, where he is currently pursuing the Ph.D. degree in computer science and technology. He is working with the Changchun Automobile Industry Institute, and is also an important member of the Jilin Province Key Laboratory of Network and Information Security. His research interests include computer network security and information security.

XIAOQIANG DI received the B.S. degree in computer science and technology and the M.S. and Ph.D. degrees in communication and information systems from the Changchun University of Science and Technology, in 2002, 2007, and 2014, respectively. He was a Visiting Scholar with the Norwegian University of Science and Technology, Norway, from August 2012 to August 2013. He is currently a Professor and a Supervisor of Ph.D. with the Changchun University of Science and Technology, where he is also the Head of the Jilin Province Key Laboratory of Network and Information Security and Information Center. His major research interests include network information security and integrated networks.