IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received April 8, 2019, accepted May 2, 2019, date of publication May 15, 2019, date of current version June 6, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2916889

A Lightweight Secure Auditing Scheme for Shared Data in Cloud Storage

JUNFENG TIAN AND XUAN JING
Cyberspace Security and Computer College, Hebei University, Baoding 071000, China
Key Laboratory on High Trusted Information System in Hebei Province, Baoding 071000, China

Corresponding author: Xuan Jing (abidble@gmail.com)

This work was supported by the Natural Science Foundation, China, under Grant 61800060654.

ABSTRACT A cloud platform provides users with shared data storage services. To ensure shared data integrity, it is necessary to validate the data effectively. An audit scheme that enables group members to modify data conducts the integrity verification of the shared data, but this approach results in complex calculations for the group members. The audit scheme of the designated agent implements a lightweight calculation for the group members, but it ignores the security risks between the group members and the agents. By introducing Hashgraph technology and designing a Third Party Medium (TPM) management strategy, a lightweight secure auditing scheme for shared data in cloud storage (LSSA) is proposed, which achieves security management of the groups and a lightweight calculation for the group members. Meanwhile, a virtual TPM pool is constructed by combining the TCP sliding window technology and interconnected functions to improve agent security. We evaluate our scheme in numerical analysis and in experiments, the results of which demonstrate that our scheme achieves lightweight computing for the group members and ensures the data verification process for security.

INDEX TERMS Shared data, virtual TPM pool, lightweight calculation, agent security.

I. INTRODUCTION

Cloud computing is a new computing mode that was created after peer-to-peer computing, grid computing, utility computing and distributed computing. The core concept of cloud computing is resource renting, application hosting and service outsourcing [1]. Through virtualization technology, it forms distributed computing nodes into a shared virtualization pool in order to provide services for users.

With cloud computing technology, users and enterprises do not need to spend much on the acquisition and maintenance of hardware in their early stages. In addition, powerful computing and storage capabilities also make users more willing to rely on the cloud to handle a variety of complex tasks. When users choose to deploy a large number of applications and data to the cloud computing platform, the cloud computing system accordingly becomes the cloud storage system. Cloud storage systems give users mass storage capacity at a relatively low price, and provide a platform for sharing data between users (data sharing means that a user in a group uploads data to the cloud, and the rest of the group can access/modify the data) [2]. However, highly centralized computing resources means cloud storage faces severe security challenges.

According to a survey conducted by Gartner in 2009, 70% of CEOs of surveyed companies refused to adopt cloud computing models on a large scale due to concerns about the privacy of cloud data. Furthermore, in recent years the security storage problem exposed by cloud operators has aroused people's concern. For example, in March 2011, Google Gmail failed, which caused data loss to approximately 150,000 users. In the same year, Amazon's enormous EC2 cloud service crashed, permanently destroying some users' data. While the data loss was apparently small relative to the total amount of data stored, anyone who runs a website can immediately understand the horrible level of data loss [3]. Thus, the secure storage of data in the cloud has hindered the large-scale use of cloud computing in the IT field [4]. To achieve the secure storage of cloud data, researchers have developed the cloud data integrity verification scheme.

A. RELATED WORK

In 2007, Ateniese *et al.* first proposed a Provable Data Possession (PDP) model, which can verify the integrity of cloud

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

data without retrieving all of the data [5]. Then, Juels *et al.* proposed the Proofs of Retrievability (POR) scheme, which enables a back-up or archive service to produce proof that the data can be retrieved by the verifier [6]. In a subsequent study, Ateniese *et al.* implemented a PDP scheme that supports dynamic operations [7], which means that the data uploader has full control over any operation performed on the cloud data, including block deletion, modification, and insertion. Then, Waters *et al.* proposed a full-dynamic PDP scheme by utilizing the authenticated flip table [8].

Differing from these works, the following schemes [9]–[14] focus on how to audit the integrity of the shared data. In this scenario, users can easily modify and share data as a group with the cloud services, where every group member in the group is not only able to access and modify the shared data but also share the version that he/she has modified with the rest of the group [11].

In 2016, Yang *et al.* proposed a BLS-based signature scheme supporting flexible management in the group [9]. Jiang *et al.* proposed data integrity based on the vector commitment technique, which is resistant to collusion attacks of a cloud service provider and a group member [10]. By combining proxy cryptography with the encryption technique, in 2017 Luo *et al.* proposed a scheme with secure user revocation [11]. Recently, Huang *et al.* realized efficient key distribution within groups based on the logical hierarchy tree, thereby protecting the identity privacy of the group members [12]. Huang *et al.* subsequently proposed a certificateless audit scheme by eliminating key escrow, which further improved the user's privacy security [13]. Following Huang *et al.*'s pioneering work. Fu *et al.* proposed an audit scheme that can restore the latest correct shared data blocks by changing the binary tree tracking data in the group [14].

In the above scheme [9]–[14], in order to verify the integrity of the shared data stored in the cloud, the group members need to block the data and then calculate the data authentication label for each block. Finally, the group member uploads the shared data along with the corresponding authentication labels to the cloud. The integrity verification of the shared data relies on the correctness of these data authentication labels. However, the cost of calculating the authentication label is generally great, because the formula requires a large number of exponentiations, e.g., when the block size is 2 KB, the authentication label generation overhead for a 10 GB file is nearly 18 hours. Therefore, it is necessary to propose a lightweight auditing scheme to reduce the resource utilization of users. Li *et al.* proposed a new cloud storage auditing scheme with a cloud audit server and a cloud storage server [15]. The cloud audit server generates authentication labels for users before uploading them to the cloud storage server. Although this scheme can reduce users' computation overhead, it fully reveals the user's private key and the user's data to the cloud audit server. As a result, malicious cloud service providers can pass the verification process without storing the user's data. Guan *et al.* used an indistinguishable confusing approach to build an audit scheme for cloud storage [16], thereby reducing the time that is required to generate authentication labels but increasing the time to verify the integrity of the cloud data. Wang *et al.* introduced agents to assist group members in generating authentication labels and auditing data integrity [17], which alleviated the computational burden for group members. However, in order to guarantee data privacy, the group member needs to encrypt the data before sending them to the proxy, which inevitably increases the computational burden. Shen *et al.* proposed a lightweight audit scheme by introducing the Third Party Medium (called the agent) to replace group members with generating authentication labels [18]. Different from Wang *et al.*'s scheme, the scheme uses blind data instead of encrypted data to generate authentication labels, further reducing the computational burden on the group members. Although the scheme protects the data privacy and the identity privacy of group members in some ways, it does not consider the possibility of illegal access to shared data. Since the data of the malicious group member is also encrypted or blinded, it cannot be detected even after other people's data is randomly modified. What is even worse is that malicious group members can collude with agents for illegal profit.

B. MOTIVATION

A malicious cloud server is able to discard all the shared data and generate a valid proof of data possession by reserving some intermediate results or a previous valid proof, which we refer to as a replace attack and a replay attack, respectively. A malicious group member is able to modify other member's data in that group without being discovered. A malicious agent is able to collude with illegal group members to steal user data and identity information. As far as we know, the three points mentioned above are still open challenges to design a secure integrity auditing scheme for shared data with lightweight computing on the client side.

To solve those challenging problems, we proposed a lightweight secure auditing scheme for shared data in cloud storage (LSSA). Similar to the cloud storage audit scheme [18], using the Third Party Medium (TPM) instead of group members to calculate the authentication label and audit data integrity results in lightweight calculations for the group members. Differing from that scheme, we separated the group members and the TPM through a group manager, to realize the division and governance of the group members and the TPM and eliminate the collusion between them. In terms of group members, we employ a blind method to blind the data in order to protect their privacy information. In addition, by introducing a Hashgraph, the modified data records of the group members can be recorded, which results in traceability of the group membership, and illegal behaviours of the group members can be contained through Hashgraph technology. In terms of the TPM, a virtual TPM pool was designed, and multiple TPMs were authorized by group manager to perform computing tasks. The virtual TPM pool acts as a secret box that is hidden from the outside world. Only the group manager knows which TPMs are calculating in the box, and there is

no correlation between TPMs. Thus, TPM security management is realized. Additionally, the utilization of our proposed auditing process, which is free from the replay and replace attacks mentioned above, makes the auditing of our scheme more secure.

Our research contributions can be summarized as follows:

(1) By introducing an efficient blind method, this paper ensures the data privacy and identity privacy of the group members. By introducing a Hashgraph, this paper avoids the hidden security risks of group members, and simultaneously makes the user identity traceable.

(2) The TPM management strategy is designed, and the virtual TPM pool is built by the group manager. The strategy ensures the security of agent (TPM) and results in lightweight calculations for the agent. Using the TPM instead of group members to calculate the authentication label and audit data integrity results in lightweight calculations for the group members.

(3) The security analysis of the scheme shows that the scheme is safe and can resist both replace attacks and replay attacks.

(4) The experimental evaluation of the scheme shows that the scheme can achieve lightweight calculations for group members and the TPM.

C. ORGANIZATION

The remainder of this paper is organized as follows: Section II introduces the system model and design goals. Section III covers the preparation knowledge. Section IV introduces the main design idea of the LSSA. Section V covers a detailed description of the LSSA scheme. Section VI covers the security analysis. Section VII introduces the experiment evaluation. Section VIII presents the conclusion.

II. SYSTEM MODEL AND DESIGN GOALS

A. SYSTEM MODEL

The system model of this scheme consists of four different entities: the Group members (M), the Cloud, the Group Manager (GM), and the TPM. As shown in Fig. 1, there are multiple group members in a group. After the data owner (the individual or organization that owns the original data) creates the data file and uploads it to the cloud, any group member can access and modify the corresponding shared data. Note that the original data owner can play the role of GM and there is only one GM in each group. The M play two important roles: 1) blind data, and 2) record blind data and broadcast within the group through a Hashgraph. The cloud (e.g., iCloud, OneDrive, and Baidu Cloud) provides data storage services for group members and provides a platform for group members to share data. The GM plays three important roles: 1) generate the TPM's public-private key pair, 2) formulate the TPM management strategy, and 3) generate the secret seed that is used to blind the data for group members and to recover the real data for the cloud. The TPM plays two important roles: 1) generate data authentication label for group members, and 2) verify the integrity of the cloud data on behalf of the group members.

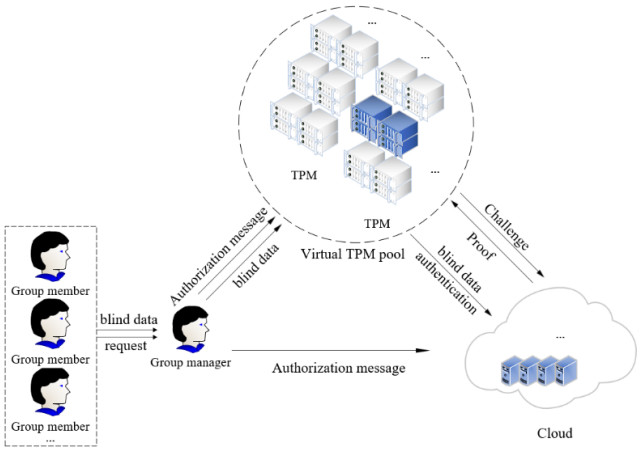

FIGURE 1. System model.

The execution procedure is divided into the data upload stage and the audit stage. Before the group members make a request to upload the modified data to the cloud, the data are first blinded by the secret seed and recorded by the Hashgraph, and then sent to the group manager. According to the TPM management strategy, the group manager selects a TPM from the virtual TPM pool (Section IV.*B* for details) for authorization, and the authorized TPM calculates the corresponding authentication labels for these blinded data within the authorization time. Then, the blind data and authentication label are sent to the cloud. Before receiving these messages, the cloud will check whether or not the authorization from the TPM is valid at the current time. If it is, he verifies whether or not the authentication labels are correct. If they are correct, he will recover the real data and compute their authentication labels. Finally, the cloud stores these real data and authentication labels. Before executing the auditing procedure, the group manager selects a TPM and creates the authorization according to the TPM management strategy. Then, the authorized TPM sends the challenge messages to the cloud. Before receiving these messages, the cloud will check whether or not the authorization from the TPM is valid. If it is, the cloud generates a proof of possession of the shared data. Finally, the TPM can verify the integrity of shared data in the cloud by checking the correctness of the proof.

B. DESIGN GOALS

(1) Lightweight computing: This approach ensures that group members do not need to perform time-consuming calculations during the generation of authentication labels or during the audit of the shared data. Multiple TPMs take part in the calculation, thereby ensuring a lightweight calculation of a single TPM.

(2) Identity traceability: The modification of data by illegal group members may lead to disputes among the group members using the same shared data. This goal ensures that the

GM can find and remove any illegal group members, thereby achieving the security management of groups.

(3) TPM management security: Each TPM works independently to ensure legal participation of the TPM. This goal ensures that the cloud only accepts and stores the data of TPMs that are authorized by the GM, and it only responds to the challenge of the TPMs that are authorized by the GM.

(4) Data privacy and identity privacy: When the TPM generates authentication labels instead of group members, it is impossible to know the actual information of the data block. The TPM cannot acquire the identity information of group members at the stages of uploading data and auditing data.

(5) Audit correctness and security: The TPM can verify the integrity of the shared data through the audit process. Malicious cloud service providers cannot complete the audit process through replace or replay attacks.

III. PREPARATION KNOWLEDGE

The main preparation knowledge is as follows: bilinear pair mapping, the Computational Diffie-Hellman (CDH) problem, the Interconnection function, and Hashgraph technology.

A. BILINEAR PAIR MAPPING [19]

Let G_1 and G_2 be two multiplicative loop groups with a large prime order p, and g_1 and g_2 be the generators of group G_1. A bilinear pairing is a map $e : G_1 \times G_1 \rightarrow G_2$

with the following properties:

(1) Bilinearity: for $\forall g_1, g_2 \in G_1$ and $a, b \in Z_p^*$, $e(g_1^a, g_2^b) = e(g_1, g_2)^{ab}$;

(2) Non-degeneracy: $e(g_1, g_2) \neq 1$; and

(3) Computability: there is an efficient algorithm to compute this pairing.

B. COMPUTATIONAL DIFFIE-HELLMAN (CDH) PROBLEM [20]

For $x, y \in Z_p^*$, given $g, g^x, g^y \in G_1$, we calculate $g^{xy} \in G_1$. The CDH assumption in G_1 holds if solving the CDH problem in G_1 is computationally infeasible.

C. INTERCONNECTION FUNCTION [21]

N input ports and N output ports are connected by an interconnection function f, where $f(x) = x$, $0 \leq x \leq N-1$, and x is the address of the port number, which is usually expressed in n bits. Therefore, the interconnection function is represented as $f(x_{n-1}x_{n-2}\ldots x_1x_0)$.

For example, in the cube permutation function $C_k(x_{n-1}x_{n-2}\ldots x_{k+1}x_kx_{k-1}\ldots x_1x_0) = x_{n-1}x_{n-2}\ldots x_{k+1}\bar{x}_kx_{k-1}\ldots x_1x_0$, the interconnection function can be represented by $n = \log_2 N$ subfunctions of cube permutation. When the number of members in the group is $N = 8$, the number of cube permutation subfunctions is $n = \log_2 8 = 3$, where $C_0(x_2x_1x_0) = x_2x_1\bar{x}_0$, $C_1(x_2x_1x_0) = x_2\bar{x}_1\ x_0$, and $C_2(x_2x_1x_0) = \bar{x}_2\ x_1x_0$.

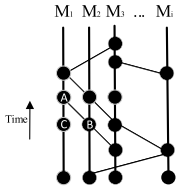

FIGURE 2. Hashgraph.

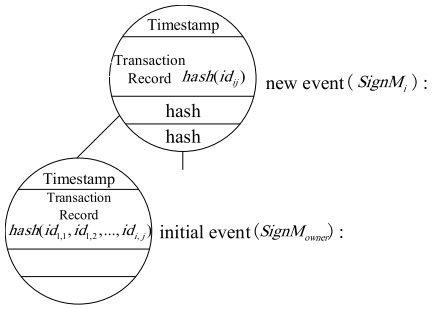

FIGURE 3. Event diagram.

Similar interconnection functions include the identity permutation, exchange permutation, butterfly permutation, bit reversal inversion permutation, shuffle-exchange permutation, and shift permutation. Each permutation function also has some corresponding permutation subfunctions.

D. HASHGRAPH TECHNOLOGY [22]

As shown in Fig. 2, each circle in the figure represents an event, which is then represented by a hash value. The earlier *Time* vertices represent early events in the historical records, and M_i represents the user i. The message is propagated in the Hashgraph network in the Gossip mode. After event B occurs, user M_2 who generated B appends this event with its own signature, Sign M_2, and randomly sends it to user M_1 randomly. User M_1 receives this message and creates a new event A. Event A contains two event hashes (his own historical event C and the user M_2 gossip synchronized event B), and user M_1 attaches event A to his own signature, Sign M_1, and randomly sends it to other users.

IV. MAIN DESIGN IDEA OF THE LSSA

In this section, we use Hashgraph technology to propose the design idea of group member management. By referring to the TCP sliding window [23] and using the Interconnection function, the design idea of the TPM management strategy is proposed.

A. DESIGN IDEA OF GROUP MEMBER MANAGEMENT

When a user registers with the group, the group manager randomly generates an account as the identity label of member M of the group, where according to the account, the actual identity of the group member can be determined. When the group member is removed, the group manager removes the account. For notational simplicity, we define the following notations.

- m: The data files are divided into n blocks (m_1, m_2,...m_n), and each block is represented by m_i, where

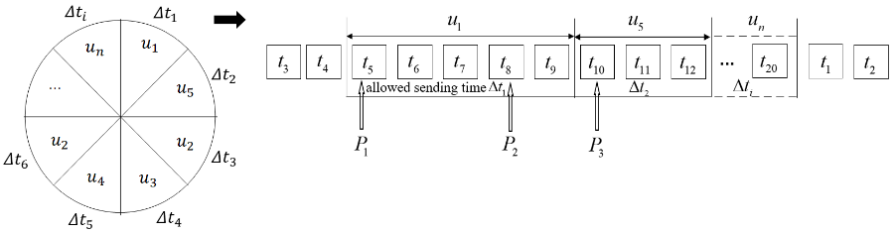

FIGURE 4. Sending window.

m_i is fragmented into s slices ($m_{i,1}$,$m_{i,2}$,...$m_{i,s}$), and each slice is represented by m_{ij}.

- $m_{ij}^{'}$: The blind data block corresponding to m_{ij}.
- $id_{i,j}$: The public identifier information of the blind data block $m_{ij}^{'}$.
- M_{Owner}: The account (ID) of the data owner.
- $hash$: The hash function, e.g., SHA-1 and SHA-256.
- $Sign$: The signature on the ID of group members.

As shown in Fig. 3, before the data owner M_{Owner} sends the blind data block $m_{ij}^{'}$ to the group manager, who calculates the hash value $hash(id_{i,j})$ of $id_{i,j}$ as the upload record (called the transaction record) of the initial event and attaches the signature $SignM_{Owner}$. According to the Hashgraph technique described in Section III.D, the group member or group manager is randomly selected to synchronize this with initial event, thereby sending the event to the nodes in the network. The members in the group can access and modify the original shared data, but the group members M_i that have modified and accessed $m_{ij}^{'}$ since then need to update the identifier of the blind block after use. Thus, the members calculate the hash value of $id_{i,j}$ as a modify/access record (called a transaction record) for a new event and attach the signature $SignM_i$ to spread it within the group.

B. DESIGN IDEA OF THE TPM MANAGEMENT STRATEGY

After each group member sends a request to upload the shared data, the group manager selects a TPM for authorization. The port number address of the group member M_i is $u_i(x_0, x_1, \ldots, x_\theta)$ (θ is the number of bits in the binary address), and the port number address of the TPM is $TPM_i(x_0, x_1, \ldots, x_\theta)$. The following describes the detailed selection method.

(1) The group manager chooses the processing time of the request.

Referring to the principle of the TCP sliding window, the sending window is set for the input end, and the sending window corresponds to a period of time. During this period, the group manager receives the application sent by the group member.

As shown in Fig. 4, the sending window has 3 pointers that slide clockwise. The window that allows u_1 to send is the time between P_1 and P_3, and the current time is P_2. Suppose that the group manager divides a certain time period into 20 parts as t_1-t_{20}. If group member M_1 sends an application at t_8, which is in the allowable sending time of the sending window,

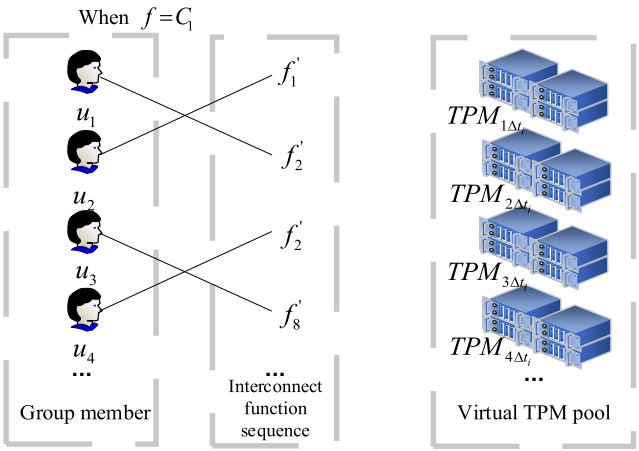

FIGURE 5. Virtual TPM pool construction diagram.

the group manager will then receive the sent application from M_1. According to the time frame rotation, if group member M_1 does not send the request in the allowed sending time, P_2 slides clockwise to $\Delta t_1^{'}$, which corresponds to M_1's port number address u_1, and then $\Delta t_1^{'}$ becomes the time to process the request of M_1.

Therefore, the distance between P_1 and P_3 determines the success rate of the application of the group members. The group manager and the cloud protocol have a consistent sending window that is updated periodically as needed. When moving to the right between P_1 and P_3, the size of the window changes. For group members who frequently use shared data, the group manager assigns them more time and cancels the allowed sending period for the revoked group members.

(2) The group manager selects the output address TPM_i based on the time of the request that was processed and the address u_i of the input end.

To increase the possibility of selection, the group manager selects $f, f_i^{'}$ to be used. (f and $f_i^{'}$ are the interaction function and the interaction function sequence, respectively, and the cloud service provider and the group manager negotiate consistent f and $f_i^{'}$.)

As shown in Fig. 5, assuming that u_2 sends the application to the group manager at Δt_1, the group manager selects the interconnection function $f = C_1$ and selects $f_i^{'} = C_0$ from the sequence of interconnection functions. According to the sending window in Fig. 4, the round is transferred to the

time period Δt_3, where the group manager can calculate the output at the moment through u_2, Δt_3, and the interconnection function C_0. That is, at that moment, the correspondence between the input and output can be represented by a matrix

$$\begin{array}{c} \\ TPM_{1\Delta t_3} \\ TPM_{2\Delta t_3} \\ TPM_{3\Delta t_3} \\ TPM_{4\Delta t_3} \end{array} \begin{array}{c} \begin{array}{cccc} u_1 & u_2 & u_3 & u_4 \end{array} \\ \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \end{array} \quad (i = 4),$$

which indicates that the group manager selects the output address TPM_1 through u_2 and C_0 at Δt_3.

(3) The group manager selects the time to authorize the TPM.

Similar to the process of selecting the processing request time, the group manager sets the sending window for the output, where the output TPM_i of the corresponding u_i is valid only at time Δt_i. For an efficient and secure TPM, the group manager allocates more time for it, and cancels the allotted sending period for the removed TPM.

Therefore, after selecting the output address TPM_1, the group manager determines whether Δt_3 corresponds to TPM_1 according to the sending window of the output end. If it does, Δt_3 will be sent to the cloud as the time authorized by the TPM; otherwise it will select the time authorized by the TPM according to the time frame rotation.

Through the above strategy, the group manager is provided with various execution modes. According to the time frame rotation and the interconnection function, the group manager can dynamically select TPM_i. The group manager adjusts the time according to the actual situation. After one rotation, the group manager can send a new f to the cloud so that both the output time and the function change. The group member sends an application to upload data, and the group manager 'randomly distributes' the computing task to the selected TPM_i according to the above rules. This approach is equivalent to building a virtual TPM pool, such that each TPM and each group member are independent of one another.

V. DETAILED DESCRIPTION OF THE LSSA SCHEME

A. OVERVIEW

In the data upload phase, the group manager generates the TPM's public-private key pair. He also generates a secret seed, and then sends it to the group members and the cloud. Because the group manager's port is the port connection point between the group members and the TPMs, he can select the send window and interaction functions, create the authorization according to the TPM management strategy, and then issue this authorization to the TPM. When the user wants to upload data to the cloud, he first computes the blinding factor using the secret seed to blind these data, then calculates the hash value of the blind data as a transaction record for a new event, then broadcasts it within the group, and then sends them to the group manager. Before receiving these messages, the group manager will check whether or not the hash value from the member is valid. If it is, he will send the authorization to the TPM. Then, the TPM will generate the corresponding authentication labels for these blinded data and upload these blinded data and their authentication labels to the cloud together. Before recovering these messages, the cloud will check whether or not the authorization from the TPM is valid at the current time. If it is, he verifies whether or not these authentication labels are correct. If they are correct, he will recover the real data using the blinding factor and compute their authentication labels. Finally, the cloud stores these real data and the authentication labels.

Before executing the audit procedure, the group manager selects the send window and interaction functions and creates the authorization according to the TPM management strategy as described above. Then, the authorized TPM generates a challenge message (CM) and sends it to the cloud. Before receiving these challenges, the cloud will check whether or not the authorization from the TPM is valid. If it is, then based off the challenge message the cloud is able to generate a proof of possession of the shared data. Finally, the TPM is able to verify the integrity of the shared data in the cloud by checking the correctness of the proof.

TABLE 1. Notations.

Symbol	Descriptions
G_1, G_2	Two multiplicative cyclic groups with a large prime order p.
g	The generator of group G_1.
Z_p^*	A prime field with nonzero elements.
H	A secure hash function such that $H_1(\cdot):\{0,1\}^* \times G_1 \rightarrow Z_p^*$, $H_2(\cdot):\{0,1\}^* \rightarrow G_1$.
$m=\{(m_{11},..,m_{ns})\}$	A shared data file with n blocks and s slices.
$m_{ij}^{'}$	The blind data block.
k_1	The secret seed (input key) of a pseudo-random function [24].
ζ_{k_1}	A pseudo-random function such that $\zeta_{k_1}: Z_p^* \times Z_p^* \rightarrow Z_p^*$.
β_i	TPM_i's private key used to generate data authentication label.
α^j	The number of random values α is consistent with the number of slices j of the data block.
α_i	The blinding factor corresponding to the ith data block such that $\alpha_i = \zeta_{k_1}(i, name)$.
ID_{group}	The identity of the group manager.
$\sigma_i^{'}$	The authentication label for the ith blind data block.
σ_i	The authentication label for the ith data block.
d	The number of TPM.

B. SCHEME DESCRIPTION

Table 1 shows the symbolic descriptions of this section.

We describe the interaction among the TPM, the cloud, the group manager and the group members in our scheme

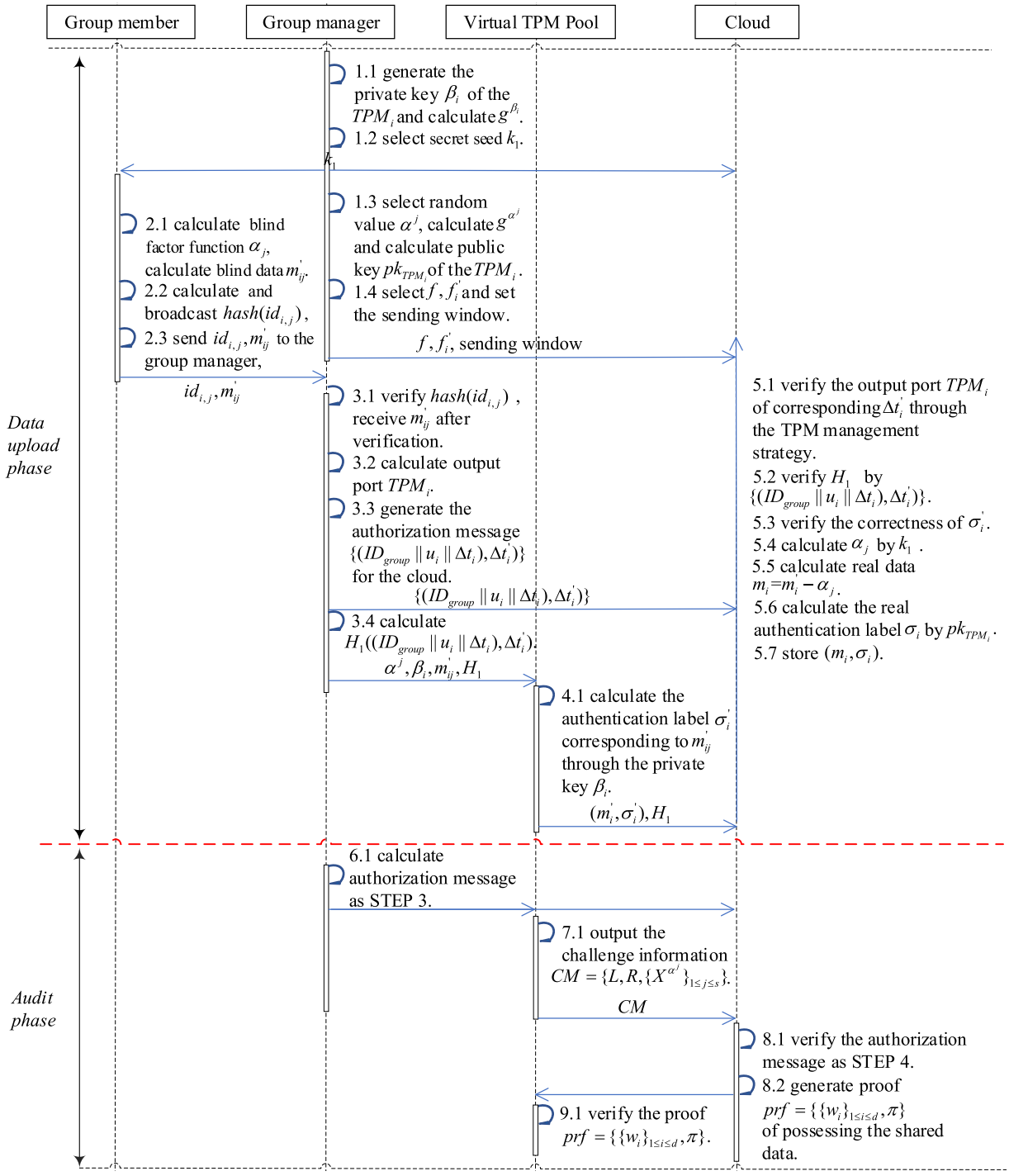

FIGURE 6. Algorithm flowchart.

in Fig. 6. The data upload phase can be completed using steps 1-6, and the audit phase can be completed using steps 7-9:

STEP 1. (Key Generation):

1) The group manager randomly selects $\beta_i \in Z_p^*$, which will act as the private key of TPM_i. He then calculates g^{β_i}.

2) The group manager randomly selects $k_1 \in Z_p^*$ and sends it to the group members and the cloud.

3) The group manager randomly selects $\alpha^j \in Z_p^*$, calculates g^{α^j}, and computes pk_{TPM_i} as the public key of TPM_i:

$$pk_{TPM_i} = (g^{\beta_i}, g^{\alpha^1 \beta_i}, g^{\alpha^2 \beta_i}, \ldots, g^{\alpha^j \beta_i}) \quad (1)$$

4) The group manager selects the interconnection function f, the interconnection function sequence $f_i^{'}$, and the

sending windows of the input and output, and then sends them to the cloud.

STEP 2. (Data Blind):

1) Group members use the secret seed k_1 to calculate the blind factor $\alpha_i = \zeta_{k_1}(i, name)$, which in turn calculates the blind data $m_{ij}^{'}$, i.e.,

$$m_{ij}^{'} = m_{ij} + \alpha_i \tag{2}$$

2) A group member sends a request to upload the data $m_{ij}^{'}$ to the group manager, calculates the hash value $hash(id_{i,j})$, and sends $id_{i,j}$,$m_{ij}^{'}$ to the group manager through the secure channel. A new event is then created by the group member. The $hash(id_{i,j})$ will be used as a transaction record for the new event and will be broadcasted within the group. After receiving the request, the group manager verifies $hash(id_{i,j})$ according to the same hash algorithm and receives $m_{ij}^{'}$ after the verification is passed.

STEP 3. (Authorize):

1) The group manager calculates the output port TPM_i in the virtual TPM pool corresponding to the input port u_i (the requesting group member) according to the TPM management strategy.

2) The group manager generates the authorization message for TPM_i as follows.

$$\left\{ (ID_{group}||u_i||\Delta t_i), \Delta t_i^{'} \right\} \tag{3}$$

where ID_{group} is the identity of the group manager, Δt_i is the time when the group manager processes the request, and $\Delta t_i^{'}$ is the time authorized by the group manager for the TPM.

3) The group manager calculates the value H_1 according to the authorization message as follows.

$$H_1 = H_1((ID_{group}||u_i||\Delta t_i), \Delta t_i^{'}) \tag{4}$$

4) The group manager then sends the authorization message (3) to the cloud, and sends α^j,β_i,$m_{ij}^{'}$, and H_1 to TPM_i.

STEP 4. (Authentication label Generation):

1) After receiving the blind data block $m_{ij}^{'}$, TPM_i uses private key β_i to generate authentication label $\sigma_i^{'}$ of $m_{ij}^{'}$, i.e.,

$$\sigma_i^{'} = (H_2(i) \cdot \prod_{j=1}^{s} (g^{\alpha^j})^{m_{ij}^{'}})^{\beta_i} \tag{5}$$

e.g., if $m_5^{'}$ is the modified blind block, then $\sigma_5^{'} = (H_2(5) \cdot \prod_{j=1}^{s} (g^{\alpha^j})^{m_{5j}^{'}})^{\beta_i}$.

Then, TPM_i send the data file $(m_{ij}^{'}, \sigma_i^{'})$ and H_1 to the cloud.

2) After receiving the $(m_{ij}^{'}, \sigma_i^{'})$ of the corresponding TPM_i and the authorization message (3) of the corresponding group manager, the cloud first calculates the output port TPM_i. If TPM_i just sends the message at $\Delta t_i^{'}$, then the cloud calculates $H_1((ID_{group}||u_i||\Delta t_i), \Delta t_i^{'})$ and determines whether the value is consistent with the value $H_1((ID_{group}||u_i||\Delta t_i), \Delta t_i^{'})$ from TPM_i. If they are consistent, STEP 5 is executed; otherwise the execution is refused.

STEP 5. (Authentication label Check):

The cloud verifies the correctness of label $\sigma_i^{'}$ using the following equation:

$$e(\sigma_i^{'}, g) = e(H_2(i) \cdot \prod_{j=1}^{s} (g^{\alpha^j})^{m_{ij}^{'}}, g^{\beta_i}) \tag{6}$$

If the equation is true, $(m_{ij}^{'}, \sigma_i^{'})$ is received and stored; otherwise it is rejected.

STEP 6. (Data Recovery):

The cloud calculates $\alpha_i = \zeta_{k_1}(i, name)$ based on k_1, and then computes the real data m_{ij} using the following equation:

$$m_{ij} = m_{ij}^{'} - \alpha_i \tag{7}$$

The cloud calculates the real authenticator label σ_i according to pk_{TPM_i}, i.e.,

$$\sigma_i = \sigma_i^{'} \cdot \prod_{j=1}^{s} (g^{\alpha^j \beta_i})^{-\alpha_i} = (H_2(i) \cdot \prod_{j=1}^{s} (g^{\alpha^j})^{m_{ij}})^{\beta_i} \tag{8}$$

Finally, the cloud stores the real data blocks $m_{ij} = (m_{i1}, m_{i2}, \ldots, m_{is})$ and their corresponding real authenticator labels σ_i.

STEP 7. (Challenge):

When the group manager wants to initiate a challenge to the cloud, he randomly selects Δt as the authorization time to TPM_i, where Δt corresponds to the u_i of sending window on the input side. The group manager then sends an audit authorization command to the TPM_i through u_i at Δt, and sends $\left\{ ID_{group}||u_i||\Delta t \right\}$ as the audit authorization information to the cloud.

After receiving the authorization command from the group manager, the TPM_i implements the audit process.

1) TPM_i randomly selects c blocks from all blocks of the shared data and denotes the indexs of the selected blocks as L.

2) TPM_i generates two random numbers $o, r \in Z_p^*$, and calculates $X = \mathrm{g}^o$ and R=g^r.

3) TPM_i calculates $\{X^{\alpha^j}\}_{1 \le j \le s}$.

4) TPM_i outputs the challenge information:

$$CM = \{L, R, \{X^{\alpha^j}\}_{1 \le j \le s}\} \tag{9}$$

Then, TPM_i sends CM to the cloud.

STEP 8. (Proof Generation):

After receiving the challenge information CM, the cloud first calculates the output port TPM_i according to $\left\{ ID_{group}||u_i||\Delta t \right\}$. The cloud then uses the method in STEP 4 to verify the authorization message $\left\{ ID_{group}||u_i||\Delta t \right\}$. The cloud then generates the proof of possessing shared data as follows:

1) The index set L of the selected blocks is divided into subsets $L_1, \ldots, L_d$, where L_i is the subset of the selected blocks that are signed by TPM_i.

2) For each subset L_i, the cloud server calculates $u_{ij} = \sum_{l \in L_i} m_{lj}$ and $\pi_i = \prod_{l \in L_i} e(\sigma_l, R) = e(\prod_{l \in L_i} H_2(l) \prod_{j=1}^{s} g^{\alpha^j \sum_{l \in L_i} m_{lj}}, g)^{\beta_i r}$, where $1 \le i \le d$ and $1 \le j \le s$.

3) The cloud server calculates $w_i = \prod_{j=1}^{s} X^{\alpha^j u_{ij}}$ and $\pi = \prod_{i=1}^{d} \pi_i$. Then it returns prf as a response to the challenge message from the TPM_i, i.e.,

$$prf = \{\{w_i\}_{1 \le i \le d}, \pi\} \tag{10}$$

STEP 9. (Proof Check):

Based on the received prf and the challenge message CM, TPM_i verifies the integrity of the shared data by checking the correctness of the following equation:

$$\prod_{i=1}^{d} e(\eta_i^o, pk_{TPM_i}^r) \cdot e(w_i, pk_{TPM_i}^r) = \pi^o \tag{11}$$

where $\eta_i = \prod_{l \in L_i} H_2(l)$, $1 \le i \le d$. The above equation can be further rewritten as $(\prod_{i=1}^{d} e(\eta_i^o w_i, pk_{TPM_i}))^r = \pi^o$.

If the equation is true, then TPM_i outputs *True*; otherwise *False* is returned. In other words, if the selected block in the challenge has been tampered with, the cloud service provider cannot generate valid evidence, and the cloud service provider will not be able to pass the audit process from the TPM_i.

VI. SECURITY ANALYSIS

This section performs the security analysis separately from the audit correctness, audit security, data privacy, identity privacy, TPM security, and traceability of the group membership analyses.

A. AUDIT CORRECTNESS

When the shared data is properly stored in the cloud server, if the cloud provides a valid integrity certificate $prf = \{\{w_i\}_{1 \le i \le d}, \pi\}$, the validation procedure can then verify the integrity of the data.

Proof: According to the nature of bilinear mapping, the right side of the equation can be derived from the left side of the equation to prove the correctness of equation (11):

$$\begin{aligned}
&\prod_{i=1}^{d} e(\eta_i^o, pk_{TPM_i}^r) \cdot e(w_i, pk_{TPM_i}^r) \\
&= \prod_{i=1}^{d} e(\prod_{l \in L_i} H_2(l)^o, g^{\beta_i r}) \cdot e(\prod_{j=1}^{s} (g^o)^{\alpha^j \sum_{l \in L_i} m_{lj}}, g^{\beta_i r}) \\
&= \prod_{i=1}^{d} e(\prod_{l \in L_i} H_2(l), g)^{o\beta_i r} \cdot e(\prod_{j=1}^{s} g^{\alpha^j \sum_{l \in L_i} m_{lj}}, g)^{o\beta_i r} \\
&= \prod_{i=1}^{d} e(\prod_{l \in L_i} H_2(l) \prod_{j=1}^{s} g^{\alpha^j \sum_{l \in L_i} m_{lj}}, g)^{o\beta_i r} \\
&= \prod_{i=1}^{d} \pi_i^o = (\prod_{i=1}^{d} \pi_i)^o = \pi^o
\end{aligned}$$

Therefore, as long as the evidence comes from complete data, the validation equation will hold.

B. AUDIT SECURITY

The malicious cloud service provider cannot complete the audit process through replace attacks or replay attacks. ① Because each time the TPM initiates a challenge both $L = \{L_1, \ldots, L_d\}$ and $X = g^o$ are randomly generated, the cloud service provider cannot calculate $u_{ij} = \sum_{l \in L_i} m_{lj}$ and $w_i = \prod_{j=1}^{s} X^{\alpha^j u_{ij}}$ in advance, and therefore cannot implement a replace attack. ② Cloud service providers must calculate η^o ($\eta = \prod_{l \in L} H_2(l) \in G_1$) if they want to implement replay attacks. Suppose that the cloud service provider chooses $\psi \in Z_p^*$ to meet $\eta = g^\psi$. Since the CDH problem is computationally infeasible, the cloud service provider still cannot calculate $g^{\psi o}$ based on g^ψ, g and g^o.

C. DATA PRIVACY AND IDENTITY PRIVACY

① In the user upload data phase, the TPM cannot extract the real data m_{ij} through the blind data block $m_{ij}^{'}$. This finding is observed because the TPM receives $m_{ij}^{'} = m_{ij} + \alpha_i (j \in [1, s])$, where $\alpha_i = \zeta_{k_1}(i, name)$ is generated by group members through a random function. ② The TPM management strategy is flexible and secure. This strategy expands the method to select TPM_i and solves the problem of insufficient computing power of a single TPM. Each TPM independently performs computing tasks and cannot find more valuable information about group members through randomly distributed blind data blocks.

D. TPM SECURITY

The TPM public key is used to verify the integrity of the shared data. The final authentication label of the data block is actually encrypted using the TPM private key. Therefore, it is necessary to prevent the TPM from leaking the private messages for some reasons. ① A malicious group member may collude with the TPM. To this end, the group manager specifies multiple TPM, and each TPM works independently and distributes different private keys for it, which avoids the above problems. ② It is necessary to prevent the TPM from being maliciously attacked for some reason. By constructing a virtual TPM pool, only the group manager can calculate TPM_i, and those outside cannot find the target of the attack.

E. IDENTITY TRACEABILITY

Through Swirld's Hashgraph Consistency Algorithm [25], group members agree on the order of events (that is, the order of transaction records within the event) and the timestamp for each event (transaction record). The transaction records of each event can be determined in chronological order according to the Hashgraph. Once a member of the group maliciously modifies the data block, the dirty data block may be discovered by other group members. Once such a data dispute is generated, the group member may trace the usage

history of the modified data block according to a Hashgraph. Legal group members can open the data block information to prevent the illegal group members from collapsing the structure, and finally the group member whose data block information has illegal data is designated as an illegal group member.

VII. EXPERIMENT EVALUATION

According to the analysis of Section VI, the scheme of this paper avoids potential security risks through a more secure method. In the audit scheme of shared data, group members are more concerned with the efficiency problem when using data [18]. This section first analyses the computational overhead of the LSSA scheme, and then evaluates it in the specific operating environment. The final results prove that the scheme can achieve lightweight calculations for group members, and that LSSA has high security compared with similar audit schemes.

A. ANALYSIS OF CALCULATION OVERHEAD

For the convenience of the analysis, the following symbols are used to indicate the specific operations in the scheme: Mul_{G_1},Mul_{G_2}, and Mul_{Z_p} represent the multiplication time in G_1,G_2, and Z_p^*, respectively; Exp_{G_1} and Exp_{G_2} represent the exponential operation times in G_1 and G_2, respectively; Add_{Z_p} represents the addition time in Z_p^*, and $Pair$ represents the calculation time $e : G_1 \times G_1 \rightarrow G_2$ of a bilinear pair.

In the data upload phase, group members need to blind the data, the cost of which is $n \cdot s \cdot Add_{Z_p}$. In addition, $hash(id_{ij})$ needs to be calculated. The TPM needs to calculate the authentication label of the blind data, the cost of which is $(s+1)Exp_{G_1} + sMul_{G_1}$.

In the audit phase, the computational overhead of the challenge initiated by the TPM is $(s+2)Exp_{G_1}$. The TPM verifies the evidence from the cloud, where the computational overhead is $cExp_{G_1} + cMul_{G_1} + cPair + (d-1)Mul_{G_2} + 2Exp_{G_2}$.

B. EXPERIMENT RESULTS

The experiment was implemented using the Ubuntu 16.04 operation system with an Intel Core i7 3.4 GHz processor and an 8GB memory. The programme is written in *C*, and it uses the library functions in the Pairing-Based Cryptography (PBC) library to simulate the cryptographic operations, where the benchmark threshold is 512 bits, the size of the element is $|p|$ =160 bits in Z_p^*, and the size of the shared data is 20 MB. The experimental results are the averages of the 10 experiments.

1) OVERHEAD IN THE AUDIT PHASE

Due to the powerful computing power of the cloud, more attention is often paid to the overhead of the TPM in the process of a data integrity audit. According to Section V, we know that the auditing task of the TPM is divided into two phases: the challenge generation and proof verification. To effectively evaluate the auditing computation overhead of our scheme, we analyse the time cost of the two above phases. The experimental results are presented in Fig. 7. We choose to challenge different blocks from 100 to 500 in intervals of 50 and set the number of TPMs to $d = 20$. As shown in Fig. 7, The time overhead of the challenge phase is constant, which is approximately 31 ms. The running time of the proof verification ranges from 80 ms to 260 ms. In step 8, $H_2(l)$ ($l \in L_i$, where L has c elements) is linear with the value c. Thus, the computation overhead of the proof verification linearly increases with the number of the challenged blocks. As we know, a greater number of challenged blocks results in a more accurate integrity verification. However, this will incur more computational overhead. Thus, it is desirable to find a trade-off between auditing computational cost and integrity guarantee.

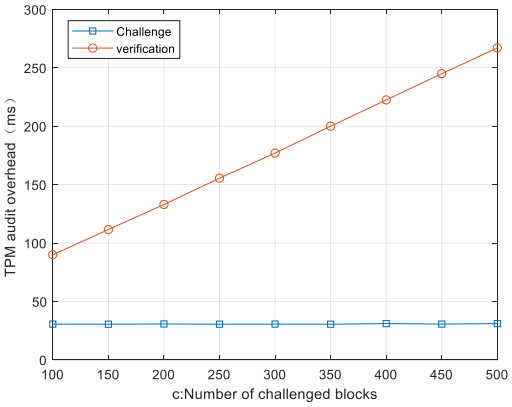

FIGURE 7. TPM audit overhead.

2) OVERHEAD IN THE DATA UPLOAD PHASE

(1) The time overhead of the hash algorithm

To test the time complexity of the hash algorithm, experiments are performed by calculating the hash values of different $id_{i,j}$ sizes. The tested hash algorithm selects the best current SHA-1 and SHA-256 algorithms. As shown in Fig. 8, as the size of $id_{i,j}$ increases, the time overhead also increases. When the size of $id_{i,j}$ increases from 0 to 14 bits, the time overhead of calculating the hash algorithm increases slowly. Therefore, in order to reduce the computational burden of the group members and group manager, the size of $id_{i,j}$ is set to 14 bits, which is enough to record the public identifier information of the shared data. When the id size is set to 14 bits, the time overheads of the two algorithms are 0.011 s and 0.012 s, respectively.

(2) Calculation overhead of authentication label generation

As shown in Fig. 9, the experimental results show that the time overhead of the authentication label generation increases linearly with the number of shared data slices s. When the data slice $s = 500$, the time overhead of the authentication label generation is 720 ms. Fig. 9 also shows that the time required for blinding data by group members is very small, and it therefore can be ignored.

We used a test to illustrate the effect of our scheme in a big data scenario. For each block size (from 1KB to 1000KB), we tested the overhead of the authentication label generation for

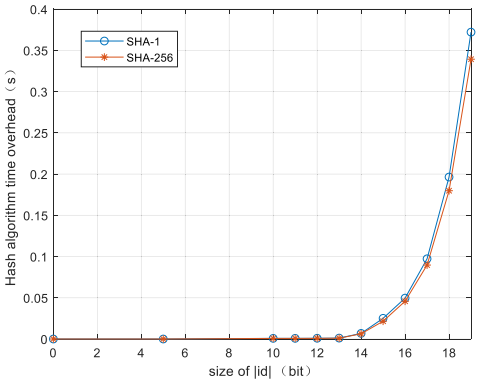

FIGURE 8. Time overhead of the hash algorithm.

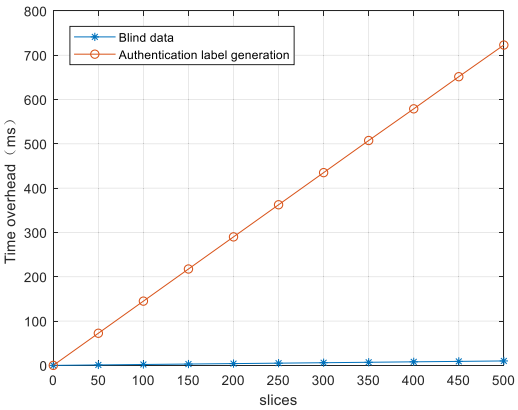

FIGURE 9. Calculation overhead of the blind data and the authentication label generation.

2 GB data, 20 GB data and 200 GB data. As shown in Table 2, the cost of the authentication label computation for the same file decreases almost linearly with an increase in the block size. According to our authentication label computation formula (5), we can see that each authentication label generation involves $(s + 2)$ exponentiations, which are the main overhead for computing the authentication label. For same size data, a larger block fragmentation means less authentication labels and therefore less exponentiations. However, in actual scenarios the group members usually divide a smaller block fragmentation to achieve a flexible operation of the cloud data, including block deletion, modification, and insertion. This increases the computational burden of the TPM. The scheme of this paper adopts the TPM management strategy. The group manager can allocate the computing tasks to each TPM, which can reduce the computational burden of a single TPM. As shown in Fig. 9, if 500 data slices are uploaded to d TPMs, the average computing overhead of each TPM is $720/d$ ms.

In summary, the experiment is evaluated using two better hash algorithms, and the optimal digits of the *id* are selected. At this time, the calculation overhead of the group member is the sum of the time overheads for the hash algorithm and for blinding the data, which is lightweight for group members. Moreover, we can find that multiple TPMs takes part in the calculation, thereby achieving the lightweight calculation of

TABLE 2. Authentication label generation time overhead with different block sizes.

Block size	1KB	10KB	100KB	1000KB
2GB data (s)	12562	1264	12	1
20GB data (s)	126530	12546	126	12
200GB data (s)	1253612	125960	1235	124

TABLE 3. Comparison table.

	1	2	3	4	5	6
scheme[15]	x	x	√	x	--	√
scheme[16]	x	√	√	x	--	x
scheme[17]	x	√	√	x	x	x
scheme[18]	x	√	√	x	x	√
LSSA	√	√	√	√	√	√

"√" means "support", "x" means "does not support", "--" means "not mentioned". Labels 1-6 represent the following properties: 1. Anti-replace/replay attack. 2. Data privacy. 3. Identity privacy. 4. Identity traceability. 5. Agent security. 6. Lightweight overhead.

a single TPM during the generation of authentication labels. Nevertheless, it is worth to note that the group size and the number of TPMs is specified by the group manager, which enables us to formulate flexible strategies to control the throughput according to the actual scenario, to achieve a better application of our scheme.

We describe a high-level comparison between LSSA and existing studies [15]–[18] as shown in Table 3. We can see that LSSA supports identity privacy, traceability, data privacy, agent security, a lightweight overhead, and it can resist the replace and replay attacks.

VIII. CONCLUSION

In this paper, we proposed a provable shared data possession for a lightweight and security audit process in cloud storage. By introducing a Hashgraph, the traceability of group membership is achieved, and the illegal behaviours of group members can be contained through Hashgraph technology. By specifying multiple TPMs for calculation and management according to the TPM management strategy, each group member and each TPM are independent of one another, which ensures that the cloud data verification process is secure and achieves a lightweight calculation of the TPM. Through a security analysis, the scheme in this paper can avoid replay attacks and replace attacks while protecting the identity privacy and data privacy of group members and ensuring secure storage of the shared data. Therefore, this scheme has important significance and value for the secure storage of shared data.

REFERENCES

[1] M. Armbrust *et al.*, "Above the clouds: A Berkeley view of cloud computing," Dept. Elect. Eng. Comput. Sci., Univ. California, Berkeley, Berkeley, CA, USA, Tech. Rep. UCB/EECS-2009-28, 2009. [Online]. Available: https://www2.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-28.pdf

[2] P. Mell and T. Grance, "The National Institute of Standards and Technology (NIST) denition of cloud computing," NIST, Washington, DC, USA, NIST Special Publication 800-145, 2011. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-145.pdf
[3] K. Julisch and M. Hall, "Security and control in the cloud," *Inf. Secur. J. Global Perspective*, vol. 19, no. 6, pp. 299–309, 2010.
[4] D. G. Feng, M. Zhang, Y. Zhang, and Z. Xu, "Study on cloud computing security," *J. Softw.*, vol. 22, no. 1, pp. 71–83, 2011.
[5] G. Ateniese *et al.*, "Provable data possession at untrusted stores," in *Proc. 14th ACM Conf. Comput. Commun. Secur. (CCS)*, 2007, pp. 598–609.
[6] A. Juels and B. S. Kaliski, "Pors: Proofs of retrievability for large files," in *Proc. 14th ACM Conf. Comput. Commun. Secur. (CCS)*, 2007, pp. 584–597.
[7] G. Ateniese, R. Di Pietro, L. V. Mancini, and G. Tsudik, "Scalable and efficient provable data possession," in *Proc. 4th Int. Conf. Secur. Privacy Commun. Netw. (ICST)*, Istanbul, Turkey, 2008, pp. 22–25.
[8] H. Shacham and B. Waters, "Compact proofs of retrievability," in *Proc. Int. Conf. Theory Appl. Cryptol. Inf. Secur.* Berlin, Germany: Springer, 2008, pp. 90–107.
[9] G. Yang, J. Yu, W. Shen, Q. Su, Z. Fu, and R. Hao, "Enabling public auditing for shared data in cloud storage supporting identity privacy and traceability," *J. Syst. Softw.*, vol. 113, pp. 130–139, Mar. 2016. doi: 10.1016/j.jss.2015.11.044.
[10] T. Jiang, X. Chen, and J. Ma, "Public integrity auditing for shared dynamic cloud data with group user revocation," *IEEE Trans. Comput.*, vol. 65, no. 8, pp. 2363–2373, Aug. 2016. doi: 10.1109/TC.2015.2389955.
[11] Y. Luo, M. Xu, K. Huang, D. Wang, and S. Fu, "Efficient auditing for shared data in the cloud with secure user revocation and computations outsourcing," *Comput. Secur.*, vol. 73, pp. 492–506, Mar. 2018. doi: 10.1016/j.cose.2017.12.004.
[12] L. Huang, G. Zhang, and A. Fu, "Privacy-preserving public auditing for dynamic group based on hierarchical tree," *J. Comput. Res. Develop.*, vol. 53, no. 10, pp. 2334–2342, 2016. doi: 10.7544/issn1000-1239.2016.20160429.
[13] L. X. Huang, G. M. Zhang, and A. M. Fu, "Certificateless public verification scheme with privacy-preserving and message recovery for dynamic group," in *Proc. Australas. Comput. Sci. Week Multiconf.*, Melbourne, VIC, Australia, 2017, p. 76.
[14] A. Fu, S. Yu, Y. Zhang, H. Wang, and C. Huang, "NPP: A new privacy-aware public auditing scheme for cloud data sharing with group users," *IEEE Trans. Big Data*, to be published. doi: 10.1109/TBDATA.2017.2701347.
[15] J. Li, X. Tan, X. Chen, D. S. Wong, and F. Xhafa, "OPoR: Enabling proof of retrievability in cloud computing with resource-constrained devices," *IEEE Trans. Cloud Comput.*, vol. 3, no. 2, pp. 195–205, Apr. 2015. doi: 10.1109/TCC.2014.2366148.
[16] C. W. Guan, K. Ren, F. Zhang, F. Kerschbaum, and J. Yu, "Symmetric-key based proofs of retrievability supporting public verification," in *Proc. 20th Eur. Symp. Comput. Secur.* Berlin, Germany: Springer-Verlag, 2015.
[17] H. Wang, D. He, and S. Tang, "Identity-based proxy-oriented data uploading and remote data integrity checking in public cloud," *IEEE Trans. Inf. Forensics Security*, vol. 11, no. 6, pp. 1165–1176, Jun. 2016. doi: 10.1109/TIFS.2016.2520886.
[18] W. Shen, J. Yu, H. Xia, H. Zhang, X. Lu, and R. Hao, "Light-weight and privacy-preserving secure cloud auditing scheme for group users via the third party medium," *J. Netw. Comput. Appl.*, vol. 82, pp. 56–64, 2017. doi: 10.1016/j.jnca.2017.01.015.
[19] J. Heintz, "On the computational complexity of polynomials and bilinear mappings. A survey," in *Proc. Int. Conf. Appl. Algebra, Algebraic Algorithms, Error-Correcting Codes*. Berlin, Germany: Springer, 1987, pp. 269–300.
[20] *Encyclopedia of Cryptography and Security*. Accessed: Jul. 7, 2018. [Online]. Available: https://link.springer.com/referenceworkentry/10.1007%2F978-1-4419-5906-5_882
[21] W. M. Zheng and Z. Z. Tang, "Interconnection function," in *Computer System Architecture*, 3rd ed. Beijing, China: Tsinghua Univ. Press, 1999.
[22] *Hashgraph Consensus: Detailed Examples*. Accessed: Sep. 20, 2018. [Online]. Available: https://www.swirlds.com/swirlds-hashgraph-consensus-identity-leemon-baird/
[23] A. S. Tanenbaum and D. J. Wetherall, "TCP sliding window,"in Computer Networks, 5th ed. China: Mechanical Industry Press, 2011.
[24] J. Shen, J. Shen, X. Chen, X. Huang, and W. Susilo, "An efficient public auditing protocol with novel dynamic structure for cloud data," *IEEE Trans. Inf. Forensics Security*, vol. 12, no. 10, pp. 2402–2415, Oct. 2017.
[25] *The Swirld Hashgraph Consensus Algorithm: Fair, Fast, Byzantine Fault Tolerance*. Accessed: Aug. 28, 2018. [Online]. Available: https://www.swirlds.com/swirlds-hashgraph-consensus-identity-leemon-baird/

JUNFENG TIAN received the degree from Hebei University, in 1986, the master's degree in computer science from the Xi'an University of Electronic Science and Technology, in 1991, the master's degree, in 1995, and the degree from the University of Science and Technology of China, with a major in computer science and technology, in 2004. He is currently a Professor, the Deputy Secretary-General, the Open System Specialized Committee of China Computer Federation, the Young and Middle-aged Specialist with Outstanding Contribution in Hebei Province, and the Young and Middle-aged Teacher with Outstanding Contribution in Hebei Province. He has been involved in teaching and research in the fields of distributed computing and network technology for many years.

XUAN JING was born in 1994. He is currently pursuing the master's degree. His main research interests include the security of information and trusted compute and distributed computation. He is also a CCF student member.