Lightweight Secure Cloud Auditing Scheme for Shared Data Supporting Identity Privacy and Traceability

Jun-Feng Tian[1,2], Xuan Jing[1,2](✉), and Rui-Fang Guo[1,2]

[1] School of Cyberspace Security and Computer Institute, Hebei University, Baoding 071000, China
abidble@gmail.com

[2] Hebei Key Laboratory of High Confidence Information Systems, Hebei University, Baoding 071000, China

Abstract. Cloud platform provides users with shared data storage services. To ensure shared data integrity, it is necessary to validate the data effectively. The audit scheme that supports the group dynamic operations conducts the integrity verification of the shared data, but this approach results in complex calculations for group members. The audit scheme of the designated agent implements the lightweight calculation of 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 cloud auditing scheme for shared data supporting identity privacy and traceability (LSSA) is proposed, which realizes the security management of dynamic groups and the lightweight calculations for group members. Meanwhile, a virtual TPM pool is constructed by combining TCP sliding window technology and interconnected functions to improve agent security. Experiments on real data sets show that the theoretical analysis and experimental results are consistent, thereby reflecting the feasibility and efficiency of the scheme.

Keywords: Shared data · Dynamic groups · Lightweight calculation · Agent security

1 Introduction

Users can easily communicate with one another on cloud platforms to share data. Data sharing means that a user in a group uploads shared data to the cloud, and the rest of the group can access the shared data. Many cloud storage service providers (e.g., iCloud, OneDrive, and Baidu Cloud) currently use cloud data sharing as one of their main services. However, there are some data damage threats in the cloud, such as unavoidable hardware failures, external attacks, and cloud service provider damage. Therefore, in order to verify the integrity of the data stored in the cloud, Ateniese *et al.* first proposed a provable data possession (PDP) model, which can verify the integrity of cloud data without retrieving all data [1].

To support effective group dynamic operations, some researchers have proposed some audit schemes for shared data based on the PDP model [2–4], where group dynamics operations refer to supporting effective changes, additions, and deletions of group members. Yang *et al.* used the IDL table to record the use of shared data by group members and achieved the traceability of group members [2]. Jiang et al. adopted the vector commitment technique, and the verifier removed the illegal group members. This technique achieved the purpose of resisting collusion attacks of the cloud service provider and the group members [3]. Fu *et al.* proposed an audit scheme that can restore the latest correct shared data blocks by changing the binary tree tracking data and implementing homomorphic authentication in the group [4].

In the above scheme [1–4], in order to verify the integrity of the data stored in the cloud, the group members need to block the data and calculate the authentication label before uploading the data, but the computational burden is large. Guan *et al.* used an indistinguishable confusing approach to build an audit scheme for cloud storage [5], thereby reducing the time that is required to generate authentication label 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 [6], which alleviated the computational burden for group members. However, in order to guarantee data privacy, blinding data is needed before each data upload, which inevitably increases the computational burden. Shen *et al.* realized the lightweight calculation of group members by introducing an agent to replace group members for generating authentication label [7], but it failed to consider the security risks that may occur when malicious group members collude with agents to obtain group keys.

Through the above analysis, how to ensure lightweight computing costs for group members while realizing the group dynamic operations is a problem to be solved in the current audit scheme of shared data. Considering the problem of improving the security and lightweight computing for agents, a lightweight secure cloud auditing scheme for shared data supporting identity privacy and traceability (LSSA) is proposed. The main contributions of this paper are as follows.

(1) By introducing a Hashgraph, the traceability of group membership can be guaranteed, and then effective changes, additions, and deletions of group members can be realized.
(2) The Third Party Medium (TPM) management strategy is designed, and the virtual TPM pool is built by the group manager. The strategy ensures the data privacy and identity privacy of group members and prevents the TPM from leaking group keys. This strategy also realizes lightweight calculations for a single TPM. Using the TPM instead of group members to calculate the authentication label and audit data integrity realizes the lightweight calculations for group members.
(3) The security analysis of the scheme shows that the scheme is safe and can resist replace attacks and replay attacks.

2 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 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 cloud provides data storage services for group members. The TPM replaces group members to calculate the authentication label and audit the integrity of shared data.

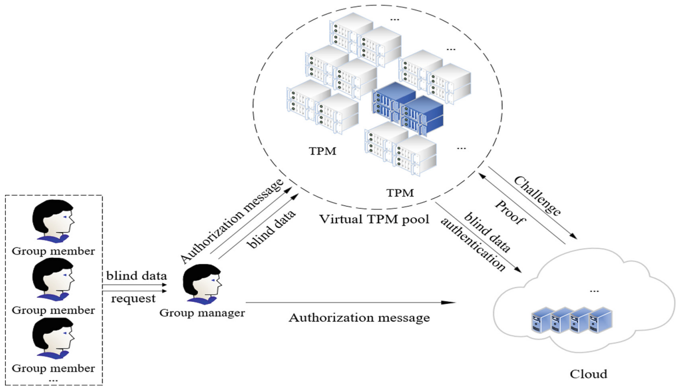


Fig. 1. System model

Group members need to share blind data and send requests to the GM to upload blind data. According to the request of group members, the GM chooses the TPM and the processing time for the blind data. The GM formulates the selection strategy (Sect. 3.2 for details) and builds a virtual TPM pool through the selection strategy, thereby making it difficult for entities other than the GM to know the TPM that is performing a computational task. This approach avoids the threats of attackers and prevents malicious group members from obtaining the TPM's private key. The GM authorizes the selected TPM and sends the blind data to it. The TPM calculates the corresponding authentication label based on the received blind data and uploads the blind data and authentication label to the cloud. The cloud verifies the TPM's identity information according to the authorization information sent by the GM. After verification, the real data and authenticated label are recovered from the blind data and the corresponding authentication label and stored. When the GM wants to verify the integrity of the cloud data, the GM selects the TPM according to the selection strategy, and then the TPM challenges the cloud. After receiving this challenge, the cloud returns the evidence of shared data to the TPM. Finally, the TPM verifies the correctness of the evidence to judge the integrity of the shared data.

3 The Main Design Ideal of LSSA

In this section, we use the Hashgraph technology to propose the design idea of group member management. By referring to the TCP sliding window [8] and using the Interconnection function [9], the design idea of the TPM management strategy is proposed.

3.1 Design Idea of Group Member Management

Assume that the data files are divided into n blocks (m1, m2, … mn), each data block mi is fragmented into s sectors (mi,1, mi,2, … mi,s), and blind data can be represented by $\mathrm{m}'_{i,j}(1 \leq i \leq n, 1 \leq j \leq s)$. As shown in Fig. 2, before the data owner MOwner sends the blind data block $\mathrm{m}'_{i,j}$ to the group manager, it calculates the hash value hash(idi,j) of idi,j (idi,j is the public identifier information of the blind data block $\mathrm{m}'_{i,j}$) as the transaction record of the initial event and attaches the signature SignMOwner. According to the Hashgraph technique [10], it randomly selects the group member or group manager to synchronize it 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 Mi that have modified and accessed $\mathrm{m}'_{i,j}$ since then need to update the identifier of the blind block after use. Thus, the members calculate the hash value of idi,j as a transaction record for a new event and attach the signature SignMi to spread it within the group.

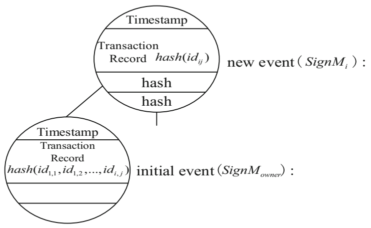

Timestamp
Transaction Record $hash(id_{ij})$
hash
hash
new event ($SignM_i$) :
Timestamp
Transaction Record
$hash(id_{1,1}, id_{1,2}, \ldots, id_{i,j})$
initial event ($SignM_{owner}$) :

Fig. 2. Event diagram

3.2 Design Idea of 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 Mi 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. 3, the sending window has 3 pointers that slide clockwise. The window that allows u1 to send is the time between P1 and P3, and the current time is P2. Suppose that the group manager divides a certain time period into 20 parts as t1–t20. When group member M1 sends an application at t8, it is just in the allowable sending time of the sending window, and then the group manager receives the sent application from M1. According to the time frame rotation, if group member M1 does not send the request in the allowed sending time, P2 slides clockwise to $\Delta t_1'$, which corresponds to M_1*'s* port number address u_1, and then $\Delta t_1'$ is the time to process the request of M_1.

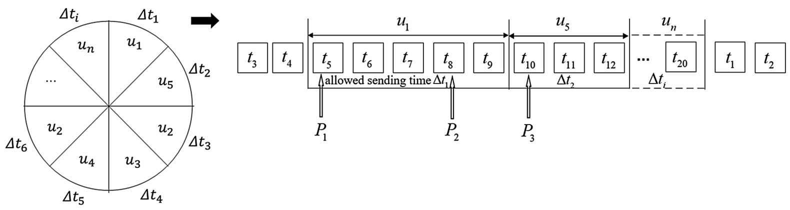


Fig. 3. Sending window

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

As shown in Fig. 4, 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, the round is transferred to the time period Δt_3, and the group manager calculates the output at the moment through u_2, Δt_3 and the interconnection function C_0. At that moment, the correspondence between the input and output can be represented by a matrix

$$\begin{array}{c|c} & \begin{array}{cccc} u_1 & u_2 & u_3 & u_4 \end{array} \\ \hline \begin{array}{c} TPM_{1\Delta t_3} \\ TPM_{2\Delta t_3} \\ TPM_{3\Delta t_3} \\ TPM_{4\Delta t_3} \end{array} & \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \end{array}$$

$(i = 4)$, which indicates that the group manager selects the output address TPM_1 through u_2 and C_0 at Δt_3.

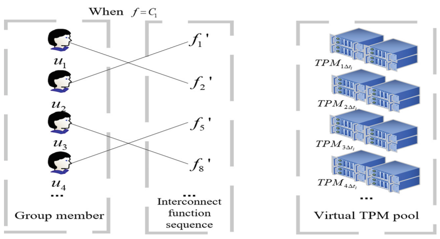
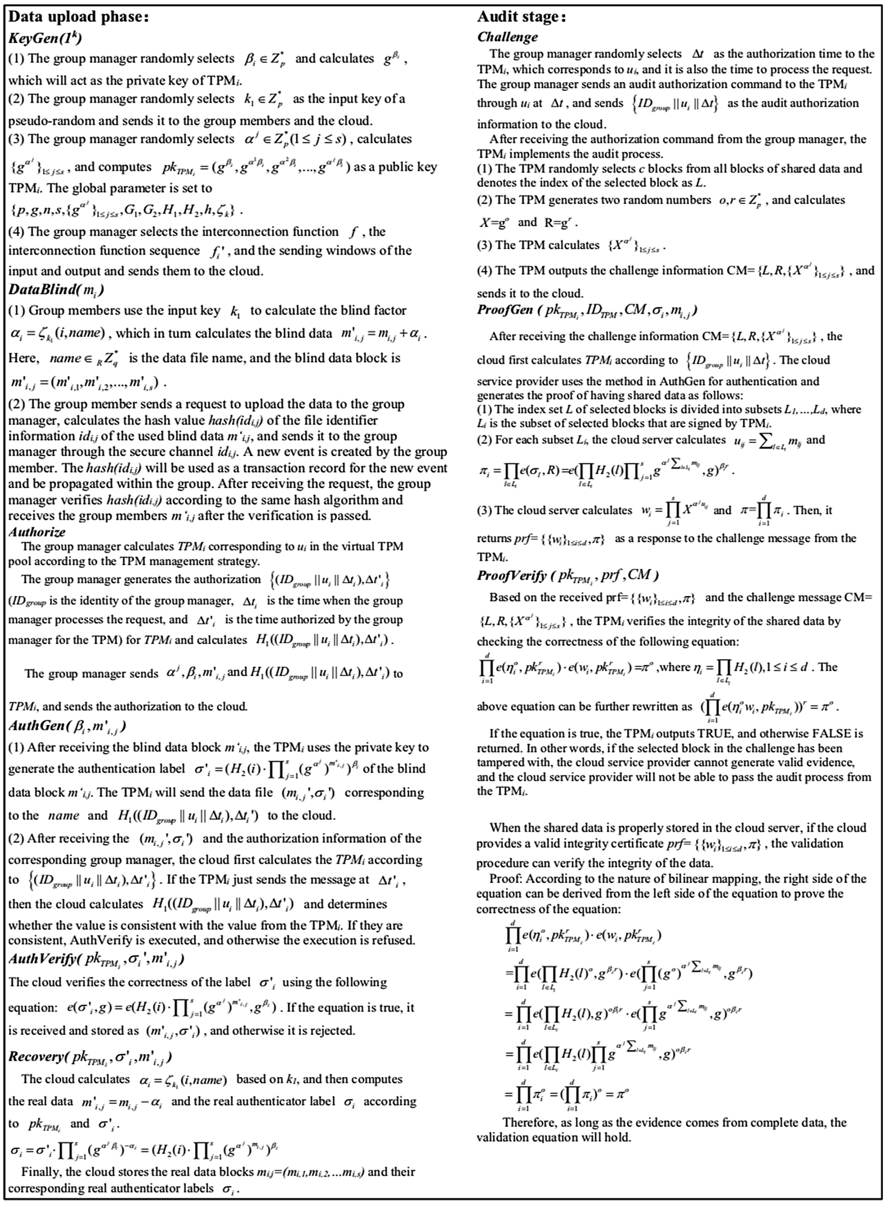


Fig. 4. Virtual TPM pool construction diagram

4 Detailed Description of the LSSA Scheme

The LSSA scheme consists of nine algorithms: *KeyGen*, *DataBlind*, *Authorize*, *AuthGen*, *AuthVerify*, *Recovery*, *Challenge*, *ProofGen*, and *ProofVerify*. The parameters used in the algorithm are as follows (Fig. 5).



Data upload phase:

KeyGen(1^k)

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

(2) The group manager randomly selects $k_1 \in Z_p^*$ as the input key of a pseudo-random and sends it to the group members and the cloud.

(3) The group manager randomly selects $\alpha^j \in Z_p^*(1 \le j \le s)$, calculates $\{g^{\alpha^j}\}_{1\le j\le s}$, and computes $pk_{TPM_i} = (g^{\beta_i}, g^{\alpha^1\beta_i}, g^{\alpha^2\beta_i}, ..., g^{\alpha^j\beta_i})$ as a public key TPM$_i$. The global parameter is set to $\{p, g, n, s, \{g^{\alpha^j}\}_{1\le j\le s}, G_1, G_2, H_1, H_2, h, \zeta_k\}$.

(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 sends them to the cloud.

DataBlind(m_i)

(1) Group members use the input key k_1 to calculate the blind factor $\alpha_i = \zeta_{k_1}(i, name)$, which in turn calculates the blind data $m'_{i,j} = m_{i,j} + \alpha_i$. Here, $name \in {}_R Z_q^*$ is the data file name, and the blind data block is $m'_{i,j} = (m'_{i,1}, m'_{i,2}, ..., m'_{i,s})$.

(2) The group member sends a request to upload the data to the group manager, calculates the hash value *hash(id$_{i,j}$)* of the file identifier information *id$_{i,j}$* of the used blind data *m'$_{i,j}$*, and sends it to the group manager through the secure channel *id$_{i,j}$*. A new event is created by the group member. The *hash(id$_{i,j}$)* will be used as a transaction record for the new event and be propagated within the group. After receiving the request, the group manager verifies *hash(id$_{i,j}$)* according to the same hash algorithm and receives the group members *m'$_{i,j}$* after the verification is passed.

Authorize

The group manager calculates *TPM$_i$* corresponding to *u$_i$* in the virtual TPM pool according to the TPM management strategy.

The group manager generates the authorization $\{(ID_{group} \| u_i \| \Delta t_i), \Delta t'_i\}$ (*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) for *TPM$_i$* and calculates $H_1((ID_{group} \| u_i \| \Delta t_i), \Delta t'_i)$.

The group manager sends $\alpha^j, \beta_i, m'_{i,j}$ and $H_1((ID_{group} \| u_i \| \Delta t_i), \Delta t'_i)$ to *TPM$_i$*, and sends the authorization to the cloud.

AuthGen($\beta_i, m'_{i,j}$)

(1) After receiving the blind data block *m'$_{i,j}$*, the TPM$_i$ uses the private key to generate the authentication label $\sigma'_i = (H_2(i) \cdot \prod_{j=1}^{s} (g^{\alpha^j})^{m'_{i,j}})^{\beta_i}$ of the blind data block *m'$_{i,j}$*. The TPM$_i$ will send the data file $(m_{i,j}', \sigma_i')$ corresponding to the *name* and $H_1((ID_{group} \| u_i \| \Delta t_i), \Delta t_i')$ to the cloud.

(2) After receiving the $(m_{i,j}', \sigma_i')$ and the authorization information of the corresponding group manager, the cloud first calculates the *TPM$_i$* according to $\{(ID_{group} \| u_i \| \Delta t_i), \Delta t'_i\}$. If the 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 from the TPM$_i$. If they are consistent, AuthVerify is executed, and otherwise the execution is refused.

AuthVerify($pk_{TPM_i}, \sigma_i', m'_{i,j}$)

The cloud verifies the correctness of the label σ'_i using the following equation: $e(\sigma'_i, g) = e(H_2(i) \cdot \prod_{j=1}^{s} (g^{\alpha^j})^{m'_{i,j}}, g^{\beta_i})$. If the equation is true, it is received and stored as $(m'_{i,j}, \sigma'_i)$, and otherwise it is rejected.

Recovery($pk_{TPM_i}, \sigma'_i, m'_{i,j}$)

The cloud calculates $\alpha_i = \zeta_{k_1}(i, name)$ based on *k$_1$*, and then computes the real data $m'_{i,j} = m_{i,j} - \alpha_i$ and the real authenticator label σ_i according to pk_{TPM_i} and σ'_i.

$$\sigma_i = \sigma'_i \cdot \prod_{j=1}^{s} (g^{\alpha^j\beta_i})^{-\alpha_i} = (H_2(i) \cdot \prod_{j=1}^{s} (g^{\alpha^j})^{m_{i,j}})^{\beta_i}$$

Finally, the cloud stores the real data blocks *m$_{i,j}$=(m$_{i,1}$,m$_{i,2}$,...m$_{i,s}$)* and their corresponding real authenticator labels σ_i.

Audit stage:

Challenge

The group manager randomly selects Δt as the authorization time to the TPM$_i$, which corresponds to *u$_i$*, and it is also the time to process the request. The group manager sends an audit authorization command to the TPM$_i$ through *u$_i$* at Δt, and sends $\{ID_{group} \| u_i \| \Delta t\}$ 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) The TPM randomly selects *c* blocks from all blocks of shared data and denotes the index of the selected block as *L*.

(2) The TPM generates two random numbers $o, r \in Z_p^*$, and calculates X=g^o and R=g^r.

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

(4) The TPM outputs the challenge information CM=$\{L, R, \{X^{\alpha^j}\}_{1\le j\le s}\}$, and sends it to the cloud.

ProofGen ($pk_{TPM_i}, ID_{TPM}, CM, \sigma_i, m_{i,j}$)

After receiving the challenge information CM=$\{L, R, \{X^{\alpha^j}\}_{1\le j\le s}\}$, the cloud first calculates *TPM$_i$* according to $\{ID_{group} \| u_i \| \Delta t\}$. The cloud service provider uses the method in AuthGen for authentication and generates the proof of having shared data as follows:

(1) The index set *L* of selected blocks is divided into subsets *L$_1$,...,L$_d$*, where *L$_i$* is the subset of 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 $\pi_i = \prod_{l\in L_i} e(\sigma_l, R) = e(\prod_{l\in L_i} H_2(l) \prod_{j=1}^{s} g^{\alpha^j \sum_{l\in L_i} m_{lj}}, g)^{\beta_i r}$.

(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*=$\{\{w_i\}_{1\le i\le d}, \pi\}$ as a response to the challenge message from the TPM$_i$.

ProofVerify (pk_{TPM_i}, prf, CM)

Based on the received prf=$\{\{w_i\}_{1\le i\le d}, \pi\}$ and the challenge message CM=$\{L, R, \{X^{\alpha^j}\}_{1\le j\le s}\}$, the TPM$_i$ verifies the integrity of the shared data by checking the correctness of the following equation:

$\prod_{i=1}^{d} e(\eta_i^o, pk_{TPM_i}^r) \cdot e(w_i, pk_{TPM_i}^r) = \pi^o$, 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, the TPM$_i$ outputs TRUE, and 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$_i$.

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 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 the equation:

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

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

Fig. 5. Algorithm diagram

5 Security Analysis

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

5.1 Audit Security

The malicious cloud service provider cannot complete the audit process through replace attacks and replay attacks. ① Since each time the TPM initiates a challenge both $L = \{L_1, \ldots, L_d\}$ and $X = g^{\circ}$ 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 cannot implement the replace attack. ② Cloud service providers must calculate η° $(\eta = \prod_{l \in L} H_2(l) \in G_1)$ 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 $g^{\psi \circ}$ based on g^{ψ}, g and g°.

5.2 Data Privacy and Identity Privacy

① In the user upload data phase, the TPM cannot extract the real data $m_{i,j}$ through the blind data block $m'_{i,j}$. This finding is observed because the TPM receives $m'_{i,j} = m_{i,j} + \alpha_i (j \in [1, s])$, 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 through randomly distributed blind data blocks $m'_{i,j}$.

5.3 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 key for some reasons. ① A malicious group member may collude with the TPM to obtain a private key. To this end, the group manager specifies multiple TPMs, 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 reasons. By constructing a virtual TPM pool, only the group manager can calculate TPM$_i$, and those outside cannot find the target of the attack.

5.4 Identity Traceability

Through Swirld's Hashgraph Consistency Algorithm [11], 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 as an illegal group member.

6 Summary

By introducing a Hashgraph, a group manager can flexibly register or remove group members and achieve the traceability of group membership. By specifying multiple TPMs for calculation and management according to the TPM management strategy, each group member and each TPM are independent of each other, which ensures the secure calculations of the TPM and realizes the lightweight calculation of the TPM. Through security analysis, the scheme of this paper can avoid replay attacks and replay attacks while protecting the identity privacy and data privacy of group members and ensuring secure storage of shared data.

References

1. Ateniese, G., Burns, R., Curtmola, R., et al.: Provable data possession at untrusted stores. In: ACM Conference on Computer and Communications Security, pp. 598–609. ACM (2007)
2. Yang, G., Yu, J., Shen, W., et al.: Enabling public auditing for shared data in cloud storage supporting identity privacy and traceability. J. Syst. Softw. **113**(C), 130–139 (2016)
3. Luo, Y., Xu, M., Huang, K., et al.: Efficient auditing for shared data in the cloud with secure user revocation and computations outsourcing. Comput. Secur. **73**, 492–506 (2018)
4. Fu, A., Yu, S., Zhang, Y., et al.: NPP: a new privacy-aware public auditing scheme for cloud data sharing with group users. IEEE Trans. Big Data **PP**(99), 1 (2017)
5. Guan, C., Ren, K., Zhang, F., Kerschbaum, F., Yu, J.: Symmetric-key based proofs of retrievability supporting public verification. In: Pernul, G., Ryan, P.Y.A., Weippl, E. (eds.) ESORICS 2015. LNCS, vol. 9326, pp. 203–223. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-24174-6_11
6. Wang, H., He, D., Tang, S.: Identity-based proxy-oriented data uploading and remote data integrity checking in public cloud. IEEE Trans. Inf. Forensics Secur. **11**(6), 1165–1176 (2016)
7. Shen, W., Yu, J., Xia, H., et al.: Light-weight and privacy-preserving secure cloud auditing scheme for group users via the third party medium. J. Netw. Comput. Appl. **82**, 56–64 (2017)
8. Xie, X.: Computer Networks. Publishing House of Electronics Industry, Beijing (2008)

9. Zheng, W., Tang, Z.: Computer System Architecture. Tsinghua University Press, Beijing (1999)
10. Baird, L.: Hashgraph consensus: detailed examples (2016)
11. Baird, L.: The Swirld Hashgraph Consensus algorithm: fair, Fast, Byzantine Fault Tolerance (2016)