A Secured Framework for Privacy-Preserving Auditing for Data Storage in Cloud Computing

Muppineni Renuka[1], Alabazar Ramesh[2], G. Nikhil Sai[3], G. Chaitanya[4], Y. Praveen Kumar[5]

1 Student, Dept Of CSE, Qis College of Engineering and Technology, Ongole, Prakasam (Dt)
2 Assistant professor, Dept Of CSE, Qis College of Engineering and Technology, Ongole, Prakasam (Dt)
3 Student, Dept Of CSE, Qis College of Engineering and Technology, Ongole, Prakasam (Dt)
4 Student, Dept Of CSE, Qis College of Engineering and Technology, Ongole, Prakasam (Dt)
5 Student, Dept Of CSE, Qis College of Engineering and Technology, Ongole, Prakasam (Dt)

Abstract: With the popularity of cloud computing, mobile devices can store/retrieve personal data from anywhere at any time. Consequently, the data security problem in mobile cloud becomes more and more severe and prevents further development of mobile cloud. There are substantial studies that have been conducted to improve the cloud security. However, most of them are not applicable for mobile cloud since mobile devices only have limited computing resources and power. Solutions with low computational overhead are in great need for mobile cloud applications. In this paper, we propose a lightweight data sharing scheme (LDSS) for mobile cloud computing. It adopts CP-ABE, an access control technology used in normal cloud environment, but changes the structure of access control tree to make it suitable for mobile cloud environments. LDSS moves a large portion of the computational intensive access control tree transformation in CP-ABE from mobile devices to external proxy servers. Furthermore, to reduce the user revocation cost, it introduces attribute description fields to implement lazy-revocation, which is a thorny issue in program based CP-ABE systems. The experimental results show that LDSS can effectively reduce the overhead on the mobile device side when users are sharing data in mobile cloud environments.

Keywords: Shared data Dynamic groups Lightweight calculation Agent security

I. INTRODUCTION

With the event of cloud computing and therefore the popularity of smart mobile devices, people are gradually getting familiar with a replacement era of knowledge sharing model during which the info is stored on the cloud and therefore the mobile devices are wont to store/retrieve the info from the cloud. Typically, mobile devices only have limited space for storing and computing [1] power. On the contrary, the cloud has enormous amount of resources. In such a scenario, to realize the satisfactory performance, it's essential to use the resources provided by the cloud service provider (CSP) to store and share the info.

Nowadays, various cloud mobile applications are widely used. In these applications, people (data owners) can upload their photos, videos, documents and other files to the cloud and share these data with people (data users) they wish to share. CSPs also provide data management functionality [2] for data owners. Since personal data files are sensitive, data owners are allowed to settle on whether to form their data files public or can only be shared with specific data users. Clearly, data privacy of the private sensitive data may be a big concern for several data owners.

The state-of-the-art privilege management /access control mechanisms provided by the CSP are either not sufficient or not very convenient. They can't meet all the requirements [3], [4] of knowledge owners. First, when people upload their data files onto the cloud, they're leaving the info during a place where is out of their control, and therefore the CSP may spy on user data for its commercial interests and/or other reasons. Second, people need to send password to every data user if they only want to share the encrypted data with certain users, which is extremely cumbersome. To simplify the privilege management, the info owner can divide data users into different groups and send password to the groups which they need to share the info. However, this approach requires fine-grained access control. In both cases, password management may be a big issue.

Apparently, to unravel the above problems, personal sensitive data should be encrypted before uploaded onto the cloud in order that the info is secure against the CSP. However, the info encryption brings new problems. The way to provide efficient access control mechanism on ciphertext decryption in order that only the authorized users can access the plaintext data is challenging. Additionally, system must offer data owner's effective user privilege management capability, in order that they can grant/revoke data access privileges easily on the info users. There are substantial researches on the difficulty of knowledge access control over ciphertext. In these researches, they need the subsequent common assumptions. First, the CSP is taken into account honest and curious. Second, all the sensitive data are encrypted before uploaded to the Cloud. Third, user authorization on certain data is achieved through encryption/decryption key distribution. Generally, we will divide these approaches into four categories: simple ciphertext access control, hierarchical access control, access control supported Fully homomorphism encryption [1][2] and access control supported attribute-based encryption (ABE). Of these proposals are designed for non-mobile cloud environment [16]. They consume large amount of storage and computation resources, which are not available for mobile devices. Consistent with the experimental leads to [19], [20], the essential ABE operations take much longer time on mobile devices than laptop or desktop computers. It's a minimum of 27 times longer to execute on a sensible phone than a private computer (PC).

To address this issue, during this paper, we propose a Lightweight Data Sharing Scheme (LDSS) for mobile cloud computing environment. The main contributions of LDSS are as follows:

(1) We design an algorithm called LDSS-CP-ABE based on Attribute-Based Encryption (ABE) method to supply efficient access control over ciphertext.

(2) We use proxy servers for encryption and decryption operations. In our approach, computational intensive operations in ABE are conducted on proxy servers, which greatly reduce the computational overhead on client side mobile devices. Meanwhile, in LDSS-CP-ABE, so as to take care of data privacy, a version attribute is additionally added to the access structure. The decryption key format is modified in order that it is often sent to the proxy servers during a secure way.

(3) We introduce lazy re-encryption and outline field of attributes to scale back the revocation overhead when dealing with the user revocation problem.

The rest of this paper is organized as follows. Section 2 presents some fundamental concepts in secure mobile cloud data sharing and therefore the security premise. Section 3 gives the detailed design of LDSS. Section 4 and 5 give the safety assessment and performance evaluation, respectively. Section 6 presents related works. Finally, Section 7 concludes our work with the longer term work.

II. RELATED WORK

In 2007, Ateniese et al. first proposed a Provable Data Possession (PDP) model, which may verify the integrity of cloud data without retrieving all of the info [5]. Then, Juels et al. proposed the Proofs of Retrievability (POR) scheme, which enables a back-up or archive service to supply proof that the data are often retrieved by the verifier [6]. During a subsequent study, Ateniese et al. implemented a PDP scheme that supports dynamic operations [7], which suggests that the info 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 subsequent schemes [9] [14] specialize in the way 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 isn't only ready 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 within the group [9]. Jiang et al. proposed data integrity supported the vector commitment technique, which is immune to collusion attacks of a cloud service provider and a gaggle 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 supported the logical hierarchy tree, thereby protecting the identity privacy of the group members [12]. Huang et al. subsequently proposed a certificate less 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 which will restore the newest correct shared data blocks by changing the binary tree tracking data within the group [14].

In the above scheme [9][14], so as to verify the integrity of the shared data stored within the cloud, the group members got to block the info then calculate the info authentication label for every block. Finally, the group member uploads the shared data along side the corresponding authentication labels to the cloud. The integrity verification of the shared data relies on the correctness of those data authentication labels. However, the value of calculating the authentication label is usually great, because the formula requires an outsized number of exponentiations, e.g., when the block size is 2 KB, the authentication label generation over- head for a ten GB le is almost 18 hours. Therefore, it is necessary to propose a light-weight auditing scheme to scale back the resource utilization of users. Li et al. proposed a replacement cloud storage auditing scheme with a cloud audit server and a cloud storage server [15].

Motivation

A malicious cloud server is in a position to discard all the shared data and generate a legitimate proof of knowledge possession by reserving some intermediate results or a previous valid proof, which we ask as a replace attack and a replay attack, respectively. A malicious group member is in a position to switch other member's data therein group without being discovered. A malicious agent is in a position to collude with illegal group members to steal user data and identity information. As far as we all 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). Almost like the cloud storage audit scheme [18], using the Third Party Medium (TPM) rather than group members to calculate the authentication label and audit data integrity leads to lightweight calculations for the group members. Additionally, the use of our proposed auditing processes, which is free from the replay and replace attacks, mentioned above, makes the auditing of our scheme more secure. Our research contributions are often summarized as follows:

[1] By introducing an efficient blind method, this paper ensures the info 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 meant, and the virtual TPM pool is made by the group manager. The strategy ensures the safety of agent (TPM) and leads to lightweight calculations for the agent. Using the TPM rather than group members to calculate the authentication label and audit data integrity leads to lightweight calculations for the group members.

III. SYSTEM MODEL AND STYLE 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 therefore the TPM. As shown in Fig. 1, there are multiple group members during a group. After the info owner (the individual or organization that owns the first data) creates the info le and uploads it to the cloud, any group member can access and modify the corresponding shared data. Note that the first data owner can play the role of GM and there's just one GM in each group. The Mplay two important roles: 1) blind data, and 2) record blind data and broadcast within the group through a Hashgraph. The cloud (e.g., iCloud, One Drive, 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's wont to blind the info for group members and to recover the important data for the cloud. The TPM plays two important roles: 1) generate data authentication label for

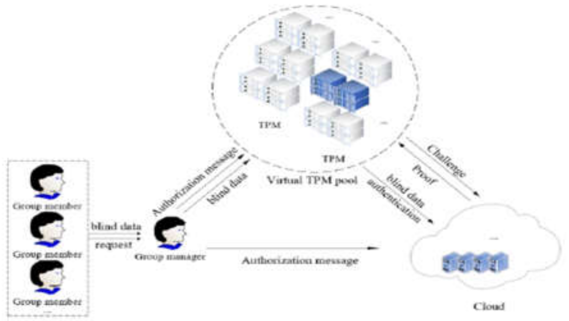


Figure 1: System Model

group members and 2) verify the integrity of the cloud data on behalf of the group members.

The execution procedure is split into the info upload stage and therefore 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 key seed and recorded by the Hashgraph, then sent to the group manager. Consistent with the TPM management strategy, the group manager selects a TPM from the virtual TPM pool (Section IV.B for details) for authorization, and therefore the authorized TPM calculates the corresponding authentication labels for these blinded data within the authorization time. 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 consistent with the TPM management strategy. Then, the authorized TPM sends the challenge messages to the cloud.

B. Design Goals

(1) Lightweight computing: This approach ensures that group members don't have to perform time-consuming calculations during the generation of authentication labels or during the audit of the shared data. Multiple TPMs participate within the calculation, thereby ensuring a light-weight calculation of a single TPM.

(2) Identity traceability: The modification of knowledge by illegal group members may cause disputes among the group members using an equivalent shared data. This goal ensures that the GM can find and take away any illegal group members, thereby achieving the safety management of groups.

(3) TPM management security: Each TPM works independently to make sure legal participation of the TPM. This goal ensures that the cloud only accepts and stores the info 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 rather than group members, it is impossible to understand the particular information of the info 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.

Main Design Idea of the Lssa

In this section, we use Hashgraph technology to propose the design idea of group member management. By pertaining to the TCP window [20] and using the Interconnection function, the planning 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 because the identity label of member M of the group, where consistent with the account, the particular identities of the group member are often determined. When the group member is removed, the group manager removes the account. For notational simplicity, we did the subsequent notations.

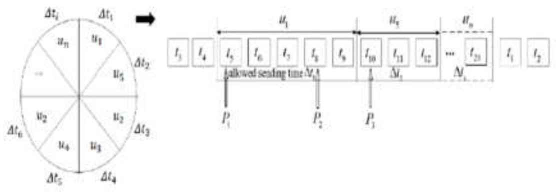


Figure 2: Sending Window

m: the info les are divided into n blocks (m1, m2,. . .mn), and every block is represented by mi, where m_i is fragmented into s slices (m_i,1, m_i,2,. . .m_i, s), and each slice is represented by m_{ij}.

As shown in Fig. 2, before the data owner M_{Owner} sends the blind data block m_{ij} to the group manager, who calculates the hash value $hash(idi, j)$ of $id_{i,j}$ as the upload record (called the transaction record) of the initial event and attaches the signature *Sign MOwner* . 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 *Mi 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 *idi*;*j* as a modify/access record (called a transaction record) for a new event and attach the signature *Sign Mi* to spread it within the group.

B. Design Idea of the TPM Management Strategy

After each group member sends an invitation to upload the shared data, the group manager selects a TPM for authorization. The port number address of the group member Mi is ui(x0; x1; : : : ; x) (is that the number of bits within the binary address), and therefore the port number address of the TPM is TPMi(x0; x1; : : : ; x). the subsequent describes the detailed selection method.

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

Referring to the principle of the TCP window, the sending window is about for the input end, and therefore the sending window corresponds to a period of your time. During this era, the group manager receives the appliance sent by the group member.

As shown in Fig. 3, the sending window has 3 pointers that slide clockwise. The window that permits u1 to send is that the time between P1 and P3, and therefore the current time is P2. Suppose that the group manager divides a particular period of time into 20 parts as t1-t20. If group member M1 sends an application at t8, which is within the allowable sending time of the sending window,

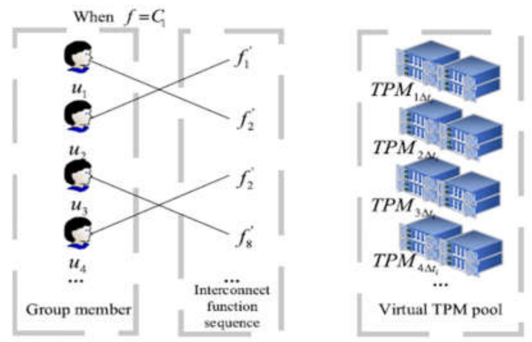


Figure 3: Virtual TPM pool Construction diagram.

the group manager will then receive the sent application from M_1. Consistent with the time-frame rotation, if group member M_1 doesn't send the request within the allowed sending time, P_2 slides clockwise to 1t0 1, which corresponds to M1's port number address u_1, then 1t0 1 becomes the time to process the request of M_1.

Therefore, the space between P1 and P3 determines the success rate of the appliance of the group members.

(1) The group manager and therefore the cloud protocol have a uniform sending window that's updated periodically as required. When moving to the proper between P1 and P3, the dimensions of the window changes. For group members who frequently use shared data, the group manager assigns them longer 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 therefore the address u_i of the input end. To increase the likelihood 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 therefore the group manager negotiate consistent f and f_i.) As shown in Fig. 5, assuming that u_2 sends the appliance to the group manager at 1t1, the group manager selects the interconnection function $f = C_1$ and selects $f_i = C_0$ from the sequence of interconnection functions. Consistent with the sending window in Fig. 4, the round is transferred to the time period 1t3, where the group manager can calculate the output at the instant through u2, 1t3, and therefore the interconnection function C0. That is, at that moment, the correspondence between the input and output are often represented by a matrix

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

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 method of choosing 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 1ti. For an efficient and secure TPM, the group manager allocates longer for it, and cancels the allotted sending period for the removed TPM.

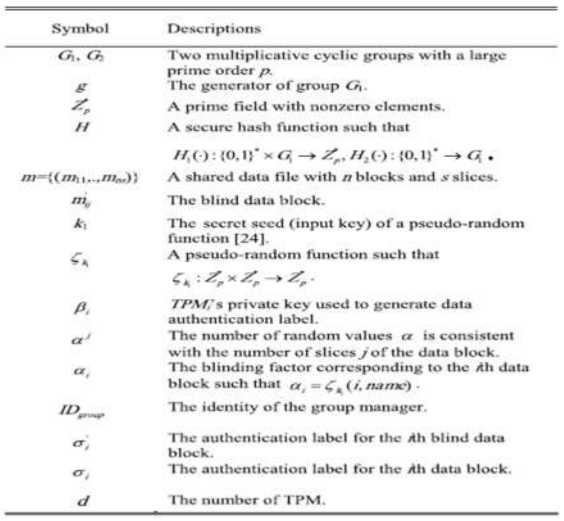
Therefore, after selecting the output address TPM_1, the group manager determines whether 1t3 corresponds to TPM1 consistent with the sending window of the output end. If it does, 1t3 are going to be sent to the cloud because the time authorized by the TPM; otherwise it'll select the time authorized by the TPM consistent with the time-frame rotation.

Through the above strategy, the group manager is provided with various execution modes. Consistent with the time-frame rotation and therefore the interconnection function, the group manager can dynamically select TPM_i. The group manager adjusts the time consistent with the particular situation. After one rotation, the group manager can send a replacement f to the cloud in order that both the output time and therefore the function change. The group member sends an application to upload data, and therefore the group manager randomly distributes' the computing task to the selected TPM_i consistent with the above rules. This approach is like building a virtual TPM pool, such that each TPM and every group member are independent of one another.

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 therefore the cloud. Because the group manager's port is that the port connection point between the group members and therefore the TPMs, he can select the send window and interaction functions, create the authorization consistent with 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 key 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



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 $H_1(\cdot):\{0,1\}^* \times G_1 \to Z_p, H_2(\cdot):\{0,1\}^* \to G_1$.
$m=\{(m_{11},..,m_{ns})\}$	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 $\zeta_{k_1}: Z_p \times Z_p \to Z_p$.
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.
σ_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.

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 present time. If it is, he verifies whether or not these authentication labels are correct. If they're correct, he will recover the important data using the blinding factor and compute their authentication labels. Finally, the cloud stores these real data and therefore the authentication labels.

Before executing the audit procedure, the group manager selects the send window and interaction functions and creates the authorization consistent with the TPM management strategy as described above. Then, the authorized TPM generates a challenge message (CM) and sends it to the cloud.

IV. SECURITY ANALYSIS

This section performs the safety 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 correctly stored within the cloud server, if the cloud provides a legitimate integrity certificate the validation procedure can then verify the integrity of the info. Proof: consistent with the character of bilinear mapping, the right sides of the equation are often 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 because the evidence comes from complete data, the validation equation will hold.

B. IDENTITY TRACEABILITY

Through Swirld's Hashgraph Consistency Algorithm [20], group members agree on the order of events (that is, the order of transaction records within the event) and therefore the timestamp for each event (transaction record). The transaction records of each event are often determined in chronological order according to the Hashgraph. Once a member of the group maliciously modifies the info block, the dirty data block may be discovered by other group members. Once such a knowledge dispute is generated, the group member may trace the usage history of the modified data block consistent with a Hashgraph.

C. EXPERIMENT EVALUATION

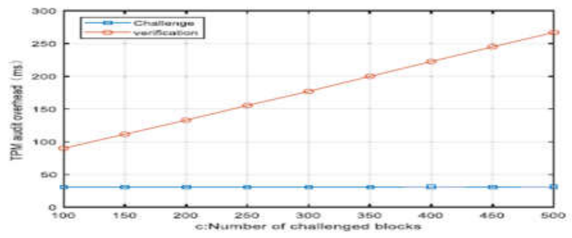
According to the analysis of Section VI, the scheme of this paper avoids potential security risks through a safer method. Within the audit scheme of shared data, group members are more concerned with the efficiency problem when using data [17], [18]. This section first analyses the computational overhead of the LSSA scheme, then evaluates it within the specific operating environment. The final results prove that the schemes are able to do lightweight calculations for group members, which LSSA has high security compared with similar audit schemes.

V. 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 programmers is written in C, and it uses the library functions within the Pairing-Based Cryptography (PBC) library to simulate the cryptographic operations, where the benchmark threshold is 512 bits, the dimensions of the element is jpj D160 bits in Z p , and therefore the size of the shared data is 20 MB. The experimental results are the averages of the 10 experiments.

1) OVERHEAD WITHIN THE AUDIT PHASE

Due to the powerful computing power of the cloud, more attention is usually paid to the overhead of the TPM within the process of a knowledge integrity audit. Consistent with Section V, we know that the auditing task of the TPM is split into two phases: the challenge generation and proof verification. To effectively evaluate the auditing computation over head of our scheme, we analyse the time cost of the TWO



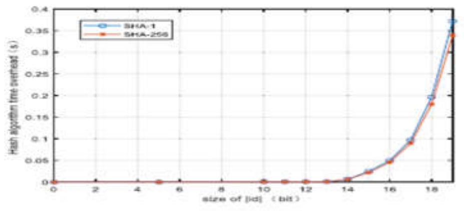
above phases. The experimental results are presented in Fig. 7.We choose to challenge different blocks from 100 to 500 in intervals of fifty and set the amount of TPMs to d D 20. As shown in Fig. 7, The time overhead of the challenge phase is constant, which is approximately 31 ms. The running time of the proof verification ranges from 80 ms to 260 ms. In step 8, H2(l) (l 2 Li, where L has c elements) is linear with the value c. Thus, the computation overhead of the proof verification linearly increases with the amount of the challenged blocks. As we all know, a greater number of challenged blocks results in a more accurate integrity verification. However, this will incur more computational overhead. Thus, it's desirable to find a trade-off between auditing computational cost and integrity guarantee.

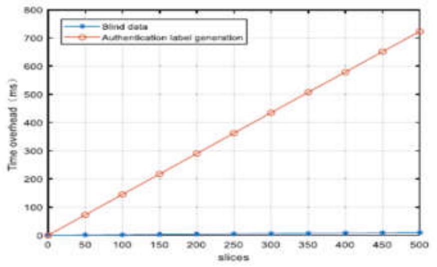
2) OVERHEAD WITHIN THE DATA UPLOAD PHASE

(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 *idi,j* sizes. The tested hash algorithm selects the simplest current SHA-1 and SHA-256 algorithms. As shown in Fig. 4, as the size of idi;j increases, the time overhead also increases. When the dimensions of idi;j increases from 0 to 14 bits, the time overhead of calculating the hash algorithm increases slowly. Therefore, so as to scale back the computational burden of the group members and group manager, the dimensions of idi;j is about to 14 bits, which is enough to record the general public identifier information of the shared data. When the id size is about to 14 bits, the time overheads of the 2 algorithms are 0.011 s and 0.012 s, respectively.

(2) Calculation overhead of authentication label generation As shown in Fig. 5, the experimental results show that the time overhead of the authentication label generation increases linearly with the amount of shared data slices s. When the data slice s D 500, the time overhead of the authentication label generation is 720 ms. Fig. 5 also shows that the time required for blinding data by group members is extremely small, and it therefore are often ignored.

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



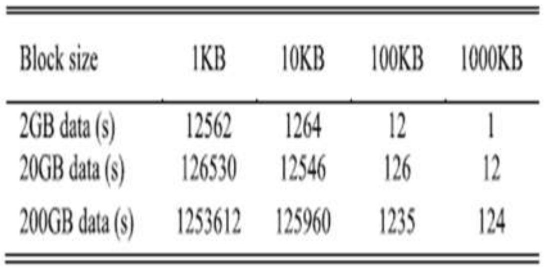


2 GB data, 20 GB data and 200 GB data. As shown in Table 2, the cost of the authentication label computation for an equivalent le decreases almost linearly with a rise within the block size. Consistent with our authentication label computation formula (5), we will see that every authentication label generation involves (s C 2) exponentiations, which

are the most overhead for computing the authentication label. The group manager can allocate the computing tasks to every TPM, which may reduce the computational burden of one TPM. As shown in Fig. 9, if 500 data slices are uploaded to d TPMs, the typical computing overhead of every TPM is 720/d ms.

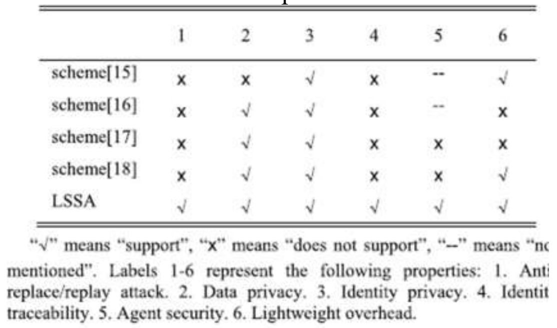
In summary, the experiment is evaluated using two better hash algorithms, and therefore the optimal digits of the id are selected. At this point, the calculation overhead of the group member is the sum of the time overheads for the hash algorithm and for blinding the info, which is lightweight for group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

Table 1: Authentication label generation time overhead with different block size



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 2: 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 "nc mentioned". Labels 1-6 represent the following properties: 1. Anti replace/replay attack. 2. Data privacy. 3. Identity privacy. 4. Identit traceability. 5. Agent security. 6. Lightweight overhead.

a single TPM during the generation of authentication labels. Nevertheless, it's worth to notice that the group size and the number of TPMs is specified by the group manager, which enables us to formulate flexible strategies to regulate the through put consistent with the particular scenario, to realize 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 will see that LSSA supports identity privacy; traceability, data privacy, agent security, a light-weight overhead, and it can resist the replace and replay attacks.

VI. CONCLUSION

In recent years, many studies on access control in cloud are based on attribute-based encryption algorithm (ABE). However, traditional ABE is not suitable for mobile cloud because it is computationally intensive and mobile devices only have limited resources. In this paper, we propose LDSS to address this issue. It introduces a novel LDSS-CP-ABE algorithm to migrate major computation overhead from mobile devices onto proxy servers, thus it can solve the secure data sharing problem in mobile cloud. The experimental results show that LDSS can ensure data privacy in mobile cloud and reduce the overhead on users' side in mobile cloud. In the future work, we will design new approaches to ensure data integrity. To further tap the potential of mobile cloud, we will also study how to do ciphertext retrieval over existing data sharing schemes.

VII. REFERENCES

[1] Gentry C, Halevi S. Implementing gentry's fully-homomorphic encryption scheme. in: Advances in Cryptology–EUROCRYPT 2011. Berlin, Heidelberg: Springer press, pp. 129-148, 2011.

[2] Brakerski Z, Vaikuntanathan V. Efficient fully homomorphic encryption from (standard) LWE. in: Proceeding of IEEE Symposium on Foundations of Computer Science. California, USA: IEEE press, pp. 97-106, Oct. 2011.

[3] Qihua Wang, Hongxia Jin. "Data leakage mitigation for discertionary access control in collaboration clouds". the 16th ACM Symposium on Access Control Models and Technologies (SACMAT), pp.103-122, Jun. 2011.

[4] Adam Skillen and Mohammad Mannan. On Implementing Deniable Storage Encryption for Mobile Devices. the 20th Annual Network and Distributed System Security Symposium (NDSS), Feb. 2013.

[5] Wang W, Li Z, Owens R, et al. Secure and efficient access to outsourced data. in: Proceedings of the 2009 ACM workshop on Cloud computing security. Chicago, USA: ACM pp. 55-66, 2009.

[6] Maheshwari U, Vingralek R, Shapiro W. How to build a trusted database system on untrusted storage. in: Proceedings of the 4th conference on Symposium on Operating System Design & Implementation-Volume 4. USENIX Association, pp. 10-12, 2000.

[7] Kan Yang, Xiaohua Jia, Kui Ren: Attribute-based fine-grained access control with efficient revocation in cloud storage systems. ASIACCS 2013, pp. 523-528, 2013.

[8] Crampton J, Martin K, Wild P. On key assignment for hierarchical access control. in: Computer Security Foundations Workshop. IEEE press, pp. 14-111, 2006.

[9] Shi E, Bethencourt J, Chan T H H, et al. Multi-dimensional range query over encrypted data. in: Proceedings of Symposium on Security and Privacy (SP), IEEE press, 2007. 350-364

[10] Cong Wang, Kui Ren, Shucheng Yu, and Karthik Mahendra Raje Urs. Achieving Usable and Privacy-assured Similarity Search over Outsourced Cloud Data. IEEE INFOCOM 2012, Orlando, Florida, March 25-30, 2012

[11] Yu S., Wang C., Ren K., Lou W. Achieving Secure, Scalable, and Fine-grained Data Access Control in Cloud Computing. INFOCOM 2010, pp. 534-542, 2010

[12] Kan Yang, Xiaohua Jia, Kui Ren, Bo Zhang, Ruitao Xie: DAC-MACS: Effective Data Access Control for Multiauthority Cloud Storage Systems. IEEE Transactions on Information Forensics and Security, Vol. 8, No. 11, pp.1790-1801, 2013.

[13] Stehlé D, Steinfeld R. Faster fully homomorphic encryption. in: Proceedings of 16th International Conference on the Theory and Application of Cryptology and Information Security. Singapore: Springer press, pp.377-394, 2010.

[14] Junzuo Lai, Robert H. Deng ,Yingjiu Li ,et al. Fully secure key-policy attribute-based encryption with constant-size ciphertexts and fast decryption. In: Proceedings of the 9th ACM symposium on Information, Computer and Communications Security (ASIACCS), pp. 239-248, Jun. 2014.

[15] Bethencourt J, Sahai A, Waters B. Ciphertext-policy attribute- based encryption. in: Proceedings of the 2007 IEEE Symposium on Security and Privacy (SP). Washington, USA: IEEE Computer Society, pp. 321-334, 2007.

[16] Liang Xiaohui, Cao Zhenfu, Lin Huang, et al. Attribute based proxy re-encryption with delegating capabilities. in: Proceedings of the 4th International Symposium on Information, Computer and Communications Security. New York, NY, USA: ACM press, pp. 276-286, 2009.

[17] Pirretti M, Traynor P, McDaniel P, et al. Secure atrribute-based systems. in: Proceedings of the 13th ACM Conference on Computer and Communications Security. New York, USA: ACM press, pp. 99-112, 2006.

[18] Yu S., Wang C., Ren K., et al. Attribute based data sharing with attribute revocation. in: Proceedings of the 5th International Symposium on Information, Computer and Communications Security (ASIACCS), New York, USA: ACM press pp. 261-270, 2010.

[19] Sandhu R S, Coyne E J, Feinstein H L, et al. Role-based access control models. Computer, 29(2): 38-47, 1996.

[20] Tian X X, Wang X L, Zhou A Y. DSP RE-Encryption: A flexible mechanism for access control enforcement management in DaaS. in: Proceedings of IEEE International Conference on Cloud Computing. IEEE press, pp.25-32, 2009

Authors Profile

Muppineni Renuka pursuing B Tech in computer science engineering from Qis college of Engineering and Technology (Autonomous & NAAC 'A' Grade), Ponduru Road, Vengamukkalapalem, Ongole, Prakasam Dist, Affiliated to Jawaharlal Nehru Technological University, Kakinada in 2016-20 respectively.

Alabazar Ramesh Has Received His B.Tech and M.Tech PG. He is dedicated to teaching field from last 4 years. He has guided 5 U.G students. At present he is working as Assistant Professor in Computer Science Engineering of Qis College of Engineering And Technology in Ongole.

G Nikhil Sai pursuing B Tech in computer science engineering from Qis college of Engineering and Technology (Autonomous & NAAC 'A' Grade), Ponduru Road, Vengamukkalapalem, Ongole, Prakasam Dist, Affiliated to Jawaharlal Nehru Technological University, Kakinada in 2016-20 respectively.

G Chaitanya pursuing B Tech in computer science engineering from Qis college of Engineering and Technology (Autonomous & NAAC 'A' Grade), Ponduru Road, Vengamukkalapalem, Ongole, Prakasam Dist, Affiliated to Jawaharlal Nehru Technological University, Kakinada in 2016-20 respectively.

Y Praveen Kumar pursuing B Tech in computer science engineering from Qis college of Engineering and Technology (Autonomous & NAAC 'A' Grade), Ponduru Road, Vengamukkalapalem, Ongole, Prakasam Dist, Affiliated to Jawaharlal Nehru Technological University, Kakinada in 2016-20 respectively.