Privacy-Preserving Public Auditing Scheme for Shared Data with Supporting Multi-function

Jianhong Zhang[1] and Xubing Zhao[2]
[1] College of Sciences, North China University of Technology, China
[2] Institute of Image Process and Pattern Recognition, North China University of Technology, Beijing, China
Email: jhzhangs@163.com

Abstract —The most common concerns for users in cloud storage are data integrity, confidentiality and availability, so various data integrity auditing schemes for cloud storage have been proposed in the past few years, some of which achieve privacy-preserving public auditing, data sharing and group dynamic, or support data dynamic. However, as far as we know, until now yet there doesn't exist a practical auditing scheme which can simultaneously realize all the functions above; In addition, in all the existing schemes, Block Authentication Tag (BAT) is adopted by data owner to achieve data integrity auditing; nevertheless, it's a arduous task to compute BATs for the resource-constrained data owner. In this paper, we propose a novel privacy-preserving public auditing scheme for shared data in the cloud, which can also support data dynamic operations and group dynamic. Our scheme has the following advantages:(1) we introduce proxy signature into the existing auditing scheme to reduce the cloud user's computation burden; (2) by introducing a Lagrange interpolating polynomial, our scheme realizes the identity's privacy-preserving without increasing computation cost and communication overhead, moreover it makes group dynamic simple;(3) it can realize the practical and secure dynamic operations of shared data bycombining the Merkle Hash Tree and index-switch table which is built by us; (4) to protect the data privacy and resist the active attack, the cloud storage server hides the actual proof information by inserting its private key in producing proof information process. Theoretical analysis demonstrates our scheme's security.

Index Terms—Cloud storage, data sharing, privacy-preserving, data dynamic, active attack, proxy signature, Lagrange interpolating polynomial.

I. Introduction

Cloud storage is an important service of cloud computing, which allows cloud users to store their detain the cloud and enjoy the on-demand cloud server. It offers great convenience to users since they do not have to care about the direct hardware or software managements. With the development of cloud computing and storage services, data are not only stored in the cloud, but also are routinely shared among a large number of users in a group and updated by the users through block modification, deletion, insertion, etc. For example, Dropbox, Google Docs and Sugar Sync enable multiple team members to work in sync, accessing and updating the same file on the cloud server. Compared to conventional systems, the integrity of data in cloud storage is subject to skepticism and scrutiny. Data stored in an untrusted cloud can easily be lost or corrupted, due to hardware failures and human errors. So it's necessary for the data owner to periodically check if the data in the cloud are stored correctly. Considering users' constrained computing and storage capabilities, public auditing schemes [1]-[3] are proposed, which only consider that let a TPA execute the auditing process for the data owner. However, the computation of the BATs is also very large for the data owner, especially for the cloud storage service supporting data dynamics, the cloud user need to recalculate the BATs for every data dynamic operation. We will elaborately discuss the solution to this problem later. Confidentiality is one of the major concerns in the adoption of cloud storage, so privacy-preserving public auditing schemes [4] have enabled the TPA to audit the data without learning the data content. A new privacy problem is introduced during the process of public auditing for shared data in the cloud, which is how to preserve identity privacy from the TPA. Because the identities of signers on shared data may indicate that a particular user in the group or a special block in shared data is a more valuable target than others. Wang and Li have proposed two identity privacy-preserving auditing mechanisms, called Oruta [5] and Knox [6]. To keep the identity of the signer on each data block private from the TPA, Oruta and Knox respectively utilize ring signature and group signature to construct the authenticator, so the computation and communication cost has increased a lot especially for the cloud user. Another major concern about data sharing across multiple users is group dynamic, which is adding new users to the group or revoking misbehaved users from the group. In the previous designs, such as [5], [6], adding or revoking users need re-compute part or all authenticators, so that introduce a significant computation burden to all users. In the paper [7] proposed by Yuan *et al.*, the cloud server will update the authentication tags of blocks that were last modified by the revoked user. Though it seems that it needs less computation and has no extra burden for the users, it provides an opportunity for the cloud server to modify the

Manuscript received March 3, 2015; revised July 26, 2015.
This work was partly by Supported by Beijing Natural Science Foundation (No: 4122024, 4132056)
Corresponding author: Zhang jianhong, email: jhzhangs@163.com.
doi:10.12720/jcm.10.7.535-542

authentication tags as his wish. The public auditing scheme in [7] also support data modification, however, by their scheme the users could not confirm whether the cloud server has updated the data correctly after they submitted the modification re-quest. The scheme in [8] has the same problem. Article [9] and [10] have introduced Merkle Hash Tree construction to the existing proof storage models to achieve efficient data dynamics. In their construction, the users can verify if the cloud server updates the data correctly by checking the root of the Merkle Hash Tree. However, the tags should be authenticated in each protocol execution rather than calculated or prestored by the verifier. Although the existing data auditing schemes have already had various properties, as far as we know, there is not yet a practical auditing scheme that can realize all the functions mentioned above simultaneously; In addition, all the existing schemes need the cloud users to compute the authentication tag for each data block. Generally the file which will be outsourced to the cloud is always large data and the cloud users' computing capabilities is constrained, so that the computation overhead is too expensive for the cloud users. Subsequently, some schemes were proposed to reduce computation cost in [11]-[14]. However, their schemes' efficiency is low yet. In this paper, we delegate the task of computing BATs to a cloud computing sever to reduce the burden for the cloud users. And our contributions can be summarized as follows:

- We bring proxy signature into the existing auditing scheme. The cloud user can outsource the computation of BATs to a cloud computing server, so that the user's burden would be greatly reduced.
- By introducing a Lagrange interpolating polynomi-al, our scheme realized that the identity privacy-preserving in the precondition of almost no any new additional computation and communication cost, moreover it makes group dynamic simple.
- We make a index-switch table and combining it with the Merkle Hash Tree to realize the practical and secure dynamic operations of the shared data by group users.
- We evaluated the performance of our scheme through both numerical analysis and experimental results, and the security of our design is proved.

II. Problrm Statement

A. System Model

Our protocol comprises three different entities as illustrated in Fig. 1. They can be identified as follows:

- CSS (Cloud Storage Server) — the party that has significant storage space resource to provide data storage service for group users defined below.
- CCS (Cloud Computing Server) — the party that has significant computational resource to provide computation service for group users defined below.
- GU (Group Users) — a group who have massive data stored on the cloud to share with each other. Group users are allowed to access and modify shared data based on access and modify control polices. And there is a master user u_0 in the group who manages the membership of the group, i.e., u_0 can invite new users to join in the group or revoke any group users when necessary.
- TPA (Third party auditor) — any party can periodically audit the integrity of data files stored in the cloud on the behalf of the group users.

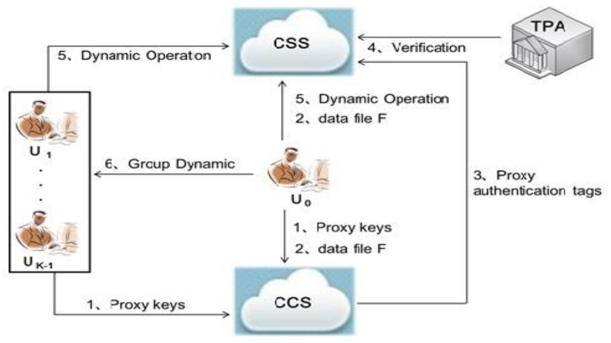

Fig. 1. System model

B. Threat Model

1) Integrity Threats: Three kinds of threats related to the integrity of shared data are possible. First, an adversary may try to corrupt the integrity of shared data and prevent users from using data correctly. Second, revoked users who no longer have data access privilege but try to illegally modify data. Third, the cloud service provider may inadvertently corrupt (or even remove) data in its storage due to hardware failures and human errors. Making matters worse, in order to avoid jeopardizing its reputation, the cloud server provider may be reluctant to inform users about such data corruption.

2) Privacy Threats: There exists two kinds of privacy threats of shared data. First, the content of data is confidential to the group. The crafty TPA who has no access permission may misbehave in order to learn some knowledge about the data stored on the cloud during the auditing process. Second, the identity of the signer on each block in shared data is private in the group. During the process of auditing, a semi-trusted TPA, who is only responsible for auditing the integrity of shared data, may try to reveal the identity of the signer on each block in shared data based on verification information. Once the TPA reveals the identity of the signer on each block, it can easily distinguish a high-value target (a particular user in the group or a special block in shared data).

C. Design Goals

1) Public Audibility: The TPA can audit the integrity of the cloud data on demand without retrieving a copy of the whole data or introducing additional online burden to the cloud users.

2) Storage Correctness: There exists no cheating cloud server that can pass the TPA's audit without indeed storing users' data intact.

3) Confidentiality: Any polynomial-time TPA cannot de-rive data content from the information collected during the auditing process.

4) Identity Privacy: During auditing, the TPA cannot distinguish the identity of the signer on each block in shared data.

5) Active Attack Resistance: Any polynomial-time adversary who is able to arbitrarily modify the cloud data cannot produce valid proof information to pass the verification algorithm in our scheme.

III. The Proposed Scheme

In this section, we present our security protocols for cloud data storage service with the aforementioned re-search goals in mind. We start with some basic solutions aiming to provide public auditing of the cloud data.

A. Notation and Preliminaries

1) F — a file to be outsourced.

2) $f_{\vec{a}(x)}$ — a polynomial with coefficient vector $\vec{a} = (a_1, a_2, \cdots, a_s)$.

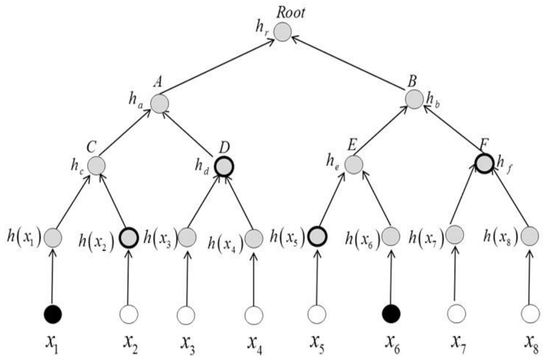

Fig. 2. An example of a RMHT

3) $H(\cdot), H_0(\cdot), H_1(\cdot), H_2(\cdot)$ — one-way hash functions.

4) pk_{ccs} — the public key of the cloud computing server.

5) pk_{css} — the public key of the cloud storage server. 6) pk_u — the public key of the group users.

We now introduce some necessary cryptographic back-grounds for our proposed scheme.

Bilinear Map. A bilinear map is a map $e: G \times G \rightarrow G_T$, where G is a $Gap\ Diffie-Hellman(GDH)$ group and G_T is another multiplicative cyclic group of order q with the properties of bilinearity, computability and non-degeneration, where q is a safe prime.

Ranked Merkle hash tree. A Ranked Merkle Hash Tree (RMHT) [9], [10] is constructed as a binary tree where the leaves in the RMHT are the hashes of authentic data values, and these leaf nodes are treated as the left-to-right sequence, so any leaf node can be uniquely determined by following this sequence and the way of computing the root in RMHT. For example, Fig. 2 depicts an example of authentication based on RMHT. The verifier with the authentic h_r requests for x_1, x_6 and requires the authentication of the received blocks. The prover provides the verifier with the auxiliary authentication information (AAI) $\Omega_1 = < h(x_2), h_d >$ and $\Omega_6 = < h(x_5), h_f >$. The verifier can then verify x_1 and x_6 by-first computing

$$h(x_1), h(x_6),$$

$$h_c = h(h(x_1) \| h(x_2)), h_e = h(h(x_5) \| h(x_6)),$$

$$h_a = h(h_c \| h_d), h_b = h(h_e \| h_f), h_r = h(h_a \| h_b),$$

and then checking if the calculated h_r is the same as the authentic one.

Index-switch table. An index-switch table is a comparison table between the *Actual Index* and the *Memory Index* of the data block. Before a file F be outsourced to the cloud, it will be splits into n data blocks, we name the index of each block *Actual Index* and the authentication tags are generated according to the Actual Index. When the file F is stored in the cloud, we call the storage order of each block *Memory Index*, which is also the load order when the cloud user accesses their data. The *Actual Index* and the *Memory Index* of the data blocks are consistent as in the Table I until there are some blocks that are deleted or some new blocks that are inserted. When a data block is deleted, the corresponding *Actual Index* in the index-switch table should be deleted and all the latter indexes are moved one index forward (see the example in Table II (a)). When a new data block is inserted, the corresponding Actual Index should be inserted in the same position of the index-switch table and all the latter indexes are moved one index backward (see the example in Table II (b)). The modification of data block will not affect the index-switch table.

Lagrange interpolating polynomial. If polynomial $F(x) = (x-a_1)(x-a_2)\cdots(x-a_n)$, where $(a_1, a_2, \cdots, a_n)$ are n different numbers,

$$\sum_{i=1}^{n} \frac{F(x)}{(x-a_i)F'(a_i)} = 1$$

where $F'(x)$ is the derived function of $F(x)$.

$$L(x) = \sum_{i=1}^{n} \frac{b_i F(x)}{(x-a_i)F(a_i)}$$

is a Lagrange interpolating polynomial with $L(a_i) = b_i, i = 1, 2, \cdots n.$

Actual Index	1	2	3	4	5	...	n
Memory Index	1	2	3	4	5	...	n

TABLE I: INDEX-SWITCH TABLE

(a) Delete the 4th data block

Actual Index	1	2	3	4	5	…	n
Memory Index	1	2	3	4	5	…	n

$\downarrow$ *Delete Actual Index 4*

Actual Index	1	2	3	5	6	…	n
Memory Index	1	2	3	4	5	…	n-1

TABLE II: EXAMPLE OF INDEX-SWITCH TABLE UPDATE UNDER BLOCK INSERTION AND DELETION OPERATION

(b) insert a new block after the 3th block

Actual Index	1	2	3	4	5	…	n
Memory Index	1	2	3	4	5	…	n

$\downarrow$ *insert n+1 after Actual Index 3*

Actual Index	1	2	3	n+1	4	…	n
Memory Index	1	2	3	4	5	…	n+1

B. *Our Construction*

Our scheme is constituted by five parts *(Proxy Signature, Outsource, Audit).* In *Proxy Signature* part, the keys are generated and the CSS computes the proxy authentication tags under the group users' authorization. *Outsource* is the course that the CSS stores the data file outsourced by the users and their proxy authentication tags in the cloud. Audit is the data integrity verification part. The detail procedure of our protocol execution is as follows:

1) Proxy signature

Setup—

1. Each group user $u_k (1 \le k \le K-1)$, first randomly chooses $\varepsilon_k \in Z_q^*$ and generates h^{ε_k}, where h is a generator of the group G, then they send h^{ε_k} and their ID ID_k to the master user u_0.

2. The master user u_0 randomly chooses $\varepsilon_0 \in Z_q^*$ and computes

$$t_k = H_0(ID_k),\ 0 \le k \le K+1$$

$$F(x) = \prod_{k=0}^{K-1}(x-t_k),\ L_1(x) = \sum_{k=0}^{K-1}\frac{h^{\varepsilon_k}F(x)}{(x-t_k)F'(t_k)}$$

3. The CCS randomly chooses $\varepsilon \in Z_q^*$ and generates $pk_{ccs} = h^{\varepsilon}$, then CCS sends pk_{ccs} to CSS and u_0.

4. The CSS randomly chooses $\xi \in Z_q^*$ and generates $pk_{css} = pk_{ccs}^{\xi}$.

5. The master user u_0 generates a warrant ω in which there is an explicit description of the description of the delegation relation between the group and CCS, and generates $g = H_1(\omega \| pk_p)$. Then the master user u_0 randomly chooses $\alpha \in Z_q^*$ and generates $pk_u = pk_{ccs}^{\alpha},\ \ P_j = g^{\alpha^j}, 0 \le j \le s.$ The master keys (MK) and public keys (PK) of the system are:

$$MK = \{\alpha\}$$

$$PK = \{q, h, pk_{ccs}, pk_{css}, pk_u, \{P_j\}_{0 \le j \le s}, L_1(x)\}$$

ProxyKeyGen—

6. In order to designate CCS as a proxy signer, each user of the group $u_k, 0 \le k \le K-1$ computes

$$g_{kj} = P_j^{1/\varepsilon_k} = (g^{\alpha^j})^{1/\varepsilon_k}, 0 \le j \le s \qquad (1)$$

and sends them to CCS. Then CCS computes $t_k = H_1(ID_k)$ and verifies these proxy keys by checking

$$e(\prod_{j=0}^{s} g_{kj}, L_1(t_k)) = e(\prod_{j=0}^{s} P_j, h) \qquad (2)$$

If Eq.2 holds, CCS accepts these proxy keys; otherwise, reject it by emitting $FALSE$.

ProxySigGen—

7. To outsource a file F, the user u_k splits data file F into n data blocks, and each block has s elements: $\{m_{ij}\}, 1 \le i \le n, 1 \le j \le s,$ and the i is the *Actual Index* of the data block m_i. Then the owner randomly chooses a file name $name \in Z_q^*$ and sends it together with F to the proxy signer CCS and the cloud storage server CSS.

8. Based on these proxy keys $g_{kj}, 0 \le j \le s,$ the CCS computes authentication tag σ_{i_k} for each data block of F as

$$\sigma_{i_k} = (g_{k0}^{B_i} \cdot \prod_{j=1}^{s} g_{kj}^{m_{ij}})^{\varepsilon} = (g^{B_i} \cdot g^{f_{\vec{\beta_i}}(\alpha)})^{\varepsilon/\varepsilon_k} \qquad (3)$$

where $\vec{\beta_i} = \{m_{i,1}, m_{i,2}, \cdots, m_{i,s}\}$ and $B_i = H_2(i \| H(m_i) \| k).$ Then the CCS sends $\Phi_k = \{\sigma_{ik}\}_{i \in [1,n]}$ and $\Psi = \{B_i\}_{i \in [1,n]}$ to the CSS.

2) Outsource

To make sure the authentication tags are generated correctly, the CSS needs to verify the signature firstly. Then the CSS will store the correct authentication tags together with the data block in the cloud.

SigVerify—

9. On receiving F from the GU u_k, the CSS computes $t_k = H_0(ID_k)$ and uses random sampling method to verify the proxy signature on σ_{i_k} as following after receiving Φ_k and Ψ from CCS:

$$e(\prod_{i\in q}\sigma_{i_k}, L_1(t_k)) = e(\prod_{i\in Q}(P_0^{B_i}\cdot\prod_{j=1}^{s}P_j^{m_{ij}}), pk_{ccs}) \quad (4)$$

where Q is a subset of $[1,n]$. If the equation fails, the CSS rejects by emitting $FALSE$; Otherwise, the CSS generates $\varphi_i = \{\sigma_{i_k}, t_k\}$, and stores $\Phi = \{\varphi_i\}_{i\in[1,n]}$ together with data file F in the cloud.

GenMHT—

10. The GU u_k and the CSS generate a same MHT, where the leave nodes of the tree are an ordered set of $H(m_i), (i = 1, 2, \cdots, n)$. Then the GU u_k generates a root R based on the construction of the MHT and signs R as: $sig_k(R) = H(R)^{1/\varepsilon_k}$. The GU u_k sends $sig_k(R)$ to the CSS and deletes $\{F, sig_k(R)\}$ from its local storage. The CSS sets $ID_r = ID_k$ and stores $\{sig_k(R), ID_R\}$ together with MHT.

3) Audit

The TPA issues a challenge to the CSS to make sure that the CSS has retained the data file F properly at the time of the audit. The CSS will derive a response message by executing *GenProof* using F and verification metadata as inputs. The TPA then verifies the response via *VerifyProof.*

Challenge—

11. The TPA randomly chooses a $c-$elements subset I of $[1,n]$ and two random numbers $R, r \xleftarrow{R} Z_q^*$. Then the TPA computes $L_2(x) = L_1(x)^R$ and sends the challenging message $CM = \{L_2(x), I, r\}$ to the CSS.

GenProof—

12. Based on the challenging message CM, the CSS firstly computes

$$\begin{aligned}\sigma_i &= e(\sigma_{i_k}, L_2(t_k)) \\ &= e((g^{B_i}\cdot g^{f_{\vec{\beta}}(\alpha)})^{\varepsilon/\varepsilon^k}, h^{\varepsilon_k R}) \qquad (5) \\ &= e((g^{B_i}\cdot g^{f_{\vec{\beta}}(\alpha)}), h)^{\varepsilon R}\end{aligned}$$

and

$$\{p_i = r^i \bmod p\}, \{\mu_i = \xi\cdot p_i\}, i\in I$$

Following the CSS produces

$$\sigma = \prod_{i\in I}\sigma_i^{\mu_i}, y = f_{\vec{A}}(r)$$

where $\vec{A} = \{\sum_{i\in I}\mu_i m_{i,1}, \cdots, \sum_{i\in I}\mu_i m_{i,s}\}$

Then the CSS divides the polynomial $f(x) - f(r)$ with $(x-r)$ using polynomial long division, and denotes the coefficients vector of the resulting quotient polynomial as $\vec{\omega} = (\omega_1, \omega_2, \cdots, \omega_s)$, that is, $f_{\vec{\omega}}(x) \equiv \frac{f_{\vec{A}}(x) - f_{\vec{A}}(r)}{x-r}$. The CSS generates

$$\psi = \prod_{j=1}^{s}P_j^{\omega_j} = g^{f_{\vec{\omega}}(\alpha)} \qquad (6)$$

Finally the CSS sends the proof information $\Pr f = \{\sigma, \psi, y\}$ to the TPA.

VerifyProof—

13. On receiving the $\Pr f$, TPA firstly computes $p_i = r^i \bmod p, i\in I$ and $\eta = g^{\sum_{i\in I}B_i p_i}$, where the B_i is downloaded from the cloud. Based on η, the TPA verifies the integrity of F together with $\Pr f = \{\sigma, \psi, y\}$ as:

$$e(\eta, pk_{css}^R)\cdot e(\psi^R, pk_u\cdot pk_{ccs}^{-r}) = \sigma\cdot e(g^{-y}, pk_{ccs}^R) \quad (7)$$

If Eq.8 holds, the user outputs *AuditRst* as accept; otherwise, outputs AuditRst as reject.

IV. Performance Evalution

A. Numerical Analysis

In this section, we numerically analyze our proposed scheme in terms of computation cost and communication cost.

Computation cost: In the Setup algorithm, the master user needs to perform $(s+2)Exp_{(G)}$ operations to generate public keys for the system, where s is the number of elements in block. In *ProxyKeyGen*, each user needs to perform $(s+1)Exp_{(G)}$ operations to generate proxy keys, and the CCS needs to perform $K\ Pair$ operations to verify these proxy keys where K is the number of group users. In the *ProxySigGen*, the CCS conducts $(s+2)nExp_{(G)} + snMul_{(G)}$ operations for each data file, where n is the number of blocks in a file. The *Outsource* consists of 2 algorithms *SigVerify* and *GenMHT.* In *SigVerify,* the CSS conducts $Pair + (s+1)Exp_{(G)} + (s+1+2d)Mul_{(G)}$ operations to verify the proxy authentication tags, where d is a constant number of blocks selected for verifying. *Proxy Signature* and *Outsource* are preprocessing procedures and will not influence the real-time verification performance.

In *Audit,* the TPA first perform the *Challenge* algorithm to generate the challenging message CM by choosing a constant number of random number and get $L_2(x)$ at negligible cost. The CSS then runs the *GenProof* algorithm with

$$cExp_{(G_T)} + sExp_{(G)} + (c-1)Mul_{(G_T)} + (s-1)Mul_{(G)}$$
$$+cExp_{(Z)} + cMul_{(Z)} + cPair$$

operations, where c is a constant number of blocks selected for challenging. To verify the integrity of the file, the TPA only needs

$$6Exp_{(G)} + 2Mul_{(G_T)} + Mul_{(G)} + cExp_{(Z)} + 3Pair$$

operations. Therefore, the total computation cost for the entire verification process is

$$cExp_{(G_T)} + (s+6)Exp_{(G)} + (c+1)Mul_{(G_T)} + sMul_{(G)}$$
$$+2cExp_{(Z)} + cMul_{(Z)} + (c+3)Pair.$$

B. Experimental Results

Each cryptographic operation involved in our construction will be evaluated in C++ on Computer of Intel(R)Core(TM)i5-3337U CPU @ 1.80GHZ with Windows7 OS as in paper [15]. We set the security parameter |q| = 160bits, we also assume the total number of blocks in data n =100000, each block contains $s=100$ elements, the size of each data block as 2 KB, and total size of shared data is 2 GB. We also choose $|n|$=20bits. The computation time for different types of basic cryptographic operations in our scheme is presented in Table III

TABLE III: COMPUTATION TIME FOR DIFFERENT OPERATIONS

Operation	Pairing	Exp/Mul(in G)
Time(ms)	9	13/11
Operation	Exp/Mul(in G_T)	Exp/Mul(in Z_q)
Time(ms)	1.7/1.6	0.05/0

1. *The Key Generation Time:* We first make a comparison between our scheme and Yuan *et al's* scheme about the time that the master user u_0 need to generate the public keys and master keys for the system. Fig. 3(a) indicates that the setup time in ref. [7] is proportional to the group size, but, the setup time in our scheme is constant. So our design reduces the computational cost of the master user u_0. However, in our scheme, to delegate the computation of authentication tags to the CCS, every user in the group need to generate the proxy keys. Fig. 3(b) indicates that the computation cost of every user in our scheme is proportional to the sector number of each block. And we can see that the total computational time of *Setup* and *ProxyKeyGen* in our scheme is almost as same as the computational time of *Setup*.

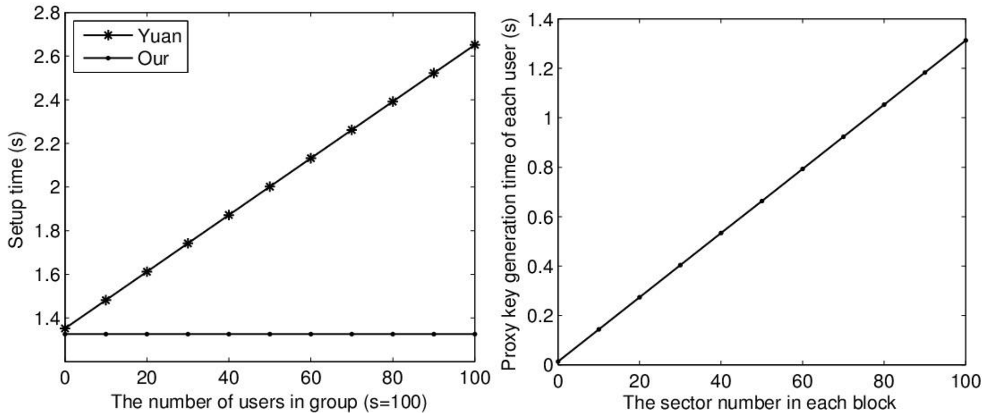

Fig. 3. (a)The setup time under different user number; (b) the proxy key generation time under different sector number

TABLE IV: PERFORMANCE UNDER DIFFERENT NUMBER OF SAMPLED BLOCKS C FOR HIGH ASSURANCE (⩾ 95%) AUDITING

s=100	Our		Yuan		Knox	
Sampled blocks c	460	300	460	300	460	300
CSS com.time(s)	8.0684	6.0924	8.0684	6.0924	11.0400	7.2000
TPA com.time(s)	0.1422	0.1342	0.1422	0.1342	147.8800	96.4560

2. *Computation cost and Communication cost of verification*: First we measure the total computation cost of verification. As discussed in [8], we can set the number of challenging block as 300 in our experiments to achieve 95% detection probability and set the challenging number as 460 to achieve 99% detection probability. We give the experiment result on performance comparison among our scheme, [7] and [6] in Table IV. It can be shown that the performance of our scheme is just the same as that of [7] and is better than that of [6], especially the less TPA computation time. Then we make another verification time comparison between our scheme and ref. [16] in Fig. 4, and we can see that the total verification time in ref. [16] is linearly increasing with the user number rising while the total verification time in our scheme is constant. To further research the performance of the verification in our scheme, we measure the computation cost of CSS and TPA separately with different challenging number c and different sector number s. The computational cost of verification on the CSS side is linearly increasing with the growth of the challenging number c and the sector number s, while

the computational cost of verification on the TPA side is almost constant regardless of the challenging number c and the sector number s. Considering the communication cost, when the challenging number c =460, we change the user number in the group from 5 to 150, Fig. 5(a) shows that our scheme complete a data integrity verification at the communication cost from 1.4KB to 4.3KB. Fig. 5(b) shows that when the user number $s = 100$, the challenging number changing from 300 to 2000, the communication cost of one audit in our scheme varies from 2.9 KB to 7KB. Compared with ref. [16], we achieved less communication cost and smaller growth rate with the user number or challenging number rising. From the above experimental results, it is clear that the computational burden of the group user will be significantly reduced by introducing the CSS in our scheme, and we can complete public privacy-preserving verification with less computation and communication cost.

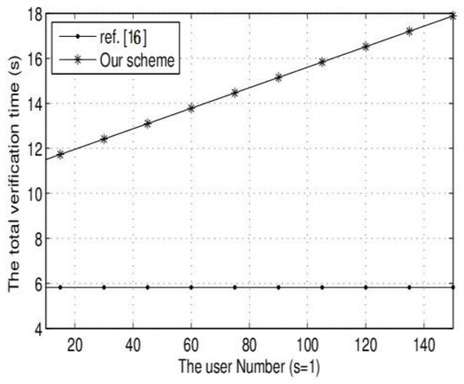

Fig. 4. The total computation cost under different user number

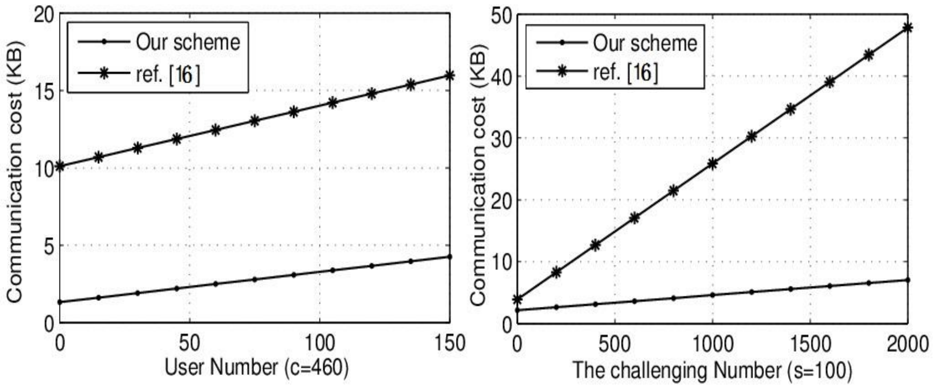

Fig. 5. (a), The communication cost of verification on different number of users; (b) The communication cost of verification on different challenging number

V. Conclusions

In this paper, we proposed a new public auditing mechanism for shared data in cloud. The TPA cannot get any information about the content of the outsourced data file and any identity information of the group user during the auditing process. The group users can do data dynamic operations on the shared data and the group master user can invite some new one to the group or revoke some group user with a negligible computation cost. The experiment result in our scheme has shown that the computation of authentication tags is too large for the cloud user, so we have outsourced the computation of authentication tags to a CSS.

References

[1] G. Ateniese, R. Burns, R. Curtmola, J. Herring, *et al.*, "Provable data possession at untrusted stores," in *Proc. CCS'07*, 2007, pp. 598-609.

[2] A. Juels, J. Burton, and S. Kaliski, "PORs: Proofs of retrivability for large files," in *Proc. of CCS'07*, October 2007, pp. 584-597.

[3] H. Shacham and B. Waters, "Compact proofs of retrievability," in *Proc. Asia Crypt*, Dec. 2008, vol. 5350, pp. 90-107.

[4] C. Wang, Q. Wang, K. Ren, and W. Lou, "Privacy-preserving public auditing for data storage security in cloud computing," in *Proc. IEEE INFOCOM*, 2010, pp. 525-533.

[5] B. Wang, B. Li, and H. Li, "Oruta: privacy-preserving public auditing for shared data in the cloud," in *Proc. IEEE International Conference on Cloud Computing*, 2012, pp. 293-302.

[6] B. Wang, B. Li, and H. Li, "Knox: Privacy-preserving public auditing for shared data with large groups in the cloud," in *Proc. of ACNS*, 2012, pp. 507-525.

[7] J. Yuan and S. Yu, "Efficient public integrity checking for cloud data sharing with multi-user modification," in *Proc. IEEE INFOCOM*, 2014, pp. 2121-2129.

[8] Y. Zhu, G. Ahn, H. Hu, S. Yau, H. An, and S. Chen, "Dynamic audit services for outsourced storages in clouds," *IEEE Transations on Services Competing*, vol. 6, no. 2, pp. 227-238, 2013.

[9] Q. Wang, C. Wang, K. Ren, W. Lou, and J. Li, "Enabling public audit ability and data dynamics for storage security in cloud computing," *IEEE Transations on Parallel and Distributed Systems*, vol. 22, no. 5, pp. 847-859, May 2011.

[10] C. Liu, J. Chen, L. Yang, X. Zhang, C. Yang, R. Ranjan, and K. Ra-mamohanrao, "Authorized public auditing of dynamic big data storage on cloud with efficient verifiable fine-grained updates," *IEEE Transitionson Parallel and Distributed Systems*, vol. 25, no. 9, pp. 2234-2244, Sep. 2014.

[11] W. Diffie and M. Hellman, "New direction in cryptography," *IEEE Information Theory*, vol. 22, no. 6, pp. 644-654, Sep. 1976.

[12] D. Boneh and X. Boeh, "Short signatures without random oracles, in *Proc. Annual International Conference on the Theory and Applications of Cryptographic Techniques*, 2004, pp. 56-73.

[13] D. Boneh and M. K. Franklin, "Identity-based encryptions from the wail pairing," in *CRYPTO'01. Springer-Verlag*, 2001, pp. 213-229.

[14] Z. Liu, J. Li, X. Chen, J. Yang, and C. Jia, "TMDS: Thin-model data sharing scheme supporting keyword search in cloud storage," *Information Security and Privacy Lecture Notes in Computer Science*, vol. 8544, pp. 115-130, 2014.

[15] B. Wang, B. Li, and L. Hui, "Panda: Public auditing for shared data with efficient user revocation in the cloud," in *Proc. IEEE INFOCOM*, pp. 2904-2912, 2013.

[16] J. W. Yuan and S. C. Yu, "Efficient public integrity checking for cloud data sharing with multi-user modification," in *Proc. 33nd IEEE International Conference on Computer Communication*, Apr. 2014, pp 359-368.

Jian-Hong Zhang was born in Hebei Province, China, in 1975. He received the B.S. degree from the Hebei Normal University, Hebei, in 1998 and the M.S. degree and Ph.D from Guizhou University and Xidian University in 2001 and 2003, respectively, His research interests include information security, cryptography, cloud security.

Xu-Bing Zhao was born in Hebei Province, China, in 1991. He received the B.S. degree from the Hebei Normal University, Hebei, in 2014 , in applied mathematics. He is currently pursuing the Master's degree with the applied mathematics. His research interest is secure cloud storage.