IPHV8I2050X
International Journal Of Advanced Research and Innovation -Vol.8, Issue .II
ISSN Online: 2319 – 9253
Print: 2319 – 9245

AN EFFICIENT AUDITING SCHEME FOR SHARED DATA WITH SUPPORTING MULTICLOUD

[#1]Dr.V.ANANDAM,
[#2]Mr. B. SEETHARAMULU,
VARDHAMAN COLLEGE OF ENGINEERING SHAMSHABAD, HYDERABAD, T.S INDIA.

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 achieved privacy-preserving public auditing, data sharing and group dynamics, or support data dynamics. However, as far as we know, until now yet there is no presence of a practical auditing scheme which can simultaneously realize all the above functions. In addition, in all the existing schemes, Block Authentication Tag (BAT) is adopted by data owner to achieve data integrity auditing; nevertheless, it's an 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 dynamics. 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 by combining 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.

Keywords: *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 data in the cloud and enjoy the on-demand cloud server. It offers great convenience to users since they do not have to take 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 synchronous, 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 (Third Party Auditor) 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 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 to re-compute part or all authenticators, so for introducing 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 authentication tags. The public auditing scheme in [7] also

IPHV8I2050X
International Journal Of Advanced Research and Innovation -Vol.8, Issue .II
ISSN Online: 2319 – 9253
Print: 2319 – 9245

supports 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 pre stored 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 polynomial, 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 an 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 to be correct.

II. SYSTEM MODEL

In this paper the system model involves three parties: the cloud server, a group of users and a public verifier. Here, two types of users in a group: the original user and a number of group users are there. The original user initially creates shared data in the cloud, and shares it with group users. Both the group user and original users are members of the group. It is allowed to access and modify shared data for every member of the group. Shared data and its verification metadata (i.e., signatures) are stored in the cloud server. A public verifier, such as a third-party auditor providing expert data auditing services or a data user outside the group intending to utilize shared data, is able to publicly verify the integrity of shared data stored in the cloud server. When a public verifier wishes to check the integrity of shared data, it first sends an auditing challenge to the cloud server. As shown in Fig. 1. Alice and Bob share a data file in the cloud after receiving the auditing challenge and a public verifier audits shared data integrity with existing mechanisms. With an auditing proof of the possession of shared data the cloud server responds to the public verifier. Then, this public verifier checks the correctness of the entire data by verifying the correctness of the auditing proof. Essentially, the process of public auditing is a challenge and- response protocol between a public verifier and the cloud server.

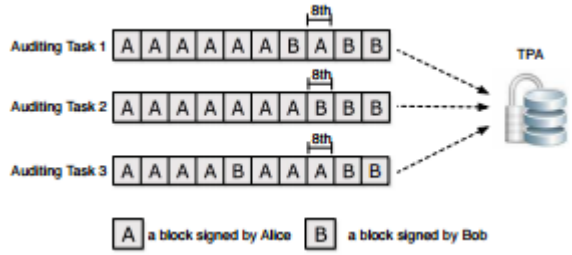


Fig. 1. Alice and Bob share a file in the cloud, and the TPA audit the integrity of data with existing mechanisms

III. PROBLEM STATEMENT

A. System Model

As illustrated in Fig. 2, our work in this paper involves three parties: the cloud server, the third party auditor (TPA) and users. There are two types of users in a group: the original user and a number of group users. The original user and group users are both members of the group. Group members are allowed to access and modify shared data created by the original user based on access control polices. Shared data and its verification information (i.e. signatures) are both stored in the cloud server. The third party auditor is able to verify the integrity of shared data in the cloud server on behalf of group members.

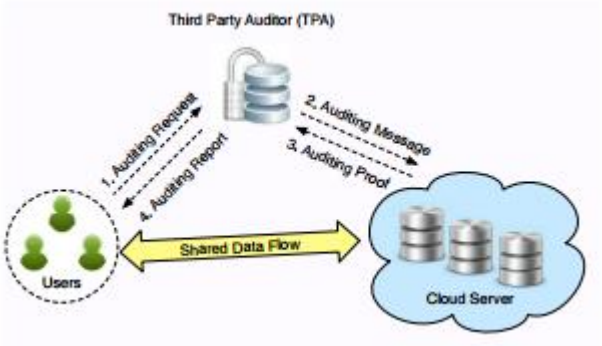


Fig. 2. Our system model includes the cloud server, the third party auditor and users

In this paper, we consider how to audit the integrity of shared data in the cloud with static groups only. It means the group is pre-defined before shared data is created in the cloud and the membership of users in the group is not changed during data sharing. The original user is responsible for deciding who is able to share data before outsourcing data to the cloud. Another interesting problem is how to audit the integrity of shared data in the cloud with dynamic groups — a new user can be added into the group and an existing group member can be revoked during data sharing — while still preserving identity privacy. We will leave this problem to our future work. When a user (either the original user or a group user) wishes to check the integrity of shared data, first it sends an auditing request to the TPA. After receiving the auditing request, the TPA generates an auditing message to the cloud server, and retrieves an auditing proof of shared data from the cloud server. Then the TPA verifies the correctness of the auditing proof. Finally, the TPA sends an auditing report to the user based on the result of the verification.

B. Threat Model

1) Integrity Threats: Two 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, 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 corruption of data.

2) Privacy Threats: The identity of the signer on each block in shared data is private and confidential to 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 Objectives: To enable the TPA efficiently and securely verify shared data for a group of users, Oruta should be designed to achieve following properties:

(1) Public Auditing: The third party auditor is able to verify the integrity of shared data for a group of users without retrieving the entire data.

(2) Correctness: The third party auditor is able to correctly detect whether there is any corrupted block in shared data.

(3) Unforgeability: Only a user in the group can generate valid verification information on shared data.

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

IV. PERFORMANCE

In this section, we first analyze the computation and communication costs of Oruta, and then evaluate the performance of Oruta in experiments.

A. Computation Cost

During an auditing task, the public verifier first generates some random values to construct an auditing challenge, which only introduces a small cost in computation. Then, after receiving the auditing challenge, the cloud server needs to compute an auditing proof $\{\lambda, \mu, \phi, \{idj\}j\epsilon J\}$. Based on the description in the computation cost of calculating an auditing proof is about (k+dc) ExpG1+ dcMulG1+ ckMulZp + kHashZp, where ExpG1 denotes the cost of computing one exponentiation in G1, MulG1 denotes the cost of computing one multiplication in G1, MulZp and HashZp respectively denote the cost of computing one multiplication and one hashing operation in Zp. To check the correctness of an auditing proof $\{\lambda, \mu, \phi, \{idj\}j\epsilon J\}$, a public verifier audits. The total cost of verifying this auditing proof is about (2k+c) ExpG1 + (2k + c) MulG1 + dMulGT + cHashG1 + (d + 2) Pair. We use Pair to denote the cost of computing one pairing operation one: $G1 \times G2 \rightarrow GT$.

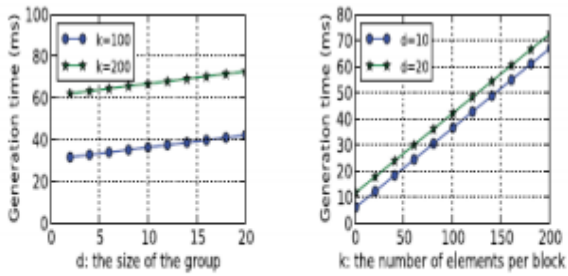
B. Communication Cost

The communication cost of Oruta is mainly introduced by two aspects: the auditing challenge and auditing proof. For each auditing challenge $\{j, yj\}$ $j\epsilon J$, the communication cost is c (|q|+ |n|) bits, where |q| is the length of an element of Zq and |n| is the length of an index. Each auditing proof $\{\lambda, \mu, \phi, \{idj\}j\epsilon J\}$ contains (k+ d) elements of G1, k elements of Zp and c elements of Zq, therefore the communication cost of one auditing proof is (2k + d)|p|+c|q| bits.

C. Experimental Results

We now evaluate the efficiency of Oruta in experiments. In our experiments, we utilized the GNU Multiple Precision Arithmetic (GMP) library and Pairing Based Cryptography (PBC) library. All the following experiments are based on C and tested on a 2.26 GHz Linux system over 1,000 times. Because Oruta needs more exponentiations than pairing operations during the process of auditing, the elliptic curve we choose in our experiments is an MNT curve with a base field size of 159 bits, which has a better performance than other curves on computing exponentiations. We choose |p|= 160 bits and |q| = 80 bits. We assume the total number of blocks in shared data is n = 1,000, 000 and |n| = 20 bits. The size of shared data is 2GB. To keep the detection probability greater than 99 percent, we set the number of selected blocks in an auditing task as c = 460 [10]. If only 300 blocks are selected, the detection probability is greater than 95 percent. We also assume the size of the group be [2, 20] in the following experiments. Certainly, if a larger group size is used, the total computation cost will increase due to the increasing number of exponentiations and pairing operations.

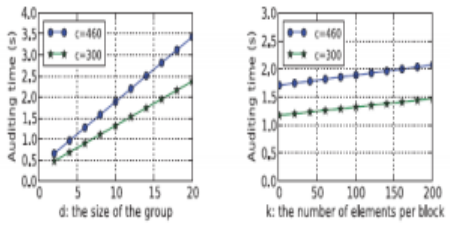
IPHV8I2050X
International Journal Of Advanced Research and Innovation -Vol.8, Issue .II
ISSN Online: 2319 – 9253
Print: 2319 – 9245



(a) Impact of d on signature generation time (ms). (b) Impact of k on signature generation time (ms).

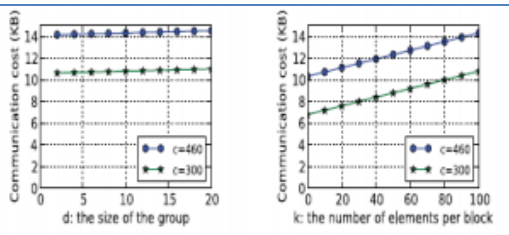
Fig.3. Performance of signature generation.

Performance of Signature Generation According to, the generation time of a ring signature on a block is determined by the number of users in the group and the number of elements in each block. As illustrated in Figs. 2a and 5b, when k is fixed, the generation time of a ring signature is linearly increasing with the size of the group; when d is fixed, the generation time of a ring signature is linearly increasing with the number of elements in each block. Specifically, when d = 10 and k = 100, a user in the group requires about 37 milliseconds to compute a ring signature on a block in shared data. Performance of Auditing Based on our proceeding analyses, the auditing performance of Oruta under different detection probabilities is illustrated in Figs. 3a and 4b, . As shown in Fig. 3a, the auditing time is linearly increasing with the size of the group. When c = 300, if there are two users sharing data in the cloud, the auditing time is only about 0.5 seconds; when the number of group member increases to 20, it takes about 2.5 seconds to finish the same auditing task. The communication cost of an auditing task under different parameters is presented in Figs. 4a and 4b. Compared to the size of entire shared data, the communication cost that a public verifier consumes in an auditing task is very small. It is clear that when maintaining a higher detection probability, a public verifier needs to consume more computation and communication overhead to finish the auditing task. Specifically, when c = 300, it takes a public verifier 1.32 seconds to audit the correctness of shared data, where the size of shared data is 2 GB; when c = 460, a public verifier needs 1.94 seconds to verify the integrity of the same shared data.



(a) Impact of d on auditing time (second), where $k = 100$. (b) Impact of k on auditing time (second), where $d = 10$.

Fig.4. Performance of auditing time



(a) Impact of d on communication cost (KB), where $k = 100$. (b) Impact of k on communication cost (KB), where $d = 10$.

Fig.5. Performance of communication cost.

As we discussed in the previous section, the privacy performance of our mechanism depends on the number of members in the group. Given a block in shared data, the probability that a public verifier fails to reveal the identity of the signer is 1-1/d, where d ≥2. Clearly, when the number of group members is larger, our mechanism has a better performance in terms of privacy. As we can see from Fig. 5a, this privacy performance increases with an increase of the size of the group. Performance of Batch Auditing As we discussed when there are multiple auditing proofs, the public verifier can improve the efficiency of verification by performing batch auditing. In the following experiments, we choose c = 300, k = 100 and d = 10. Compared to verifying a number of B auditing proofs one by one, if these B auditing proofs are for different groups, batching auditing can save 2.1 percent of the auditing time per auditing proof on average (as shown in Fig. 5a). If these B auditing tasks are for the same group, batching auditing can save 12.6 percent of the average auditing time per auditing proof (as shown in Fig. 5b). Now we evaluate the performance of batch auditing when incorrect auditing proofs existing among the B auditing proofs. As we mentioned we can use binary search in batch auditing, so that we can distinguish the incorrect ones from the B auditing proofs. However, the increasing number of incorrect auditing proofs will reduce the efficiency of batch auditing. It is important for us to find out the maximal number of incorrect auditing proofs exist in the B auditing proofs, where the batch auditing is still more efficient than separate auditing. In this experiment, we assume the total number of auditing proofs in the batch auditing is B = 128 (because we leverage binary search, it is better to set B as a power of 2), the number of elements in each block is k = 100 and the number of users in the group is d = 10. Let A denote the number of incorrect auditing proofs. In addition, we also assume that it always requires the worst case algorithm to detect the incorrect auditing proofs in the experiment; the extra computation cost in binary search is mainly introduced by extra pairing operations. As shown in Fig.5a, if all the 128 auditing proofs are for different groups, when the number of incorrect auditing proofs is less than 16 (12 percent of all the auditing proofs), batching auditing is still more efficient than separate auditing. Similarly, in Fig. 5b, if all the auditing proofs are for the same group, when the number of incorrect

auditing proofs is more than 16, batching auditing is less efficient than verifying these auditing proofs separately.

V. CONCLUSION

In this paper, we propose Oruta, the first privacy preserving public auditing mechanism for shared data in the cloud. With Oruta, the TPA is able to efficiently audit the integrity of shared data, yet cannot distinguish who is the signer on each block, which can preserve identity privacy for users. An interesting problem in our future work is how to efficiently audit the integrity of shared data with dynamic groups while still preserving the identity of the signer on each block from the third party auditor.

REFERENCES

[1] Boyang Wang, Student Member, IEEE, Baochun Li, Senior Member, IEEE, and Hui Li, Member, IEEE, "Oruta: Privacy-Preserving Public Auditing for Shared Data in the Cloud", IEEE Transactions on Cloud Computing, Vol. 2, No. 1, January-March 2014.

[2] B. Wang, B. Li, and H. Li, "Oruta: Privacy-Preserving Public Auditing for Shared Data in the Cloud," Proc. IEEE Fifth Int'l Conf. Cloud Computing, pp. 295-302, 2012.

[3] M. Armbrust, A. Fox, R. Griffith, A.D. Joseph, R.H. Katz, A. Konwinski, G. Lee, D.A. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "A View of Cloud Computing," Comm. ACM, vol. 53, no. 4, pp. 50-58, Apr. 2010.

[4] K. Ren, C. Wang, and Q. Wang, "Security Challenges for the Public Cloud," IEEE Internet Computing, vol. 16, no. 1, pp. 69-73, 2012.

[5] D. Song, E. Shi, I. Fischer, and U. Shankar, "Cloud Data Protection for the Masses," Computer, vol. 45, no. 1, pp. 39-45, 2012.

[6] C. Wang, Q. Wang, K. Ren, and W. Lou, "PrivacyPreserving Public Auditing for Data Storage Security in Cloud Computing," Proc. IEEE INFOCOM, pp. 525-533, 2010.

[7] B. Wang, M. Li, S.S. Chow, and H. Li, "Computing Encrypted Cloud Data Efficiently under Multiple Keys," Proc. IEEE Conf. Comm. and Network Security (CNS '13), pp. 90- 99, 2013.

[8] R. Rivest, A. Shamir, and L. Adleman, "A Method for Obtaining Digital Signatures and Public Key Cryptosystems," Comm. ACM, vol. 21, no. 2, pp. 120-126, 1978.

[9] The MD5 Message-Digest Algorithm (RFC1321). https://tools. ietf.org/html/rfc1321, 2014.

[10] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, and D. Song, "Provable Data Possession at Untrusted Stores," Proc. 14th ACM Conf. Computer and Comm. Security (CCS '07), pp. 598-610, 2007.

[11] H. Shacham and B. Waters, "Compact Proofs of Retrievability," Proc. 14th Int'l Conf. Theory and Application of Cryptology and Information Security: Advances in Cryptology (ASIACRYPT '08), pp. 90-107, 2008.

[12] C. Erway, A. Kupcu, C. Papamanthou, and R. Tamassia, "Dynamic Provable Data Possession," Proc. 16th ACM Conf. Computer and Comm. Security (CCS'09), pp. 213-222, 2009.