International Journal of Scientific Research and Modern Education (IJSRME)
ISSN (Online): 2455 – 5630
(www.rdmodernresearch.com) Volume I, Issue I, 2016

A CLOUD STORAGE SYSTEM FOR SHARING DATA SECURELY WITH PRIVACY PRESENTATION

M. Ramu*, G. Sudhakar*, A. Benwin Joshua**
& Kishore Joseph***

* Assistant Professor, Department of Information Technology, Dhanalakshmi Srinivasan Engineering College, Perambalur, Tamilnadu
** UG Scholar, Department of information Technology, Dhanalakshmi Srinivasan Engineering College, Perambalur, Tamilnadu
*** UG Scholar, Department of Computer Science & Engineering, Dhanalakshmi Srinivasan Engineering College, Perambalur, Tamilnadu

Abstract:

With cloud data services, it is commonplace for data to be not only stored in the cloud, but also shared across multiple users. Unfortunately, the integrity of cloud data is subject to skepticism due to the existence of hardware/software failures and human errors. Several mechanisms have been designed to allow both data owners and public verifiers to efficiently audit cloud data integrity without retrieving the entire data from the cloud server. However, public auditing on the integrity of shared data with these existing mechanisms will inevitably reveal confidential information—identity privacy—to public verifiers. In this paper, we propose a novel privacy-preserving mechanism that supports public auditing on shared data stored in the cloud. And also we introduced thump impression technique. From this new technology we give High level security to the share data in clouds. In particular, we exploit ring signatures to compute verification metadata needed to audit the correctness of shared data. With our mechanism, the identity of the signer on each block in shared data is kept private from public verifiers, who are able to efficiently verify shared data integrity without retrieving the entire file. In addition, our mechanism is able to perform multiple auditing tasks simultaneously instead of verifying them one by one. Our experimental results demonstrate the effectiveness and efficiency of our mechanism when auditing shared data integrity.

Index Terms: Public Auditing, Privacy-Preserving, Shared Data & Cloud Computing

1. Introduction:

Cloud service providers offer users efficient and scalable data storage services with a much lower marginal cost than traditional approaches. It is routine for users to leverage cloud storage services to share data with others in a group, as data sharing becomes a standard feature in most cloud storage offerings, including Dropbox, iCloud and Google Drive.

The integrity of data in cloud storage, however, is subject to skepticism and scrutiny, as data stored in the cloud can easily be lost or corrupted due to the inevitable hardware/software failures and human errors. To make this matter even worse, cloud service providers may be reluctant to inform users about these data errors in order to maintain the reputation of their services and avoid losing profits. Therefore, the integrity of cloud data should be verified before any data utilization, such as search or computation over cloud data.

The traditional approach for checking data correctness is to retrieve the entire data from the cloud, and then verify data integrity by checking the correctness of signatures or hash values of the entire data. Certainly, this conventional approach is able to successfully check the correctness of cloud data. However, the efficiency of using this traditional approach on cloud data is in doubt.

The main reason is that the size of cloud data is large in general. Downloading

the entire cloud data to verify data integrity will cost or even waste user's amounts of computation and communication resources, especially when data have been corrupted in the cloud. Besides, many uses of cloud data do not necessarily need users to download the entire cloud data to local devices. It is because cloud providers, such as Amazon, can offer users computation services directly on large-scale data that already existed in clouds.

Recently, many mechanisms have been proposed to allow not only a data owner itself but also a *public verifier* to efficiently perform integrity checking without downloading the entire data from the cloud, which is referred to as *public auditing* . In these mechanisms, data is divided into many small blocks,

	PDP [9]	WWRL [5]	Oruta
Public Auditing	√	√	√
Data Privacy	×	√	√
Identity Privacy	×	×	√

A A block signed by Alice B A block signed by Bob

Where each block is independently signed by the owner; and a random combination of all the blocks instead of the whole data is retrieved during integrity checking. A public verifier could be a data user who would like to utilize the owner's data via the cloud or a third-party auditor who can provide expert integrity checking services. Moving a step forward, Wang et al. designed an advanced auditing mechanism, so that during public auditing on cloud data, the content of private data belonging to a personal user is not dis-closed to any public verifiers. Unfortunately, current public auditing solutions mentioned above only focus on personal data in the cloud.

We believe that sharing data among multiple users is perhaps one of the most engaging features that motivate cloud storage. Therefore, it is also necessary to ensure the integrity of shared data in the cloud is correct. Existing public auditing mechanisms can actually be extended to verify shared data integrity. However, a new significant privacy issue introduced in the case of shared data with the use of existing mechanisms is the leakage of *identity privacy* to public verifiers.

For instance, Alice and Bob work together as a group and share a file in the cloud. The shared file is divided into a number of small blocks, where each block is independently signed by one of the two users with existing public auditing solutions .Once a block in this shared file is modified by a user, this user needs to sign the new block using his/her private key. Eventually, different blocks are signed by different users due to the modification introduced by these two different users. Then, in order to correctly audit the integrity of the entire data, a public verifier needs to choose the appropriate public key for each block. As a result, this public verifier will inevitably learn the identity of the signer on each block due to the unique binding between an identity and a public key via digital certificates under public key infrastructure.

Failing to preserve identity privacy on shared data during public auditing will reveal significant confidential information to public verifiers. Specifically, after performing several auditing tasks, this public verifier can first learn that Alice may be a

more important role in the group because most of the blocks in the shared file are always signed by Alice; on the other hand, this public verifier can also easily deduce that the eighth block may contain data of a higher value, because this block is frequently modified by the two different users. In order to protect this confidential information, it is essential and critical to preserve identity privacy from public verifiers during public auditing.

In this paper, to solve the above privacy issue on shared data, we propose Oruta, a novel privacy-preserving public auditing mechanism. More specifically, we utilize ring signatures to construct homomorphic authenticators in Oruta, so that a public verifier is able to verify the integrity of shared data without retrieving the entire data while the identity of the signer on each block in shared data is kept private from the public verifier.

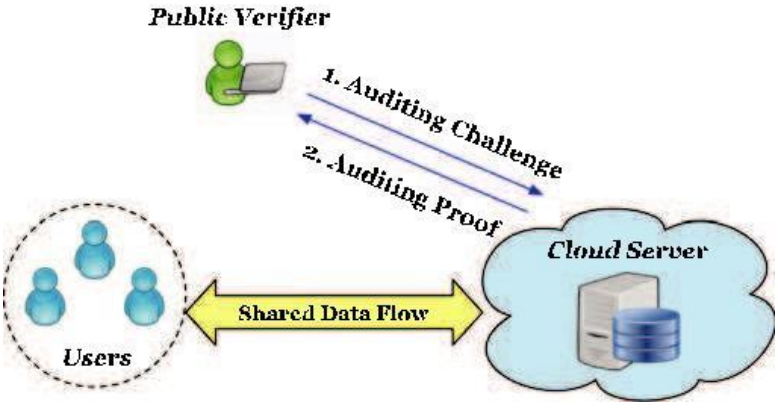
In addition, we further extend our mechanism to support batch auditing, which can perform multiple auditing tasks simultaneously and improve the efficiency of verification for multiple auditing tasks. Meanwhile, Oruta is compatible with random masking, which has been utilized in WWRL and can preserve data privacy from public verifiers. Moreover, we also leverage index hash tables from a previous public auditing solution to support dynamic data. A high-level comparison among Oruta and existing mechanisms is presented.

2. Problem Statement:

2.1 System Model:

The system model in this paper involves three parties: the cloud server, a group of users and a public verifier. There are two types of users in a group: the original user and a number of group users. The original user initially creates shared data in the cloud, and shares it with group users. Both the original user and group users are members of the group. Every member of the group is allowed to access and modify shared data. Shared data and its verification metadata are both 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. After receiving the auditing challenge, the



cloud server responds to the public verifier with an auditing proof of the possession of shared data. 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.

2.2 Threat Model:

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. Second, the cloud service provider may inadvertently corrupt data in its storage due to hardware failures and human errors. Making matters worse, the cloud ser-vice provider is economically motivated, which means it may be reluctant to inform users about such corruption of data in order to save its reputation and avoid losing profits of its services.

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 public verifier, who is only allowed to verify the correctness of shared data integrity, may try to reveal the identity of the signer on each block in shared data based on verification metadata. Once the public verifier reveals the identity of the signer on each block, it can easily distinguish a high-value target from others.

2.3 Design Objectives

Our mechanism, Oruta, should be designed to achieve following properties: (1) *Public Auditing*: A public verifier is able to publicly verify the integrity of shared data without retrieving the entire data from the cloud. (2) *Correctness*: A public verifier is able to correctly verify shared data integrity. (3) *Unforgeability*: Only a user in the group can generate valid verification metadata on shared data.(4) *Identity Privacy*: A public verifier cannot distinguish the identity of the signer on each block in shared data during the process of auditing.

2.4 Possible Alternative Approaches:

To preserve the identity of the signer on each block during public auditing, possible alternative approach is to ask all the users of the group to share a *global private key*. Then, every user is able to sign blocks with this global private key. However, once one user of the group is compromised or leaving the group, a new global private key must be generated and securely shared among the rest of the group, which clearly introduces huge overhead to users in terms of key management and key distribution. While in our solution, each user in the rest of the group can still utilize its own private key for computing verification metadata without generating or sharing any new secret keys.

Another possible approach to achieve identity privacy, is to add a trusted proxy between a group of users and the cloud in the system model. More concretely, each member's data is collected, signed, and uploaded to the cloud by this trusted proxy, then a public verifier can only verify and learn that it is the proxy signs the data, but cannot learn the identities of group members. Yet, the security of this method is threatened by the single point failure of the proxy. Besides, sometimes, not all the group members would like to trust the same proxy for generating signatures and uploading data on their behalf. Utilizing group signatures is also an alternative option to preserve identity privacy. Unfortunately, as shown in our recent work how to design an efficient public auditing mechanism based on group signatures remains open.

Trusted Computing offers another possible alternative approach to achieve the design objectives of our mechanism.

3. Preliminaries:

In this section, we briefly introduce cryptographic primitives and their corresponding properties that we implement in Oruta.

3.1 Ring Signature:

With ring signatures, a verifier is convinced that a signature is computed using one of group members' private keys, but the verifier is not able to deter-mine which one. More concretely, given a ring signature and a group of d users, a verifier cannot distinguish the signer's identity with a probability more than 1=d. This property can be used to preserve the id entity of the signer from a verifier.

The ring signature scheme introduced to constructed on bilinear maps. We will extend this ring signature scheme to construct our public auditing mechanism and Thump Impression technique.

Homomorphic authenticators are basic tools to construct public auditing mechanisms .Besides unforgeability, a homomorphic authenticable signature scheme satisfy block less verifiability and Non-malleability to audit the correctness of shared data in the cloud server.

4. New Ring Signature Scheme:

4.1 Overview:

As we introduced in previous sections, we intend to utilize ring signatures to hide the identity of the signer on each block, so that private and sensitive information of the group is not disclosed to public verifiers. However, traditional ring signatures cannot be directly used into public auditing mechanisms, because these ring signature schemes do not support block less verifiability. Without block less verifiability, a public verifier has to download the whole data file to verify the correctness of shared data, which consumes excessive bandwidth and takes very long verification times.

Therefore, we design a new homomorphic authenticable ring signature (HARS) scheme, which is extended from a classic ring signature scheme. The ring signatures generated by HARS are not only able to preserve identity privacy but also able to support block less verifiability. We will show how to build the privacy-preserving public auditing mechanism for shared data in the cloud based on this new ring signature scheme in the next section.

4.2 Construction of HARS:

HARS contains three algorithms: **Key Gen, Ring Sign** and **Ring Verify.** In KeyGen, each user in the group generates his/her public key and private key. In **Ring Sign,** a user in the group is able to generate a signature on a block and its block identifier with his/her private key and all the group members' public keys. A block identifier is a string that can distinguish the corresponding block from others. A verifier is able to check whether a given block is signed by a group member in **Ring Verify.**

5. Public Auditing Mechanism:

5.1 Overview:

Using HARS and its properties we established in the previous section, we now construct Oruta, a privacy-preserving public auditing mechanism for shared data in the cloud. With Oruta, the public verifier can verify the integrity of shared data without retrieving the entire data. Meanwhile, the identity of the signer on each block in shared data is kept private from the public verifier during the auditing.

5.2 Reduce Signature Storage:

Another important issue we should consider in the construction of Oruta is the size of storage used for ring signatures. According to the generation of ring signatures in HARS, a block m is an element of Z_p and its ring signature contains d elements of G_1, where G_1 is a cyclic group with order p. It means a bit block requires a bit ring signature, which forces users to spend a huge amount of space on storing ring signatures. It will be very frustrating for users, because cloud service providers, such as

Amazon, will charge users based on the storage space they use.

The length of a ring signature is only d=k of the length of a block. Similar methods to reduce the storage space of signatures can also be found in [15]. Generally, to obtain a smaller size of a ring signature than the size of a block, we choose k > d. As a trade-off, the communication cost of an auditing task will be increasing with an increase of k

5.3 Support Dynamic Operations:

To enable each user in the group to easily modify data in the cloud, Oruta should also support dynamic operations on shared data. A dynamic operation includes an insert, delete or update operation on a single block. However, since the computation of a ring signature includes an identifier of a block, traditional methods, which only use the index of a block as its identifier, are not suitable for supporting dynamic operations on shared data efficiently.

The reason is that, when a user modifies a single block in shared data by performing an insert or delete operation, the indices of blocks that after the modified block are all changed, and the changes of these indices require users, who are sharing the data, to re-compute the signatures of these blocks, even though the content of these blocks are not modified.

By utilizing an index hash table, switch is a data structure indexing each block based on its hash value, our mechanism can allow a user to efficiently perform a dynamic operation on a single block, and avoid this type of re-computation on other blocks. Examples of different dynamic operations on shared data with our index hash tables are described in Figs. 5 and 6.

Specifically, the value of r generated by H_2 ensures that each block has a unique identifier. The virtual indices are able to ensure that all the blocks in shared data are in the right order. For example, if $v_i < v_j$, then block m_i is ahead of block m_j in shared data. When shared data is created by the original user.

To support dynamic data without the above assumption, the combination of coding techniques and Oblivious RAM can be utilized as introduced in recent work. Unfortunately, this proposed solution requires much more computation and communication overhead.

5.4 Construction of Oruta:

Now, we present the details of our public auditing mechanism. It includes five algorithms: KeyGen, SigGen, Modify, ProofGen and ProofVerify. In KeyGen, users generate their own public/private key pairs. In SigGen, a user is able to compute ring signatures on blocks in shared data by using its own private key and all the group members' public keys. Each user in the group is able to perform an insert, delete or update operation on a block, and compute the new ring signature on this new block in Modify. ProofGen is operated by a public verifier and the cloud server together to interactively generate a proof of possession of shared data. In ProofVerify, the public verifier audits the integrity of shared data by verifying the proof.

The main reason of this type of re-computation on signatures introduced by dynamic groups is because the generation of a ring signature under our mechanism requires the signer's private key and all the current members' public keys. An interesting problem for our future work will be how to avoid this type of re-computation introduced by dynamic groups while still preserving identity privacy from the public verifier during the process of public auditing on shared data.

5.6 Batch Auditing:

Sometimes, a public verifier may need to verify the correctness of multiple

auditing tasks in a very short time. Directly verifying these multiple auditing tasks separately would be inefficient. By leveraging the properties of bilinear maps, we can further extend Oruta to support batch auditing, which can verify the correctness of multiple auditing tasks simultaneously and improve the efficiency of public auditing.

6. Performance:

6.1 Computation Cost:

During auditing, 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.

6.2Communication Cost:

The Communication cost of Oruta is mainly introduced by two aspects: the auditing challenge and auditing proof. The communication cost of one auditing proof is in bit value.

7. Related Work:

Provable data possession (PDP), proposed by Ateniese et al, allows a verifier to check the correctness of a client's data stored at an untrusted server. By utilizing RSA-based homomorphic authenticators and sampling strategies, the verifier is able to publicly audit the integrity of data without retrieving the entire data, which is referred to as public auditing. Unfortunately, their mechanism is only suitable for auditing the integrity of personal data. Juels and Kaliski defined another similar model called Proofs of Retrievability (POR), which is also able to check the correctness of data on an untrusted server. The original file is added with a set of randomly-valued check blocks called *sentinels*. The verifier challenges the untrusted server by specifying the positions of a collection of sentinels and asking the untrusted server to return the associated sentinel values. Shacham and Waters designed two improved schemes. The first scheme is built from BLS signatures, and the second one is based on pseudo-random functions.

To support dynamic data, Ateniese et al. presented an efficient PDP mechanism based on symmetric keys. This mechanism can support update and delete operations on data, however, insert operations are not available in this mechanism. Because it exploits symmetric keys to verify the integrity of data, it is not public verifiable and only provides a user with a limited number of verification requests. Wang et al. Utilized Merkle Hash Tree and BLS signatures to support dynamic data in a public auditing mechanism. Erway et al. Introduced dynamic provable data possession (DPDP) by using authenticated dictionaries, which are based on rank information. Zhu et al. Exploited the fragment structure to reduce the storage of signatures in their public auditing mechanism. In addition, they also used index hash tables to provide dynamic operations on data. The public mechanism proposed by Wang et al and its journal version are able to preserve users' confidential data from a public verifier by using random maskings. In addition, to operate multiple auditing tasks from different users efficiently, they extended their mechanism to enable batch auditing by leveraging aggregate signatures.

Wang et al leveraged homomorphic tokens to ensure the correctness of erasure codes-based data distributed on multiple servers. This mechanism is able not only to support dynamic data, but also to identify misbehaved servers. To minimize communication overhead in the phase of data repair, Chen et al. also introduced a mechanism for auditing the correctness of data under the multi-server scenario, where these data are encoded by network coding instead of using erasure codes. More recently, Cao et al. constructed an LT codes-based secure and reliable cloud storage mechanism. Compare to previous work this mechanism can avoid high decoding

computation cost for data users and save computation resource for online data owners during data repair.

8. Conclusion:

In this paper, we propose Oruta, a privacy-preserving public auditing mechanism for shared data in the cloud. We utilize ring signatures to construct homomorphic authenticators, so that a public verifier is able to audit shared data integrity without retrieving the entire data, yet it cannot distinguish who is the signer on each block. To improve the efficiency of verifying multiple auditing tasks, we further extend our mechanism to support batch auditing and thump technique. By using thump technique we give high level security for shared data in cloud server. So, the hacker does not able to retrieve the data without the permission of authenticated persons.

We will continue to study for our future work. One of them is traceability, which means the ability for the group manager to reveal the identity of the signer based on verification metadata in some special situations. Since oruta is based on ring signatures, where the identity of the signer is conditionally protected [x1],To the best of our knowledge, designing an efficient public auditing mechanism with the capabilities of preserving identity privacy and supporting traceability is still open situations. Since oruta is based on ring signatures, where the identity of the signer is conditionally protected [x1],To the best of our knowledge, designing an efficient public auditing mechanism with the capabilities of preserving identity privacy and supporting traceability is still open.

9. References:

1. 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.
2. H. Shacham and B. Waters, "Compact Proofs of Retrievability," Proc. 14th Int'l Conf. Theory and Application of Cryptology and Infor-mation Security: Advances in Cryptology (ASIACRYPT '08), pp. 90-107, 2008.
3. C. Erway, A. Kupcu, C. Papamanthou, and R. Tamassia, "Dynamic Provable Data Possession," Proc. 16th ACM Conf. Com-puter and Comm. Security (CCS'09), pp. 213-222, 2009.
4. Q. Wang, C. Wang, J. Li, K. Ren, and W. Lou, "Enabling Public Verifiability and Data Dynamic for Storage Security in Cloud Computing," Proc. 14th European Conf. Research in Computer Secu-
5. C. Wang, Q. Wang, K. Ren, and W. Lou, "Ensuring Data Storage Security in Cloud Computing," Proc. 17th Int'l Workshop Quality of Service (IWQoS'09), pp. 1-9, 2009.
6. S.S Yau, "Dynamic Audit Services for Integrity Verification of Outsourced Storages in Clouds," Proc. ACM Symp. Applied Computing (SAC'11), pp. 1550-1557, 2011.
7. N. Cao, S. Yu, Z. Yang, W. Lou, and Y.T. Hou, "LT Codes-Based Secure and Reliable Cloud Storage Service," Proc. IEEE INFO-COM, 2012.
8. R.L. Rivest, A. Shamir, and Y. Tauman, "How to Leak a Secret," Proc. Seventh Int'l Conf. Theory and Application of Cryptology and Information Security: Advances in Cryptology (ASIACRYPT'01), 552-565, 2001.
9. D. Cash, A. Kupcu, and D. Wichs, "Dynamic Proofs of Retriev-ability via Oblivious RAM," Proc. 32nd Ann. Int'l Conf. Theory and Applications of Cryptographic Techniques: Advances in Cryptology (EUROCRYPT), pp. 279-295, 2013.
10. S. Yu, C. Wang, K. Ren, and W. Lou, "Achieving Secure, Scalable, and Fine-Grained Data Access Control in Cloud Computing," Proc. IEEE INFOCOM, pp. 534-542, 2010.