Secrecy for Shared Data in Cloud using Digital Signatures

Yashwanth Kumar N R
PG Student, Dept of CSE,
P.E.S College of Engineering Mandya,

Chaitra B P
Asst Professor, Dept of CSE
P.E.S College of Engineering Mandya

***Abstract* – Nowadays with the available cloud data services, cloud has become most commonly used place for data to be not only stored, but also shared among multiple users. Unfortunately, the integrity of cloud data is subject to threat 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 and identity privacy to public verifiers. In this paper, we propose a efficient privacy-preserving mechanism that supports public auditing on shared data stored in the cloud. In particular, we use digital 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 profit. 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 users amounts of computation and communication resources, especially when data have been corrupted in the cloud. Besides, many uses of cloud data (e.g., data mining and machine learning) 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 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.

1.1 Purpose:

The purpose of this project is to implement to implement a new privacy preserving public auditing mechanism for the shared data in untrusted cloud, so that third party auditor is able to verify integrity of the data for group of users without retrieving the entire data.

1.2 Objective:

The objective of this project is to implement a auditing mechanism for shared data in cloud which satisfies the following aspects:

- Public Auditing.
- Correctness.
- Unforgeability.
- Identity Privacy.

2 LITERATURE SURVEY

2.1 Existing System:

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, 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 (e.g., researcher) who would like to utilize the owner's data via

the cloud or a third-party auditor (TPA) 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 disclosed 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 motivates 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.

2.1.1 Drawbacks of Existing System:

- Failing to preserve identity privacy on shared data during public auditing will reveal significant confidential information to public verifiers.
- Complex task to download entire data during auditing.

2.2 Proposed System:

In this paper, to solve the above privacy issue on shared data, we propose a novel privacy-preserving public auditing mechanism using digital signatures.

More specifically, we utilize digital signatures to construct homomorphic authenticators, 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.

2.2.1 Advantages of Proposed System:

- A public verifier is able to correctly verify shared data integrity.
- A public verifier cannot distinguish the identity of the signer on each block in shared data during the process of auditing.
- The ring signatures generated for not only able to preserve identity privacy but also able to support block less verifiability.

3 SYSTEM OVERVIEW:

3.1 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 (i.e., signatures) 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.

3.2 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 share d data. Second, the cloud service provider may inadvertently corrupt data in its storage due to hardware failures and human errors. Making matters worse, the c loud ser-vic e provider is economic all y motivated, which means it may be reluctant to inform users about such corruption of data in order t o save its reputation and avoid losing profits of its services.

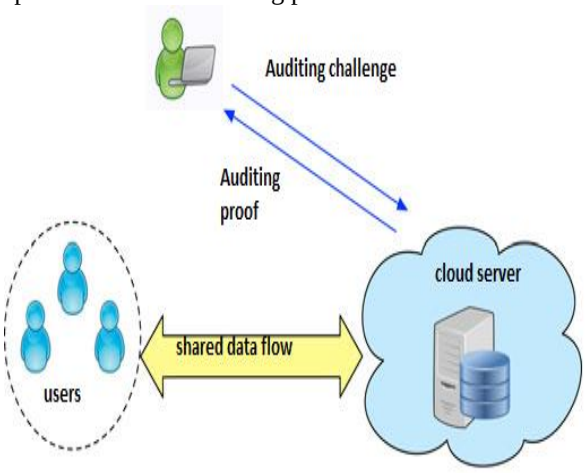

Fig 1: System model including cloud server, group of users and a public verifier.

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.

3.3 *Design Objectives*

Our mechanism 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 (i.e., signatures) 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.

4 DIGITAL SIGNATURE SCHEMA

4.1 *Overview*

As we introduced in previous sections, we intend to utilize digital 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 digital signatures cannot be directly used into public auditing mechanisms, because these 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 digital signature (HADS) scheme, which is extended from a classic ring signature scheme. The digital signatures generated by HADS are not only able to preserve identity privacy but also able to support block less verifiability.

4.2 *Construction of HADS*

HADS contains three algorithms: KeyGen, DigSign and DigVerify. In KeyGen, each user in the group generates his/her public key and private key. In DigSign, 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 DigVerify. Details of this scheme are described in Fig. 2.

5 PUBLIC AUDITING MECHANISM:

5.1 *Overview*

Using HADS and its properties we established in the previous section, we now construct a privacy-preserving public auditing mechanism for shared data in the cloud. With this mechanism, 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 *Construction Of Auditing Mechanism*

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 (either the original user or a group user) is able to compute digital 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.

Note that for the ease of understanding, we first assume the group is static, which means the group is pre-defined before shared data is created in the cloud and the membership of the group is not changed during data sharing. Specifically, before the original user outsources shared data to the cloud, he/she decides all the group members. We will discuss the case of dynamic groups now.

Dynamic Groups- We now discuss the scenario of dynamic groups under our proposed mechanism. If a new user can be added in the group or an existing user can be revoked from the group, then this group is denoted as a dynamic group. To support dynamic groups while still allowing the public verifier to perform public auditing, all the digital signatures on shared data need to be re-computed with the signer's private key and all the current users' public keys when the membership
of the group is changed.

Let $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ be multiplicative cyclic groups of order p, g_1 and g_2 be generators of $\mathbb{G}_1$ and $\mathbb{G}_2$ respectively. Let $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ be a bilinear map, and $\psi : \mathbb{G}_2 \to \mathbb{G}_1$ be a computable isomorphism with $\psi(g_2) = g_1$. There is a public map-to-point hash function H_1: $\{0,1\}^* \to \mathbb{G}_1$, which can map a string $\{0,1\}^*$ into an element of $\mathbb{G}_1$ (i.e., an point on an elliptic curve). The total number of users in the group is d. The global parameters are $(e, \psi, p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, H_1, d)$.

KeyGen. For a user u_i, he/she randomly picks $x_i \xleftarrow{R} \mathbb{Z}_p$ and computes $w_i = g_2^{x_i} \in \mathbb{G}_2$. Then, user u_i's public key is $\mathbf{pk}_i = w_i$ and his/her private key is $\mathbf{sk}_i = x_i$.

RingSign. Given all the d users' public keys $(\mathbf{pk}_1, ..., \mathbf{pk}_d) = (w_1, ..., w_d)$, a block $m \in \mathbb{Z}_p$, the identifier of this block id and the private key $\mathbf{sk}_s$ for some s, user u_s randomly chooses $a_i \in \mathbb{Z}_p$ for all $i \neq s$, where $i \in [1, d]$, and let $\sigma_i = g_1^{a_i}$. Then, he/she computes

$$\beta = H_1(id) g_1^m \in \mathbb{G}_1, \quad (1)$$

and sets

$$\sigma_s = \left(\frac{\beta}{\psi(\prod_{i \neq s} w_i^{a_i})} \right)^{1/x_s} \in \mathbb{G}_1. \quad (2)$$

The ring signature of block m is $\boldsymbol{\sigma} = (\sigma_1, ..., \sigma_d) \in \mathbb{G}_1^d$.

RingVerify. Given all the d users' public keys $(\mathbf{pk}_1, ..., \mathbf{pk}_d) = (w_1, ..., w_d)$, a block m, an identifier id and a ring signature $\boldsymbol{\sigma} = (\sigma_1, ..., \sigma_d)$, a verifier first computes $\beta = H_1(id) g_1^m \in \mathbb{G}_1$, and then checks

$$e(\beta, g_2) \stackrel{?}{=} \prod_{i=1}^{d} e(\sigma_i, w_i). \quad (3)$$

If the above equation holds, then the given block m is signed by one of these d users in the group. Otherwise, it is not.

Fig 2: Details of HADS

The main reason of this type of re-computation on signatures introduced by dynamic groups, is because the generation of a digital 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.

5.3 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 RELATED WORK

Provable data possession(PDP),proposed by Ateniese et al.[10], 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[7] 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 [2] designed two improved schemes. The first scheme is built from BLS signatures [6], and the second one is based on pseudo-random functions.

To support dynamic data, Ateniese et al. [8] 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. [3] utilized Merkle Hash Tree and BLS signatures[6] 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. [4] 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 Wanget al. [1] 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 [5].

7 CONCLUSION AND FUTURE WORK

In this paper, we propose 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.

There are two interesting problems we will continue to study for our future work. One of them is traceability, which means the ability for the group manager (i.e., the original user) to reveal the identity of the signer based on verification metadata in some special situations. Since this is based on digital signatures, where the identity of the signer is unconditionally protected, the current design of ours does not support traceability. 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. Another problem for our future work is how to prove data freshness (prove the cloud possesses the latest version of shared data) while still preserving identity privacy.

REFERENCES

1. C. Wang, Q. Wang, K. Ren, and W. Lou, "Privacy-Preserving Public Auditing for Data Storage Security in Cloud Computing," Proc. IEEE INFOCOM, pp. 525-533, 2010
2. 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.
3. 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 Security (ESORICS'09), pp. 355-370, 2009.
4. Y. Zhu, H. Wang, Z. Hu, G.-J. Ahn, H. Hu, and 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.
5. D. Boneh, C. Gentry, B. Lynn, and H. Shacham, "Aggregate and Verifiably Encrypted Signatures from Bilinear Maps," Proc. 22nd Int'l Conf. Theory and Applications of Cryptographic Techniques: Advances in Cryptology (EUROCRYPT'03), pp. 416-432, 2003.
6. D. Boneh, B. Lynn, and H. Shacham, "Short Signatures from the Weil Pairing," Proc. Seventh Int'l Conf. Theory and Application of Cryptology and Information Security: Advances in Cryptology (ASIACRYPT'01), pp. 514-532, 2001.
7. A. Juels and B.S. Kaliski, "PORs: Proofs of Retrievability for Large Files," Proc. 14th ACM Conf. Computer and Comm. Security (CCS'07), pp. 584-597, 2007.
8. G. Ateniese, R.D. Pietro, L.V. Mancini, and G. Tsudik, "Scalable and Efficient Provable Data Possession," Proc. Fourth Int'l Conf. Security and Privacy in Comm. Networks (SecureComm'08), 2008.
9. B. Wang, B. Li, and H. Li, "Oruta: Public Auditing for Data in the Cloud," Proc. IEEE Fifth Int'l Conf. Cloud Computing, pp. 295-302, 2012.
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.