www.ijiarec.com

ISSN: 2348-2079 MAY-2015

International Journal of Intellectual Advancements and Research in Engineering Computations

CLOUD DATA SHARING SERVICES WITH PUBLIC VERIFIER

[1]M. Nithiya, [2]A. N. Karthikeyan.

ABSTRACT

Data auditing process is performed in the centralized verification point called public verifier or Third Party Auditor (TPA). "One Ring to RUle Them All" (Oruta) scheme is used for privacy-preserving public auditing process. In oruta homomorphic authenticators are constructed using Ring Signatures. Ring signatures are used to compute verification metadata needed to audit the correctness of shared data. The identity of the signer on each block in shared data is kept private from public verifiers. Homomorphic authenticable ring signature (HARS) scheme is applied to provide identity privacy with block less verification. Batch auditing mechanism supports to perform multiple auditing tasks simultaneously. Oruta is compatible with random masking to preserve data privacy from public verifiers. Dynamic data management process is handled with index hash tables. Traceability is not supported in oruta scheme. Data dynamism sequence is not managed by the system. The system obtains high computational overhead. The multi user data auditing scheme is enhanced with data dynamism and batch auditing support. Traceability features are provided with identity privacy. Group manager or data owner can be allowed to reveal the identity of the signer based on verification metadata. Data version management mechanism is integrated with the system.

INTRODUCTION

Nowadays, as the cloud offers data storage services with much lower prices than the cost of maintaining data on personal devices, people tend to outsource the hosting of their data to the cloud. By enjoying such storage services in the cloud, data owners are able to freely access their outsourced data on different devices and locations and easily share their data with others. Although cloud providers have designed a series of security protections for these data storage services, casting the image of a more reliable and secure place to store data than personal devices, the integrity of data stored in the cloud may still be in doubt due to the existence of hardware/software failures and human errors. For example, Dropbox, a well-known cloud-based data storage service with over 100 million users, accidentally allowed anybody to access Dropbox accounts without passwords for several hours after an unsuccessful code update in June 2011.

To efficiently audit data integrity in an untrusted cloud, many mechanisms have been proposed [9], [5], [1], [10], [4], [6]. One of the most attractive features of these works is allowing not only the data owner herself but also a public verifier, such as a data user who would like to utilize cloud data, to verify the integrity of cloud data without retrieving the entire data from the cloud, referred to as public auditing. Another common feature of these previous works is that choosing the correct public key of the data owner during the verification on cloud data integrity is based on the security of Public Key Infrastructure (PKI).

In traditional PKI, the assurance of the binding between an owner's identity and her public/private key is delivered by the Certificate Authority (CA) and certificates issued by the CA. Although PKI has been widely used in the construction of public key cryptography, it still faces

Author for Correspondence:

[1]Final year ME., Mahendra Institute of Teconolgy, Mahendhrapuri, Tamilnadu, India.

[2]AP/CSE, Mahendra Institute of Teconolgy, Mahendhrapuri, Tamilnadu, India.

many security risks. One of the most fundamental issues is the management of certificates, including distribution, storage, revocation and verification. For example, a certificate can only be trusted by users if the root certificate of this certificate is trustworthy; however, since the root certificate is self-signed by a CA itself, to determine the trustworthiness of this root certificate in the first place is not an easy task, even for a security expert. It is an even harder — and sometimes confusing — process for the general public, who have no special knowledge of cryptography and security. All they can do is perhaps to click the button shows Accept and install a so-called "trustworthy" certificate anyway.

Considering these security risks, the certificate of a data owner that a public verifier obtains may not be trustworthy and the public key used for verifying cloud data integrity may not even belong to the expected data owner. In this case, even the verification result is positive, the cloud data that a public verifier intend to utilize may not be actually signed by the data owner herself. Note that some symmetric key based solutions can certainly be leveraged to verify the correctness of data stored in an untrusted cloud without involving certificates. They are not public verifiable. Therefore, how to avoid managing certificates at public verifiers while still designing a public key-based mechanism to securely and efficiently audit data integrity in the cloud is a necessary task.

To avoid managing certificates in a public auditing mechanism, utilizing Identity-Based Signatures (IBS) seems to be an option in the first place. Unfortunately, IBS has an inherent drawback — the key escrow problem. By leveraging the existing technique of certificateless signatures (CLS), a public verifier should be able to audit data integrity without managing certificates or suffering the key escrow problem. In particular, a public verifier should be able to leverage the owner's identity, such as her name or email address, to ensure the right public key of this owner is used during the auditing of cloud data integrity. The main challenge of building a certificateless public auditing mechanism in the cloud is that, traditional certificateless signature schemes [7] cannot satisfy one of the most significant features that a public auditing mechanism should be capable of — verifying the integrity without downloading the entire data, which is referred to as blockless verifiability.

RELATED WORK

Ateniese et al. are the first to consider public auditability in their "provable data possession" (PDP) model for ensuring possession of data files on untrusted storages. They utilize the RSA-based homomorphic linear authenticators for auditing outsourced data and suggest randomly sampling a few blocks of the file. Among their two proposed schemes, the one with public auditability exposes the linear combination of sampled blocks to external auditor. When used directly, their protocol is not provably privacy preserving and thus may leak user data information to the external auditor. Juels et al. describe a "proof of retrievability" (PoR) model, where spot checking and error-correcting codes are used to ensure both "possession" and "retrievability"of data files on remote archive service systems. The number of audit challenges a user can perform is fixed a priori and public auditability is not supported in their main scheme. Although they describe a straightforward Merkle-tree construction for public PoRs, this approach only works with encrypted data. Later, Bowers et al. [8] propose an improved framework for POR protocols that generalizes Juels' work. Dodis et al. also give a study on different variants of PoR with private auditability. Shacham and Waters design an improved PoR scheme built from BLS signatures with proofs of security in the security model. Similar to the construction, they use publicly verifiable homomorphic linear authenticators that are built from provably secure BLS signatures. Again, their approach is not privacy preserving due to the same reason. Shah et al. propose introducing a TPA to keep online storage honest by first encrypting the data then sending a number of precomputed symmetric-keyed hashes over the encrypted data to the auditor. The auditor verifies the integrity of the data file and the server's possession of a previously committed decryption key. This scheme only works for

encrypted files, requires the auditor to maintain state and suffers from bounded usage, which potentially brings in online burden to users when the keyed hashes are used up.

Dynamic data have also attracted attentions in the recent literature on efficiently providing the integrity guarantee of remotely stored data. Ateniese et al. is the first to propose a partially dynamic version of the prior PDP scheme, using only symmetric key cryptography but with a bounded number of audits. In [2], Wang et al. consider a similar support for partially dynamic data storage in a distributed scenario with additional feature of data error localization. In a subsequent work, Wang et al. propose to combine BLS-based HLA with MHT to support fully data dynamics. Concurently, Erway et al. [3] develop a skip list based scheme to also enable provable data possession with full dynamics support. The verification in both protocols requires the linear combination of sampled blocks as an input and thus does not support privacy-preserving auditing.

In other related work, Sebe et al. thoroughly study a set of requirements which ought to be satisfied for a remote data possession checking protocol to be of practical use. Their proposed protocol supports unlimited times of file integrity verifications and allows preset tradeoff between the protocol running time and the local storage burden at the user. Schwarz and Miller propose the first study of checking the integrity of the remotely stored data across multiple distributed servers. Their approach is based on erasure-correcting code and efficient algebraic signatures, which also have the similar aggregation property as the homomorphic authenticator utilized in our approach. Curtmola et al. aim to ensure data possession of multiple replicas across the distributed storage system. They extend the PDP scheme to cover multiple replicas without encoding each replica separately, providing guarantee that multiple copies of data are actually maintained. Bowers et al. utilize a two-layer erasure-correcting code structure on the remotely archived data and extend their POR model [8] to distributed scenario with high-data availability assurance. While all the above schemes provide methods for efficient auditing and provable assurance on the correctness of remotely stored data, almost none of them necessarily meet all the requirements for privacy-preserving public auditing of storage. Moreover, none of these schemes consider batch auditing, while our scheme can greatly reduce the computation cost on the TPA when coping with a large number of audit delegations.

CLOUD DATA VERIFICATION METHODS

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 of the entire data. Certainly, this conventional approach is able to successfully check the correctness of cloud data. 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 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.

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 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. 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. 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.

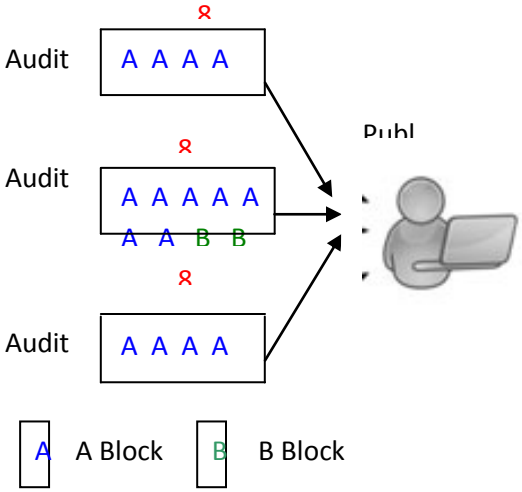

Fig. 3.1. Data Verification Methods in Clouds

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 (PKI).

Failing to preserve identity privacy on shared data during public auditing will reveal significant confidential information to public

verifiers. Specifically, as shown in Fig. 3.1, 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 these confidential information, it is essential and critical to preserve identity privacy from public verifiers during public auditing.

TABLE 3.1: COMPARISON AMONG DIFFERENT MECHANISMS

	PDA	WWRL	Oruta
Public Auditing	✓	✓	✓
Data Privacy	×	✓	✓
Identity Privacy	×	×	✓

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.

PROBLEM DEFINITION

"One Ring to RUle Them All" (Oruta) scheme is used for privacy-preserving public auditing process. In oruta homomorphic authenticators are constructed using Ring Signatures. Ring signatures are used to compute verification metadata needed to audit the correctness of shared data. The identity of the signer on each block in shared data is kept private from public verifiers. Homomorphic authenticable ring signature (HARS) scheme is applied to provide identity privacy with blockless verification. Batch auditing mechanism supports to perform multiple auditing tasks simultaneously. Oruta is compatible with random masking to preserve data privacy from public verifiers. Dynamic data management process is handled with index hash tables. The following problems are identified from the existing system. They are traceability is not supported, data dynamism sequence is not managed and high computational overhead.

ENCRYPTED CLOUD DATA VERIFICATION SCHEMES

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.

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.

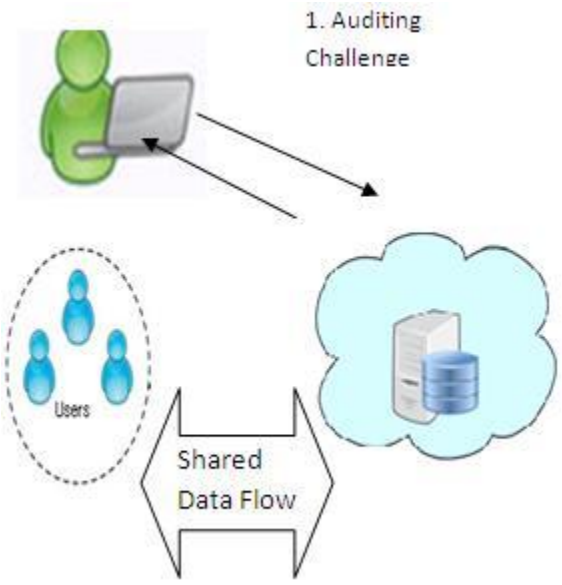

Fig. 5.1. Encrypted Cloud Data Verification Schemes

Making matters worse, the cloud service 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 [11]. 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.

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) Unforgetability: 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.

RING SIGNATURES

The concept of ring signatures was first proposed by Rivest et al. in 2001. 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 determine 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 identity of the signer from a verifier. The ring signature scheme introduced by Boneh et al. is constructed on bilinear maps. We will extend this ring signature scheme to construct our public auditing mechanism.

HOMOMORPHIC AUTHENTICATORS

Homomorphic authenticators are basic tools to construct public auditing mechanisms. Besides unforgeability, a homomorphic authenticable signature scheme, which denotes a homomorphic authenticator based on signatures, should also satisfy the following properties:

Let (pk, sk) denote the signer's public/private key pair, σ_1 denote a signature on block m_1 € Zp, σ_2 denote a signature on block m_2 € Z_p. Blockless verifiability: Given σ_1 and σ_2, two random values α_1, α_2 ε Z_p and a block m'= $\alpha_1 m_1 + \alpha_2 m_2$ € Z_p, a verifier is able to check the correctness of block m0 without knowing block m_1 and m_2. Non-malleability: Given $\sigma_1 1$ and σ_2, two random values α_1, α_2 € Z_p and a block m'= $\alpha_1 m_1 + \alpha_2 m_{2\varepsilon} Z_p$, a user, who does not have private key s_k, is not able to generate a valid signature σ' on block m' by linearly combining signature σ_1 and σ_2. Blockless verifiability allows a verifier to audit the correctness of data stored in the cloud server with a special block, which is a linear combination of all the blocks in data. If the integrity of the combined block is correct, then the verifier believes that the integrity of the entire data is correct. In this way, the verifier does not need to download all the blocks to check the integrity of data. Non-malleability indicates that an adversary cannot generate valid signatures on arbitrary blocks by linearly combining existing signatures.

ENHANCED RING SIGNATURE SCHEME

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. Traditional ring signatures cannot be directly used into public auditing mechanisms, because these ring signature schemes do not support blockless verifiability. Without blockless 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. 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 blockless 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.

CONSTRUCTION OF HARS

HARS contains three algorithms: KeyGen, RingSign and RingVerify. In KeyGen, each user in the group generates his/her public key and private key. In RingSign, 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 RingVerify. Details of this scheme are described.

PUBLIC VERIFIER BASED CLOUD DATA SHARING

The proposed system is designed to perform public data verification with privacy. Traceability features are provided with identity privacy. Group manager or data owner can be allowed to reveal the identity of the signer based on verification metadata. Data version management mechanism is integrated with the system. The system is divided in to five major modules. They are data center, third party auditor, client data dynamism handler and batch auditing. The cloud data center manages the shared data values. Auditing operations are initiated by the Third Party Auditor. Client application is designed to manage data upload and download operations. Data update operations are managed under data dynamism module. Batch auditing is designed for multi user data verification process.

The data center application is designed to allocate storage space for the data providers. Data

center maintains data files for multiple providers. Different sized storage area is allocated for the data providers. Data files are delivered to the clients. The Third Party Auditor (TPA) maintains the signature for shared data files. TPA performs the public data verification for data providers. Data integrity verification is performed is using Secure Hashing Algorithm (SHA). Homomorphic linear authenticator and random masking techniques are used for privacy preservation process.

The client application is designed to access the hard data values. The cloud user initiates the download process. Data access information is updated to the data center. Data center transfers the data as blocks. Shared data values are managed with blocks. Block update and delete operations are handled with signature update process. Block insertion operations are also supported in data dynamism process. Block signatures are also updated in data dynamism process. Data integrity verification is carried out under auditing process. Batch auditing is applied to perform simultaneous data verification process. Batch auditing is tuned for multi user environment. Data dynamism is integrated with batch auditing process.

CONCLUSION

Multi user data auditing scheme verifies the data for the single data owner with multiple users' environment. Oruta (One Ring to Rule Them All) scheme is used to support privacy ensured data verification process. Data dynamism and batch auditing are supported in Oruta. Oruta scheme is enhanced with Traceability and Data freshness features. Privacy ensured data verification is performed. Simultaneous data verification scheme is provided in the system. Computational and communication cost is reduced by the system. The system supports data dynamism for secured cloud storage environment. Traceability and version management mechanism is integrated with the system.

REFERENCES

[1]. N. Cao and Y. T. Hou, "LT Codes-based Secure and Reliable Cloud Storage Service," in the Proceedings of IEEE INFOCOM, 2012.

[2]. C. Wang, Q. Wang and W. Lou, "Towards Secure and Dependable Storage Services in Cloud Computing," IEEE Trans. Service Computing, Apr.-June 2012.

[3]. C. Erway and R. Tamassia, "Dynamic Provable Data Possession," Proc. ACM Conf. Computer and Comm. Security, 2009.

[4]. B.Wang, B. Li and H. Li, "Public Auditing for Shared Data with Efficient User Revoation in the Cloud," in the Proceedings of IEEE, 2013.

[5]. B. Wang, H. Li and M. Li, "Privacy-Preserving Public Auditing for Shared Cloud Data Supporting Group Dynamics," in the Proceedings of IEEE, 2013.

[6]. [6] B. Wang, M. Li and H. Li, "Storing Shared Data on the Cloud via Security-Mediator," in the Proceedings of ICDCS'13, 2013.

[7]. L. Zhang and F. Zhang, "A New Certificateless Aggregate Signature Scheme," Computer Communications, 2009.

[8]. K.D. Bowers, "Proofs of Retrievability: Theory and Implementation," Proc. ACM Workshop Cloud Computing Security, 2009.

[9]. B. Chen and R. Burns, "Remote Data Checking for Network Coding-based Distributed Stroage Systems," ACM, 2010.

[10]. Wang and Li, "Oruta: Privacy-Preserving Public Auditing for Shared Data in the Cloud," in the Proceedings of IEEE Cloud 2012, 2012.

[11]. Nayak, "Decentralized Access Control with Anonymous Authentication of Data Stored in Clouds", IEEE Transactions On Parallel And Distributed Systems, February 2014.