Journal of Computer Applications ISSN: 0974 – 1925,
Volume-5, Issue EICA2012-1, February 10, 2012

Data Dynamics for Storage Security and Public Auditability in Cloud Computing

Kayalvizhi S
M.E - Software Engineering
Jayaram College of Engineering and Technology
Anna University, Trichy
Pagalavadi, Tamilnadu State 621014,India
E-mail:kayalsada@yahoo.co.in

Jagadeeswari S, M.E.,
Asst Professor, Department of Computer Science
Jayaram College of Engineering and Technology
Anna University, Trichy
Pagalavadi, Tamilnadu State 621014,India
E-mail:jaga_sankaran123@yahoo.com

Abstract - Cloud Computing has been envisioned as the next-generation architecture of IT Enterprise. It moves the application software and databases to the centralized large data centers, where the management of the data and services may not be fully trustworthy. The unique paradigm brings about many new security challenges, which have not been well understood. The work studies the problem of ensuring the integrity of data storage in Cloud Computing. In particular, consider the task of allowing a third party auditor (TPA), on behalf of the cloud client, to verify the integrity of the dynamic data stored in the cloud. The introduction of TPA eliminates the involvement of the client through the auditing of whether his data stored in the cloud is indeed intact, which can be important in achieving economies of scale for Cloud Computing. The support for data dynamics via the most general forms of data operation, such as block modification, insertion and deletion, is also a significant step toward practicality, since services in Cloud Computing are not limited to archive or backup data only. While prior works on ensuring remote data integrity often lacks the support of either public auditability or dynamic data operations, it achieves both. It first identify the difficulties and potential security problems of direct extensions with fully dynamic data updates from prior works and then show how to construct an elegant verification scheme for the seamless integration of these two salient features in the protocol design. In particular, to achieve efficient data dynamics, to improve the existing proof of storage models by manipulating the classic Merkle Hash Tree construction for block tag authentication. To support efficient handling of multiple auditing tasks, and further explore the technique of bilinear aggregate signature to extend the main result into a multi-user setting, where TPA can perform multiple auditing tasks simultaneously. Extensive security and performance analysis show that the proposed schemes are highly efficient and provably secure.

Keywords: - *Data storage, public auditability, data dynamics, cloud computing.*

I. INTRODUCTION

Several trends are opening up the era of Cloud Computing, which is an Internet-based development and use of computer technology. The ever cheaper and more powerful processors, together with the "software as a service" (SaaS) computing architecture, are transforming data centers into pools of computing service on a huge scale. Meanwhile, the increasing network bandwidth and reliable yet flexible network connections make it even possible that clients can now subscribe high quality services from data and software that reside solely on remote data centers. Although envisioned as a promising service platform for the Internet, the new data storage paradigm in "Cloud" brings about many challenging design issues which have profound influence on the security and performance of the overall

system. One of the biggest concerns with cloud data storage is that of data integrity verification at untrusted servers.What is more serious is that for saving money and storage space the service provider might neglect to keep or deliberately delete rarely accessed data files which belong to an ordinary client. Consider the large size of the outsourced electronic data and the client's constrained resource capability, the core of the problem can be generalized as how can the client find an efficient way to perform periodical integrity verifications without the local copy of data files. Considering the role of the verifier in the model, all the schemes presented before fall into two categories: private auditability and public auditability. Although schemes with private auditability can achieve higher scheme efficiency, public auditability allows anyone, not just the client (data owner), to challenge the cloud server for correctness of data storage while keeping no private information. Then, clients are able to delegate the evaluation of the service performance to an independent third party auditor (TPA), without devotion of their computation resources. In the cloud, the clients themselves are unreliable or may not be able to afford the overhead of performing frequent integrity checks. Thus, for practical use, it seems more rational to equip the verification protocol with public auditability, which is expected to play a more important role in achieving economies of scale for Cloud Computing. Moreover, for efficiency consideration, the outsourced data themselves should not be required by the verifier for the verification purpose. Another major concern among previous designs is that of supporting dynamic data operation for cloud data storage applications. In Cloud Computing, the remotely stored electronic data might not only be accessed but also updated by the clients, e.g., through block modification, deletion and insertion, etc. Unfortunately, the state-of the-art in the context of remote data storage mainly focus on static data files and the importance of the dynamic data updates has received limited attention so far . Moreover, as will be show later, the direct extension of the current provable data possession (PDP) or proof of retrievability (PoR) schemes to support data dynamics may lead to security loopholes. In view of the key role of public auditability and data dynamics for cloud data storage, To propose an efficient construction for the seamless integration of these two components in the protocol design.

The contribution can be summarized as follows:

1) To motivate the public auditing system of data storage security in Cloud Computing, and propose a protocol supporting for fully dynamic data operations, especially to support block insertion, which is missing in most existing schemes;
2) To extend the scheme to support scalable and efficient public auditing in Cloud Computing. In particular, the scheme achieves batch auditing where multiple delegated auditing tasks from different user can be performed simultaneously by the TPA.
3) To prove the security of the proposed construction and justify the performance of the scheme through concrete implementation and comparisons with the state-of-the-art.

II. RELATED WORK

Recently, much of growing interest has been pursued in the context of remotely stored data verification. Ateniese et al. [1] are the first to consider public auditability in their defined "provable data possession"(PDP) model for ensuring possession of files on untrusted storages. In their scheme, utilize RSA based homomorphic tags for auditing outsourced data, thus public auditability is achieved. However, Ateniese et al. do not consider the case of dynamic data storage, and the direct extension of their scheme from static data storage to dynamic case may suffer design and security problems. In their subsequent work [2], Ateniese et al. propose a dynamic version of the prior PDP scheme. However, the system imposes a priori bound on the number of queries and does not support fully dynamic data operations, i.e., it only allows very basic block operations with limited functionality, and block insertions cannot be supported. In [20], Wang et al. consider dynamic data storage in a distributed scenario, and the proposed challenge-response protocol can both determine the data correctness and locate possible errors. Similar to [2], they only consider partial support for dynamic data operation. Juels et al. [10] 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 archive service systems. Specifically, some special blocks called "sentinels" are randomly embedded into the data file F for detection purpose, and F is further encrypted to protect the positions of these special blocks. However, like [2], the number of queries a client can perform is also a fixed priori, and the introduction of pre-computed "sentinels" prevents the development of realizing dynamic data updates. In addition, public auditability is not supported in their scheme. Shacham et al. [16] design an improved PoR scheme with full proofs of security in the security model defined in [10]. They use publicly verifiable homomorphic authenticators built from BLS signatures [4], based on which the proofs can be aggregated into a small authenticator value, and public retrievability is achieved. Still, the authors only consider static data files. Erway et al. [9] was the first to explore constructions for dynamic provable data possession. They extend the PDP model in [1] to support provable updates to stored data files using rank-based authenticated skip lists. The scheme is essentially a fully dynamic version of the PDP solution. To support updates, especially for block insertion, they eliminate the index information in the "tag" computation in Ateniese's PDP model [1] and employ authenticated skip list data structure to authenticate the tag information of challenged or updated blocks first before the verification procedure. However, the efficiency of their scheme remains unclear. Although the existing schemes aim at providing integrity verification for different data storage systems, the problem of supporting both public auditability and data dynamics has not been fully addressed. How to achieve a secure and efficient design to seamlessly integrate these two important components for data storage service remains an open challenging task in Cloud Computing. Two basic solutions (i.e., the MAC-based and signaturebased schemes) for realizing data auditability and discuss their demerits in supporting public auditability and data dynamics. Secondly, generalize the support of data dynamics to both proof of retrievability (PoR) and provable data possession (PDP) models and discuss the impact of dynamic data operations on the overall system efficiency both. In particular, emphasize that while dynamic data updates can be performed efficiently in PDP models more efficient protocols need to be designed for the update of the encoded files in PoR models.

III. PROBLEM STATEMENT

A System Model

A representative network architecture for cloud data storage is illustrated. Three different network entities can be identified as follows:

- Client: an entity, which has large data files to be stored in the cloud and relies on the cloud for data maintenance and computation, can be either individual consumers or organizations;
- Cloud Storage Server (CSS): an entity, which is managed by Cloud Service Provider (CSP), has significant storage space and computation resource to maintain the clients' data.
- Third Party Auditor (TPA): an entity, which has expertise and capabilities that clients do not have, is trusted to assess and expose risk of cloud storage services on behalf of the clients upon request.

B Security Model

The security model defined, that the checking scheme is secure if (i) there exists no polynomial time algorithm that can cheat the verifier with non negligible probability; (ii) there exists a polynomial time extractor that can recover the original data files by carrying out multiple challenges responses. The client or TPA can periodically challenge the storage server to ensure the correctness of the cloud data, and the original files can be recovered by interacting with the server. It can also define the correctness and soundness of their scheme: the scheme is correct if the verification algorithm accepts when interacting with the valid prover (e.g., the server returns a valid response) and it is sound if any cheating server that convinces the client it is storing the data file is actually storing that file. Note that in the "game" between the adversary and the client, the adversary has full access to the information stored in the server, i.e., the adversary can play the part of the prover (server). The goal of the adversary is to cheat the verifier successfully, i.e., trying to generate valid responses and pass the data verification without being detected. The security model has subtle but crucial difference from that of the existing PDP or PoR models in the verification process. As mentioned above, these schemes do not consider dynamic data operations, and the block insertion cannot be supported at all.

The construction of the signatures is involved with the file index information i. Therefore, once a file block is inserted, the computation overhead is unacceptable since the signatures of all the following file blocks should be re-computed with the new indexes. To deal with the limitation, it remove the index information i in the computation of signatures and use $H(m_i)$ as the tag for block mi instead of H(name||i) or h(v||i) , so individual data operation on any file block will not affect the others. Recall that in existing PDP or PoR models, H(name||i) or h(v||i) should be generated by the client in the verification process. However, in the new construction the client has no capability to calculate $H(m_i)$ without the data information. In order to achieve the blockless verification, the server should take over the job of computing $H(m_i)$ and then return it to the prover. The consequence of the variance will lead to a serious problem: it will give the adversary more opportunities to cheat the prover by manipulating $H(m_i)$ or m_i. Due to the construction, the security model differs from that of the PDP or PoR models in both the verification and the data updating process. Specifically, the tags in the scheme should be authenticated in each protocol execution other

than calculated or pre-stored by the verifier. In the following descriptions, it use server and prover (or client, TPA and verifier) interchangeably.

C Design Goals

Design goals can be summarized as the following:

- Public auditability for storage correctness assurance: to allow anyone, not just the clients who originally stored the file on cloud servers, to have the capability to verify the correctness of the stored data on demand.
- Dynamic data operation support: to allow the clients to perform block-level operations on the data files while maintaining the same level of data correctness assurance. The design should be as efficient as possible so as to ensure the seamless integration of public auditability and dynamic data operation support.
- Blockless verification: no challenged file blocks should be retrieved by the verifier (e.g., TPA) during verification process for efficiency concern.

IV. THE PROPOSED SCHEME

A Notation and Preliminaries

Bilinear Map: A bilinear map is a map e : $G \times G \rightarrow G_T$ where G is a Gap Diffie-Hellman (GDH) group and G_T is another multiplicative cyclic group of prime order p with the following properties : (i) Computable: there exists an efficiently computable algorithm for computing e; (ii) Bilinear: for all h_1, h2 □ G and a, b □ Zp, $e(h_1^a , h_2^b) = e(h_1, h_2)ab$; (iii) Non-degenerate: $e(g, g) \neq 1$, where g is a generator of G.

Merkle Hash Tree**:** A Merkle Hash Tree (MHT) is a wellstudied authentication structure,which is intended to efficiently and securely prove that a set of elements are undamaged and unaltered. It is constructed as a binary tree where the leaves in the MHT are the hashes of authentic data values.

B Definition

(pk, sk) ← KeyGen(1^k). This probabilistic algorithm is run by the client. It takes as input security parameter 1k, and returns public key pk and private key sk. (φ, $sig_{sk}(H(R))$) ← SigGen(sk, F). This algorithm is run by the client. It takes as input private key sk and a file F which is an ordered collection of blocks $\{m_i\}$, and outputs the signature set φ, which is an ordered collection of signatures $\{\sigma_i\}$ on $\{m_i\}$. It also outputs metadata-the signature $sig_{sk}(H(R))$ of the root R of a Merkle hash tree. In the construction, the leaf nodes of the Merkle hash tree are hashes of $H(m_i)$. (P) ← GenProof(F,φ, chal). This algorithm is run by the server. It takes as input a file F, its signatures , and a challenge chal. It outputs a data integrity proof P for the blocks specified by chal.{TRUE, FALSE} ← VerifyProof(pk, chal, P). This algorithm can be run by either the client or the third party auditor upon receipt of the proof P. It takes as input the public key pk, the challenge chal, and the proof P returned from the server, and outputs TRUE if the integrity of the file is verified as correct, or FALSE otherwise.(F',φ',P_{update})←ExecUpdate(F,φ, update). This algorithm is run by the server. It takes as input a file F, its signatures φ, and a data operation request "update" from

client. It outputs an updated file F′, updated signatures φ′ and a proof Pupdate for the operation.{(TRUE, FALSE, $sig_{sk}(H(R'))$} ←VerifyUpdate(pk, update, P_{update}). This algorithm is run by the client. It takes as input public key pk, the signature $sig_{sk}(H(R))$, an operation request "update", and the proof Pupdate from server. If the verification successes, it outputs a signature $sig_{sk}(H(R'))$ for the new root R′, or FALSE otherwise.

C The Construction

To effectively support public auditability without having to retrieve the data blocks themselves, it resort to the homomorphic authenticator technique. Homomorphic authenticators are unforgeable metadata generated from individual data blocks, which can be securely aggregated in such a way to assure a verifier that a linear combination of data blocks is correctly computed by verifying only the aggregated authenticator. In the design, to propose to use PKC based homomorphic authenticator (e.g., BLS signature or RSA signature based authenticator) to equip the verification protocol with public auditability. In the following description, it present the BLS-based scheme to illustrate the design with data dynamics support. The schemes designed under BLS construction can also be implemented in RSA construction and it believe that protocol design for supporting dynamic data operation is a major challenging task for cloud storage systems.

Setup: The client's public key and private key are generated by invoking KeyGen(·). By running SigGen(·), the data file F is pre-processed, and the homomorphic authenticators together with metadata are produced.

Default Integrity Verification: The client or TPA can verify the integrity of the outsourced data by challenging the server. Before challenging, the TPA first use spk to verify the signature on t. If the verification fails, reject by emitting FALSE; otherwise, recover u.

Dynamic Data Operation with Integrity Assurance: It show how the scheme can explicitly and efficiently handle fully dynamic data operations including data modification (M), data insertion (I) and data deletion (D) for cloud data storage. It assume that the file and the signature have already been generated and properly stored at server. The root metadata R has been signed by the client and stored at the cloud server, so that anyone who has the client's public key can challenge the correctness of data storage.

Batch Auditing for Multi-client Data**:** As cloud servers may concurrently handle multiple verification sessions from different clients, given K signatures on K distinct data files from K clients, it is more advantageous to aggregate all these signatures into a single short one and verify it at one time. To achieve the goal, to extend the scheme to allow for provable data updates and verification in a multi-client system. As in the BLS based construction, the aggregate signature scheme allows the creation of signatures on arbitrary distinct messages. Moreover, it supports the aggregation of multiple signatures by distinct signers on distinct messages into a single short signature, and thus greatly reduces the communication cost while providing efficient verification for the authenticity of all messages.

IV. CONCLUSION

To ensure cloud data storage security, it is critical to enable a third party auditor (TPA) to evaluate the service quality from an objective and independent perspective. Public auditability also allows clients to delegate the integrity verification tasks to TPA while they themselves can be unreliable or not be able to commit necessary computation resources performing continuous verifications. Another major concern is how to construct verification protocols that can accommodate dynamic data files. To explored the problem of providing simultaneous public auditability and data dynamics for remote data integrity check in Cloud Computing. The construction is deliberately designed to meet these two important goals while efficiency being kept closely in mind. To achieve efficient data dynamics, and improve the existing proof of storage models by manipulating the classic Merkle Hash Tree (MHT) construction for block tag authentication. To support efficient handling of multiple auditing tasks, to further explore the technique of bilinear aggregate signature to extend the main result into a multi-user setting, where TPA can perform multiple auditing tasks simultaneously. Extensive security and performance analysis show that the proposed scheme is highly efficient and provably secure.

REFERENCES

1. Ateniese G, Burns R, Curtmola R, Herring J, Kissner L, Peterson Z, and Song D, "Provable data possession at untrusted stores," in Proc. of CCS'07. New York, NY, USA: ACM, 2007, pp. 598–609.
2. Ateniese G, Pietro R.D, Mancini L.V, and Tsudik G, "Scalable and efficient provable data possession," in Proc. of SecureComm'08. New York, NY, USA: ACM, 2008, pp.1-10.
3. Bellare M and Rogaway P, "Random oracles are practical: A paradigm for designing efficient protocols," in Proc. of CCS'93, 1993, pp. 62–73.
4. Boneh D, Lynn B, and Shacham H, "Short signatures from the weil pairing," in Proc. of ASIACRYPT'01. London, UK: Springer- Verlag, 2001, pp. 514–532.
5. Boneh D, Gentry C, Lynn B, and Shacham H, "Aggregate and verifiably encrypted signatures from bilinear maps," in Proc. Of Eurocrypt'03. Warsaw, Poland: Springer-Verlag, 2003, pp. 416–432.
6. Bowers K.D, Juels A, and Oprea A, "Proofs of retrievability: Theory and implementation," Cryptology ePrint Archive, Report 2008/175, 2008.
7. Bowers K.D, Juels A, and Oprea A, "Hail: A high-availability and integrity layer for cloud storage," in Proc. of CCS'09. Chicago, IL, USA: ACM, 2009, pp. 187–198.
8. Chang E.C, and Xu J, "Remote integrity check with dishonest storage server," in Proc. of ESORICS'08. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 223–237.
9. Erway C, Kupcu A, Papamanthou C, and Tamassia R, "Dynamic provable data possession," in Proc. of CCS'09. Chicago, IL, USA: ACM, 2009.
10. Juels A and Kaliski B.S, Jr., "Pors: proofs of retrievability for large files," in Proc. of CCS'07. New York, NY, USA: ACM, 2007, pp. 584–597.
11. Lin S and Costello D.J, Error Control Coding, Second Edition. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 2004.
12. Merkle R.C, "Protocols for public key cryptosystems," Proc. Of IEEE Symposium on Security and Privacy'80, pp. 122–133, 1980.

13. Naor M and Rothblum G.N, "The complexity of online memory checking," in Proc. of FOCS'05, Pittsburgh, PA, USA, 2005, pp. 573–584.
14. Oprea A, Reiter M.K, and Yang K, "Space-efficient block storage integrity," in Proc. of NDSS'05, San Diego, CA, USA, 2005.
15. Schwarz T and Miller E.L, "Store, forget, and check: Using algebraic signatures to check remotely administered storage," in Proc. of ICDCS'06, Lisboa, Portugal, 2006, pp. 12–12.
16. Shacham H and Waters B, "Compact proofs of retrievability," in Proc. of ASIACRYPT'08. Melbourne, Australia: Springer-Verlag, 2008, pp. 90–107.
17. Shah M.A, Swaminathan R, and Baker M, "Privacy-preserving audit and extraction of digital contents," Cryptology ePrint Archive, Report 2008/186, 2008.
18. Wang Q, Wang C, Li J, Ren K, and Lou W, "Enabling public verifiability and data dynamics for storage security in cloud computing,"in Proc. of ESORICS'09. Saint Malo, France: Springer-Verlag, 2009, pp. 355–370.
19. Wang Q, Ren K, Lou W, and Zhang Y, "Dependable and secure sensor data storage with dynamic integrity assurance," in Proc. Of IEEE INFOCOM'09, Rio de Janeiro, Brazil, April 2009, pp. 954–962.
20. Wang C, Wang Q, Ren K, and Lou W, "Ensuring data storage security in cloud computing," in Proc. of IWQoS'09, Charleston, South Carolina, USA, 2009.