Applying Public Auditability for Securing Cloud Data from Modification Attack

Vinay Tila Patil [1] Prof. Gajendra Singh Chandel[2]
[1]M.Tech Student(Software Engineering) [2]H.O.D.
[1,2]Department of Computer Science & Engineering
[1,2]SSSIST, Sehore, (M.P.), India

Abstract— Cloud Computing has been proposed as the next-generation architecture of IT Enterprise. It moves the application software and databases to the integrated large data centers, where the management of the data and services may not be fully trustworthy. This unique paradigm brings about many new security challenges, which have not been well understood. This work studies the problem of ensuring the integrity of data storage in Cloud Computing. In particular, we 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, this paper achieves both. We 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 our protocol design. In particular, to achieve efficient data dynamics, we 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, we further explore the technique of bilinear aggregate signature to extend our 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.

Even though anticipated as a promising overhaul platform for the Internet, this new data storage model in "Cloud" brings about many thought-provoking design issues which have reflective influence on the sanctuary and performance of the overall system. One of the biggest concerns with cloud data storage is that of data integrity verification at untrusted servers. For example, the storage service provider, which experiences Byzantine failures occasionally, may decide to hide the data errors from the clients for the benefit of their own. 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.

In order to solve the problem of data integrity checking, many schemes are proposed under different systems and security models [2]–[11]. In all these works, great efforts are made to design solutions that meet various requirements: high scheme efficiency, stateless verification, unbounded use of queries and retrievability of data, etc. 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 this dynamic data updates has received limited attention so far [2]–[5], [7], [10], [12]. Moreover, as will be shown later, the direct extension of the current provable data possession (PDP) [2] or proof of retrievability (PoR) [3], [4] schemes to support data dynamics may lead to security loopholes. Although there are many difficulties faced by researchers, it is well believed that supporting dynamic data operation can be of

vital importance to the practical application of storage outsourcing services. In view of the key role of public auditability and data dynamics for cloud data storage, we propose an efficient construction for the seamless integration of these two components in the protocol design. Our contribution can be summarized as follows:

- We 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;
- We extend our scheme to support scalable and efficient public auditing in Cloud Computing. In general, our scheme realises batch auditing where numerous delegated auditing tasks from different users can be performed concurrently by the TPA.
- We prove the security of our proposed construction and justify the performance of our scheme through concrete implementation and comparisons with the state-of-the-art.

A. Related Work

Recently, much of growing interest has been pursued in the context of remotely stored data verification [2]– [10], [12]–[15]. Ateniese et al. [2] 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, they utilize RSAbased 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 [12], 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 [13], 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 [12], they only consider partial support for dynamic data operation. Juels et al. [3] 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 [12], 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. [4] design an improved PoR scheme with full proofs of security in the security model defined in [3]. They use publicly verifiable homomorphic authenticators built from BLS signatures [16], 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. [14] was the first to explore constructions for dynamic provable data possession. They extend the PDP model in [2] to support provable updates to stored data files using rank-based authenticated skip lists. This 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 [2] 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. Rations of the work offered in this critique have previously performed as an stretched abstract [1]. We revise the article a lot and add more technical details as compared to [1]. Firstly, in Section III-C, before the introduction of our proposed construction we present two basic solutions (i.e., the MAC-based and signature based schemes) for realizing data auditability and discuss their demerits in supporting public auditability and data dynamics. Secondly, we 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, we 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.

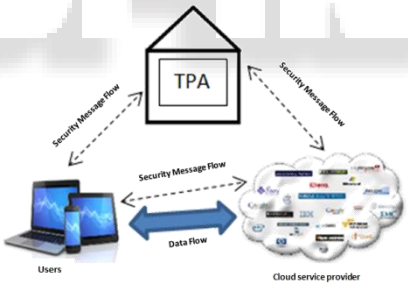

Fig. 1: Cloud Data Storage Architecture using TPA

II. PROBLEM STATEMENT

A. System Model:

Representative network architecture for cloud data storage is illustrated in Fig. 1. Three different network entities can be identified as follows:

1) 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;

2) 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;

3) 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.

In the cloud paradigm, by putting the large data files on the remote servers, the clients can be relieved of the burden of storage and computation. As clients no longer possess their data locally, it is of critical importance for the clients to ensure that their data are being correctly stored and maintained. That is, clients should be equipped with certain security means so that they can periodically verify the correctness of the remote data even without the existence of local copies. In case those clients do not necessarily have the time, feasibility or resources to monitor their data, they can delegate the monitoring task to a trusted TPA. In this paper, we only consider verification schemes with public auditability: any TPA in possession of the public key can act as a verifier. We assume that TPA is unbiased while the server is untrusted. For application purposes, the clients may interact with the cloud servers via CSP to access or retrieve their pre-stored data. More importantly, in practical scenarios, the client may frequently perform block-level operations on the data files. The most general forms of these operations we consider in this paper are modification, insertion, and deletion. Note that we don't address the issue of data privacy in this paper, as the topic of data privacy in Cloud Computing is orthogonal to the problem we study here.

B. Security Model:

Following the security model defined in [4], we say 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 intermittently challenge the storage server to guarantee the exactness of the cloud data, and the original files can be improved by co-operating with the server. The authors in [4] 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 challenger and the consumer, the adversary has full right of entry to the information deposited 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.

Our security model has subtle but crucial difference from that of the existing PDP or PoR models [2]–[4] in the verification process. As mentioned above, these schemes do not consider dynamic data operations, and the block insertion cannot be supported at all. This is since the creation of the monograms is involved with the file catalogue 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 this limitation, we remove the index information i in the computation of signatures and use H (mi) as the tag for block mi instead of H(name||i) [4] or h(v||i) [3], so individual data operation on any file block will not affect the others. Recall that in existing PDP or PoR models [2], [4], H (name||i) or h (v||i) should be generated by the client in the verification process. However, in our new construction the client has no capability to calculate H (mi) without the data information. In order to achieve this blockless verification, the server should take over the job of computing H (mi) and then return it to the prover. The consequence of this variance will lead to a serious problem: it will give the adversary more opportunities to cheat the prover by manipulating H (mi) or mi. Due to this construction; our security model differs from that of the PDP or PoR models in both the verification and the data updating process. Specifically, the tags in our scheme should be authenticated in each protocol execution other than calculated or pre-stored by the verifier (The details will be shown in section III). In the following descriptions, we will use cloud server and prover (or client, TPA and verifier) interchangeably.

C. Design Goals:

Our design goals can be summarized as the following:
1) 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;
2) 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;
3) Blockless verification:
no challenged file blocks should be retrieved by the verifier (e.g., TPA) during verification process for efficiency concern.

III. THE PROPOSED SCHEME

In this section, we present our security protocols for cloud data storage service with the aforementioned research goals in mind. We start with some basic solutions aiming to provide integrity assurance of the cloud data and discuss their demerits. Then we present our protocol which supports public auditability and data dynamics. We also show how to extent our main scheme to support batch auditing for TPA upon delegations from multiusers.

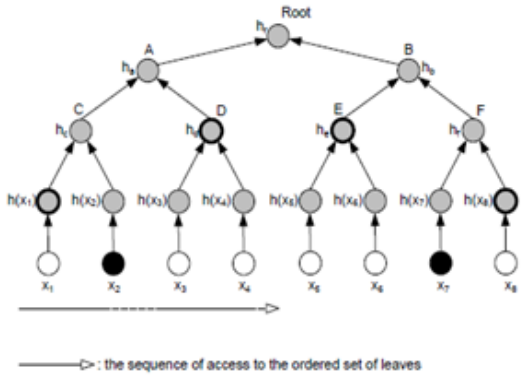

Fig. 2: MHT authentication of data elements

Merkle Hash: Tree. A Merkle Hash Tree (MHT) is a well-studied authentication structure [17], 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. Fig. 2 depicts an example of authentication. The verifier with the reliable hr requests for {x2, x7} and needs the verification of the established blocks.

MHT is commonly used to authenticate the values of data blocks. However, in this paper we further employ MHT to authenticate both the values and the positions of data blocks. We treat the leaf nodes as the left-to-right sequence, so any leaf node can be uniquely determined by following this sequence and the way of computing the root in MHT.

A. Basic Solutions:

Assume the outsourced data file F consists of a finite ordered set of blocks m1, m2. . . mn. One straightforward way to ensure the data integrity is to pre-compute MACs for the entire data file. Specifically, before data outsourcing, the data owner pre-computes MACs of F with a set of secret keys and stores them locally. During the auditing process, the data owner each time reveals a secret key to the cloud server and asks for a fresh keyed MAC for verification. This approach provides deterministic data integrity assurance straightforwardly as the verification covers all the data blocks. However, the number of verifications allowed to be performed in this solution is limited by the number of secret keys. Once the keys are exhausted, the data owner has to retrieve the entire file of F from the server in order to compute new MACs, which is usually impractical due to the huge communication overhead. Moreover, public auditability is not supported as the private keys are required for verification.

Another basic solution is to use signatures instead of MACs to obtain public auditability. The data owner precomputes the signature of each block mi ($i \in [1, n]$) and sends both F and the signatures to the cloud server for storage. To verify the correctness of F, the data owner can adopt a spot-checking approach, i.e., requesting a number of randomly selected blocks and their corresponding signatures to be returned. This basic solution can provide probabilistic assurance of the data correctness and support public auditability. However, it also severely suffers from the fact that a considerable number of original data blocks should be retrieved to ensure a reasonable detection probability, which again could result in a large communication overhead and greatly affects system efficiency. Notice that the above solutions can only support the case of static data, and none of them can deal with dynamic data updates.

B. Our Construction:

To effectively support public auditability without having to retrieve the data blocks themselves; we resort to the homomorphic authenticator technique [2], [4]. 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 our design, we propose to use PKC constructed homomorphic authenticator (e.g., BLS signature [4] or RSA signature centred authenticator [2]) to train the authentication protocol with open auditability. In the following description, we present the BLS-based scheme to illustrate our design with data dynamics support. As will be shown, the schemes designed under BLS construction can also be implemented in RSA construction. In the discussion of section III-D, we show that direct extensions of previous work [2], [4] have security problems. And we believe that protocol design for supporting dynamic data operation is a major challenging task for cloud storage systems.

1) 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.

2) Default Integrity Verification:

The client or TPA can verify the integrity of the outsourced data by challenging the server. Before challenging, the TPA first uses spk to verify the signature on t.

3) Dynamic Data Operation with Integrity Assurance:

Now we show how our 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. Note that in the following descriptions, we assume that the file F 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.

4) Data Modification:

We start from data modification, which is one of the most frequently used operations in cloud data storage. A basic data modification operation refers to the replacement of specified blocks with new ones.

5) Data Insertion:

Compared to data modification, which does not change the logic structure of client's data file, another general form of data operation, data insertion, refers to inserting new blocks after some specified positions in the data file F.

6) Data Deletion:

Data deletion is just the opposite operation of data insertion. For single block deletion, it refers to deleting the specified block and moving all the latter blocks one block forward.

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. In this paper, we explored the problem of providing simultaneous public auditability and data dynamics for remote data integrity check in Cloud Computing. Our construction is deliberately designed to meet these two important goals while efficiency being kept closely in mind. To achieve efficient data dynamics, we 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, we further explore the technique of bilinear aggregate signature to extend our 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] Q. Wang, C. Wang, J. Li, K. Ren, and W. Lou, "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.
[2] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, and D. Song, "Provable data possession at untrusted stores," in Proc. of CCS'07. New York, NY, USA: ACM, 2007, pp.598–609.
[3] Juels and B. S. Kaliski, Jr., "Pors: proofs of retrievability for large files," in Proc. of CCS'07. New York, NY, USA: ACM, 2007, pp.584–597.
[4] H. Shacham and B. Waters, "Compact proofs of retrievability," in Proc. of ASIACRYPT'08. Melbourne, Australia: Springer-Verlag, 2008, pp. 90–107.
[5] K. D. Bowers, A. Juels, and A. Oprea, "Proofs of retrievability: Theory and implementation," Cryptology ePrint Archive, Report 2008/175, 2008.
[6] M. Naor and G. N. Rothblum, "The complexity of online memory checking," in Proc. of FOCS'05, Pittsburgh, PA, USA, 2005, pp.573–584.
[7] E.-C. Chang and J. Xu, "Remote integrity check with dishonest storage server," in Proc. of ESORICS'08. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 223–237.
[8] M. A. Shah, R. Swaminathan, and M. Baker, "Privacy-preserving audit and extraction of digital contents," Cryptology ePrint Archive, Report 2008/186, 2008.
[9] Oprea, M. K. Reiter, and K. Yang, "Space-efficient block storage integrity," in Proc. of NDSS'05, San Diego, CA, USA, 2005.
[10] T. Schwarz and E. L. Miller, "Store, forget, and check: Using algebraic signatures to check remotely administered storage," in Proc. of ICDCS'06, Lisboa, Portugal, 2006, pp. 12–12.
[11] Q. Wang, K. Ren, W. Lou, and Y. Zhang, "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.
[12] G. Ateniese, R. D. Pietro, L. V. Mancini, and G. Tsudik, "Scalable and efficient provable data possession," in Proc. of SecureComm'08. New York, NY, USA: ACM, 2008, pp. 1–10.
[13] Wang, Q. Wang, K. Ren, and W. Lou, "Ensuring data storage security in cloud computing," in Proc. of IWQoS'09, Charleston, South Carolina, USA, 2009.
[14] Erway, A. Kupcu, C. Papamanthou, and R. Tamassia, "Dynamic provable data possession," in Proc. of CCS'09. Chicago, IL, USA: ACM, 2009.
[15] K. D. Bowers, A. Juels, and A. Oprea, "Hail: A high-availability and integrity layer for cloud storage," in Proc. of CCS'09. Chicago, IL, USA: ACM, 2009, pp. 187–198.
[16] Boneh, B. Lynn, and H. Shacham, "Short signatures from the weil pairing," in Proc. of ASIACRYPT'01. London, UK: Springer- Verlag, 2001, pp. 514–532.
[17] R. C. Merkle, "Protocols for public key cryptosystems," Proc. Of IEEE Symposium on Security and Privacy'80, pp. 122–133, 1980.
[18] S. Lin and D. J. Costello, Error Control Coding, Second Edition. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 2004.
[19] M. Bellare and P. Rogaway, "Random oracles are practical: A paradigm for designing efficient protocols," in Proc. of CCS'93, 1993, pp. 62–73.
[20] Boneh, C. Gentry, B. Lynn, and H. Shacham, "Aggregate and verifiably encrypted signatures from bilinear maps," in Proc. Of Eurocrypt'03. Warsaw, Poland: Springer-Verlag, 2003, pp. 416– 432.

