INTERNATIONAL JOURNAL OF RESEARCH IN COMPUTER APPLICATIONS AND ROBOTICS
ISSN 2320-7345

STRUCTURED EMINENT DYNAMIC AUDITING FOR SECURE DATA STORAGE IN CLOUD COMPUTING

Mr.Elamparithi P[1], Mr.Thanga Mariappan L[2], Mr.Ponnurajan P[3]

[1]parithics@gmail.com, [2]thangamariappanme@gmail.com, [3]rajanpsks@gmail.com

Assistant Professor Information Technology Department, Sree Sowdambika College of Engineering, Aruppukottai, Virudhunagar District – 626 101.

Abstract

Cloud computing stores the application software and databases to the centralized large data centers. The trustworthy management of data and services are improved by achieving efficient data dynamics. To verify the integrity of the dynamic data stored in the cloud TPA is used on behalf of the cloud client. The data operations such as block modification, insertion and deletion supports for data dynamics. In cloud computing services are not limited to achieve or backup data only. Initially identify the difficulties and problems of direct extensions with fully dynamic data updates from prior works

Keywords: Cloud Computing, Data Dynamics, Data Storage, Public Audit ability.

1. INTRODUCTION

Cloud computing, the trend toward loosely coupled networking of computing resources its unmooring data from local storage platforms. 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, this 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 .users today regularly access files without knowing or needing to know on what machines or in what geographical locations their files reside. 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 cap-ability, 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. 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, insertion, etc. 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 out-sourcing 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.

2. MOTIVATION

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 particular, our scheme achieves batch auditing where multiple delegated auditing tasks from different users can be performed simultaneously 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 existing security in that state of our work.

3. RELATED WORK

To consider public auditability in their defined "provable data possession" model for ensuring possession of files on untrusted storages [2]. In their scheme, they 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 [5], it only allows very basic block operations with limited functionality, and block insertions cannot be supported.

For dynamic data operation [3] describe a "proof of retrievability" 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 [5], the number of queries a client can perform is also a fixed priori, and the introduction of precomputed "sentinels" prevents the development of realizing dynamic data updates. In addition, public audit-ability is not supported in their scheme. BLS signatures [9], 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. [12] were 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.

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.

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. Second, we generalize the support of data dynamics to both PoR and PDP models and discuss the impact of dynamic data operations on the overall system efficiency both. Third, we extend our data auditing scheme for the single client and explicitly include a concrete description of the multiclient data auditing scheme. We also redo the whole experiments and present the performance comparison between the multi-client data auditing scheme and the individual auditing scheme. Finally, for the proposed theorems in this paper, we provide formal security proofs under the random oracle model, which are lacking in [1].

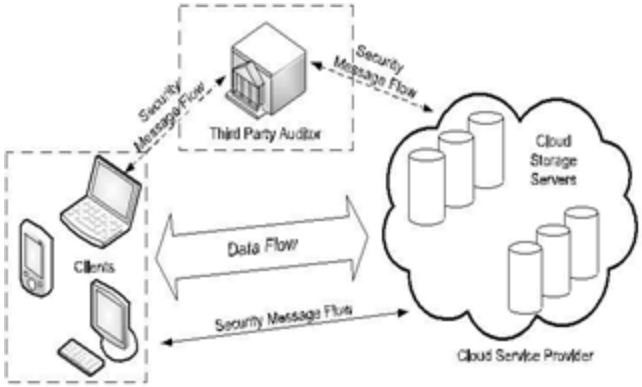


Fig. 1. Cloud data storage architecture

4. PROBLEM STATEMENT

4.1 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: 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.

4.2 Security Model

Following the security model defined in [4], we say that the checking scheme is secure if 1) there exists no polynomial- time algorithm that can cheat the verifier with non- negligible probability; and 2) there exists a polynomial-time extractor that can recover the original data files by carrying out multiple challenges-responses.

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.

4.3 Design Goals

Our 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.

5. PROPOSED SYSTEM

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.

5.1 Notation and Preliminaries

Bilinearmap. A bilinear map is a map e: G ×G →GT , where G is a Gap Diffie-Hellman (GDH) group and GT is another multiplicative cyclic group of prime order p with the following properties [9]: 1) Computable 2) Bilinear 3) Nondegenerate.

Merkle hash tree. A Merkle Hash Tree (MHT) is a well- studied authentication structure [10], 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.

5.2 Definition

(pk,sk)← KeyGen(1k). 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,sigsk(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 {mi}, and outputs the signature set _, which is an ordered collection of signatures

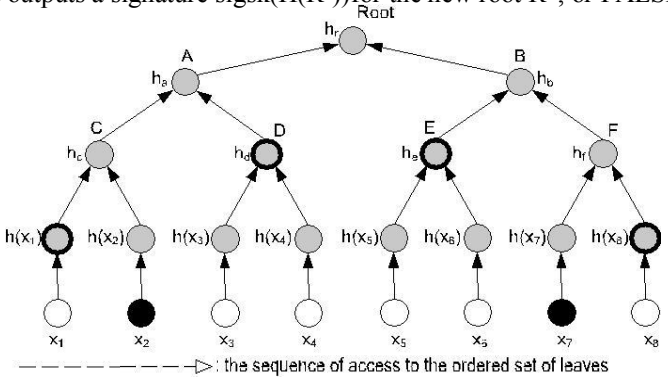
{σi} on {mi}. It also outputs metadata—the signature sigsk(H(R)) of the root R of a Merkle hash tree. In our construction, the leaf nodes of the Merkle hash tree are hashes of H(mi).

(P)←GenProof(F,Φ,chal). This algorithm is run by the server. It takes as input a file F, its signatures Φ, 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',Pupdate)←ExecUpdate(F, Φ, update). This algo-rithm 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,sigsk(H(R'))}←VerifyUpdate(pk,update,Pu pdate). This algorithm is run by the client. It takes as input public key pk, the signature sigsk(H(R)),an operation request "update," and the proof Pupdate from server. If the verification successes, it outputs a signature sigsk(H(R'))for the new root R", or FALSE otherwise.



6. CONSTRUCTION

Setup

In this phase KeyGen() method is invoked to generate public key and private key. SigGen() is meant for pre-processing and homomorphic authenticators and along with meta data. The SigGen() method takes two arguments namely secret key and file. The file content is divided into blocks. Then signature is computed for each block. Each block's hash code is taken and two nodes' hash is merged into one in order to generate the next node. This process continues for all leaf nodes until tree node is found. The root element is then taken by client and signs it and send to cloud storage server.

Data Integrity Verification

The content of outsourced data can be verified by either client or TPA. This is done by challenging server by giving some file and block randomly Up on the challenge, the cloud storage server computes the root hash code for the given file and blocks and then returns the computed root hash code and originally stored hash code along with signature. Then the TPA or client uses public key and private key in order to decrypt the content and compare the root hash code with the root hash code returned by client. This procedure is specified in the following algorithm.

Algorithm for data integrity verification

Step 1: Start

Step 2: TPA generates a random set

Step 3: CSS computes root hash code based on the filenames/blocks input.

Step 4: CSS computes the originally stored value

Step 5: TPA decrypts the given content and compares with generated root hash.

Step 6: After verification, the TPA can determine whether the integrity is breached

Step 7: Stop

Data Modification and Data Insertion

Data modifications are the frequent operations on cloud storage. It is a process of replacing specified blocks with new ones. The data modification operation can't affect the logic structure of client's data. Another operation is known as data insertion. Data Insertion is a process of inserting new record in to existing data. The new blocks are inserted into specified locations or blocks in the data file F.

Algorithm for updating and deleting data present in CSS

Step 1: Start

Step 2: Client generates new Hash for tree then sends it to CSS

Step 3: CSS updates F and computes new R'

Step 4: Client computes R

Step 5: Client verifies signature. If it fails output is FALSE

Step 6: Compute new R and verify the update and delete.

Batch Auditing for Multi-client Data

Cloud servers support simultaneous access. It does mean that in server it is possible to have different verification sessions running parallel. Therefore it is essential to have auditing functionality that works concurrently for many user sessions. The proposed scheme is extended to achieve this for provable data updates and verification of multi-client system. Here an important decision made is to make use of "Bilearaggregate Signature Scheme" [8].

7. SECURITY ANALYSIS

The proposed system enables public auditability without need for retrieving data blocks of a file. Towards this "homomorphic authenticator technique [1] [3] is used. There is the un-forgeable metadata generator computed from individual data blocks. In the proposed work two authenticators such as BLS signature [3] and RSA signature based authenticator. The security mechanism is further described here. The procedure of protocol is divided into setup, default integration verification and dynamic data operation with integrity assurance. In the last step, data modification, data insertion, and data deletion are a part. Later on batch processing with multi-client data is also discussed here.

8. RESULTS

First, we have to run the cloudserver which contains the files and data about client. Then start the TPA server which performs auditing and verification process. After that start the clientserver which approaches the cloud information. Now the TPA server audits the client authorized or not. After the auditability performed by TPA server, client can access the cloud information using their key. The key is used to extract the information which is separated in a block using merkle hash tree. Client can extract their required data in a secured manner.

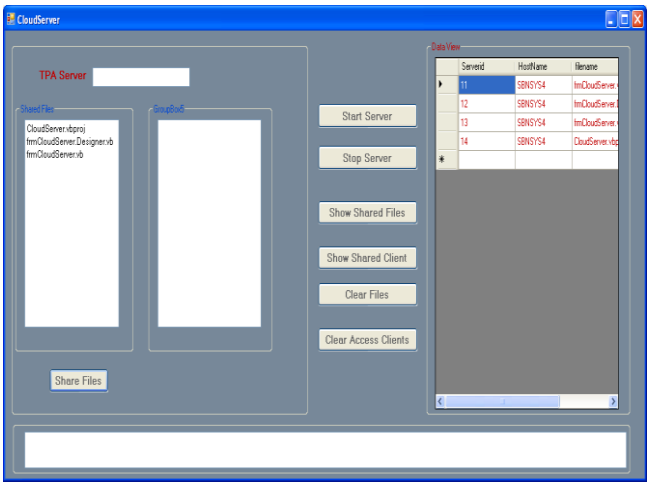


Fig 1.CLOUD SERVER

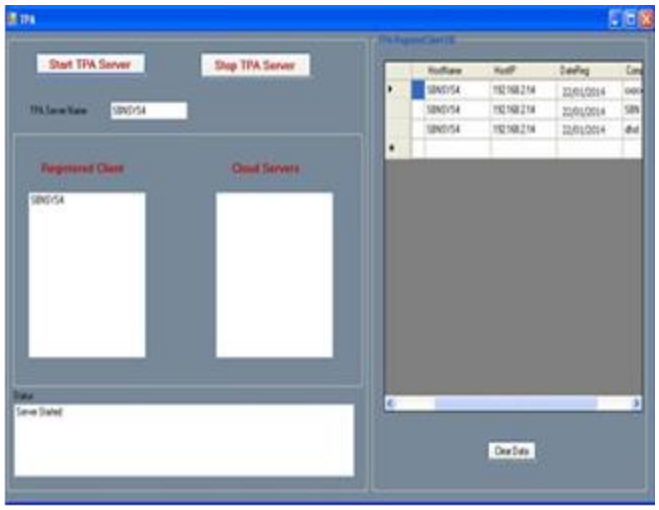


Fig 2.TPA SERVER

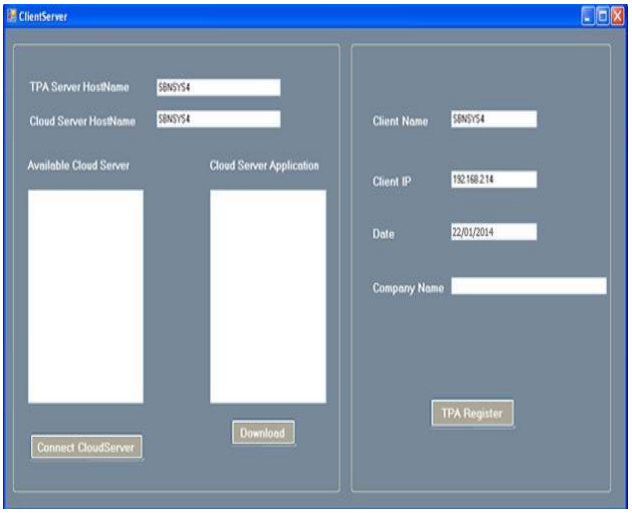


Fig 3.CLIENT SERVER

9. CONCLUSION

To ensure cloud data storage security, it is critical to enable a 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 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 multiuser 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," Proc. 14th European Symp. Research in Computer Security (ESORICS '09), pp. 355-370, 2009.

[2] 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 (CCS07), pp. 598-609, 2007.

[3] A. Juels and B.S. Kaliski Jr., "Pors: Proofs of Retrievability for Large Files," Proc. 14th ACM Conf. Computer and Comm. Security (CCS07), pp. 584-597,2007.

[4] H. Shacham and B. Waters, "Compact Proofs of Retrievability," Proc. 14th Int' Conf. Theory and Application of Cryptology and Information Security: Advances in Cryptology (ASIACRYPT'08), pp. 90-107, 2008.

[5] 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), pp. 1 -10, 2008.

[6] C. Wang, Q. Wang, K. Ren, and W. Lou, "Ensuring Data Storage Security in Cloud Computing," Proc. 17th Int‟l Workshop Quality of Service (IWQoS '09), 2009.

[7] C. Erway, A. Kupcu, C. Papamanthou, and R. Tamassia, "Dynamic Provable Data Possession," Proc. 16th ACM Conf. Computer and Comm. Security (CCS'09), 2009.

[8] K.D. Bowers, A. Juels, and A. Oprea, "Hail: A High-Availability and Integrity Layer for Cloud Storage," Proc. 16th ACM Conf. Computer and Comm. Security (CCS'09), pp. 187-198, 2009.

[9] 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 (ASIA-CRYPT '01), pp. 514-532, 2001.

[10] R.C. Merkle, "Protocols for Public Key Cryptosystems," Proc. IEEE Symp. Security and Privacy, pp. 122-133, 1980.

[11] H. Shacham and B. Waters, "Compact Proofs of Retrievability,"Proc. 14th Int' Conf. Theory and Application of Cryptology (ASIACRYPT'08), 90-107, 2008.

[12] C. Erway, A. Kupcu, C. Papamanthou, and R. Tamassia,"Dynamic Provable Data Possession," Proc. 16th ACM Conf.Computer and Comm. Security (CCS'09), 2009.

[13] M. Naor and G. N. Rothblum, "The complexity of online memory checking," in Proc. of FOCS'05, Pittsburgh, PA, USA, 2005,pp. 573–584.