Ensured Data Packet Security and Integrity using Merkle Hash Tree Algorithm in Cloud Computing System

SRIMANTA SARKAR

DEPARTMENT OF C.SE.

BHARATH UNIVERSITY

CHENNAI-600073

Email:sarkarsrimanta86@gmail.com

SYED IKRAM AHMED

DEPARTMENT OF C.SE

BHARATH UINERSITY

CHENNAI-600073

Email:ikram.catchme@gmail.com

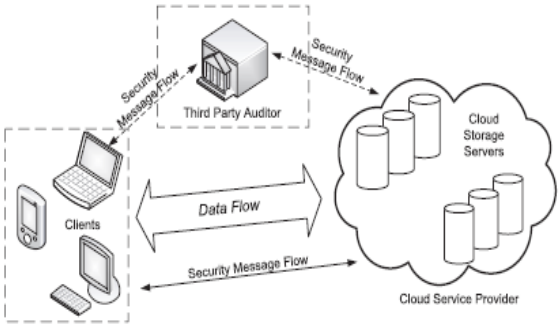
ABSTRACT

Cloud computing is considered to be the next big thing in Information Technology. Application soft wares and Databases can be stored in a remote site known as cloud data centers in which security is a concern from client point of view. The new method which is used to store and manage data without capital investment has brought many security challenges which are new to the IT world. This journal focuses on the security and integrity issues of data stored in cloud servers. Client hires a Third party auditor to check the correctness of the data periodically. The client of the data gets notifications from third party auditor when the data is corrupt or incorrect. Not only verification of data this system also supports data dynamics. The work that has been done in this field till now did not take the issue of data dynamics and public audtiiblity. The third party auditor randomly monitors data modifications, insertions and deletions. The proposed scheme is capable of supporting both public auditability and data dynamics. The review of literature shows that the data to be checked should be divide into small blocks, then each block should be checked for its integrity. Merkle Hash Tree is used to improve block by block verification Bilinear aggregate signature is used to enable TPA to perform auditing concurrently for multiple clients. The proposed system is very efficient and also secure.

1. INTRODUCTION

Cloud computing has emerged as the latest trend in IT, it enables software companies to store large databases in remote places and fetch the required data when required. It also enables IT companies to use computational power of the cloud. Cloud computing is a novel paradigm where organizations or individuals use the resources offered by cloud on rental basis and cut the cost of initial investment, and rather invest it on buying powerful processors and network elements. The clients, store important databases and data in the cloud server without having a local copy with them, as Storing data into the cloud relives the burden for storage. The client need not to have any private storage facility in this system. But while

taking pleasure of numerous benefits of this breakthrough technology there are many concerns as well, the organizations using a cloud for storage can never be sure of the integrity of the data stored. Organizations can have confidential and sensitive data which may not be completely safe as the cloud is a second party storage facility. Thus third party auditor is used by the client companies to check the correctness of the data stored in the cloud, they use various remote integrity checking protocols .Third party auditor is a reliable independent component which is trusted by both the cloud users and cloud server and has no motive to conspire with either the cloud server or user during the auditing process. Trusted third party auditor is claimed to have the more skill in the filed of data verification than a cloud user may have. In order to save time and reduce communication overhead, many organizations use the support of third party auditor (TPA). By leaving the resource consuming cryptographic operations on TPA for achieving confidentiality and integrity, cloud users can concentrate on the day to day work of the organization. The issues such as TPA becoming bottleneck, data leakage, introduction of new vulnerabilities like scalability, accountability and performance overheads, dynamic data support, extra hardware cost incurred etc. have motivated many researchers to address the data storage security problems without using a third party auditor, in this system the cloud user may be given the burden of extra application or tools which helps it periodically checks the data of that organization stored in the cloud. The proposed scheme achieves remote data storage integrity in cloud computing without making use of a third party auditor. The main design objective of the proposed scheme is to enhance user's control in the various aspects related to the secrecy of sensitive data. This can be achieved by implementing data privacy protocols to provide security of data in different categories. The proposed model is user-centric, that is cloud customer can flexibly control and manage the different privacy mechanisms necessary to protect sensitive data and achieve legal compliance. This model is good for organizations such as banks which store large amount of confidential data; this also helps them win customers trust and belief.



2. RELATED WORK

Recently, growing interest has been shown in the context of remotely stored data verification. Ateniese et al are the first to consider public auditability in their defined model for ensuring possession and control of files on un trusted remote storages. In their scheme, they utilize RSA- based algorithms for checking outsourced data, thus public auditability is reached. However, Ateniese et al. did not consider the circumstances of dynamic data storage, and they directly extended their scheme from static data storage to dynamic data. This may suffer design analysis and security issues. In their subsequent work, Ateniese et al. proposed a dynamic version of their prior PDP scheme. However this system imposes a priori bound on the number of queries and does not fully support dynamic data operations, i.e., it only allows basic block operations with limited

functionality, and block insertions are not supported. Wang et al. considered dynamic data storage in cloud scenario, and then proposed challenge- response protocol which can determine both the data correctness and locate possible errors. They also consider partial support for dynamic data operations. Juels and Kaliski described a "proof of retrievability" model, in this scheme spot-checking and error-correcting algorithms are used to ensure both "possession" and "retrievability" of data files on archive service systems. Precisely, some special blocks called "sentinels" are randomly embedded into the data file F for the purpose of detecting incorrectness, and F is further encrypted to protect the positions of these special blocks. However, like the previous model the number of queries a client can perform is also a fixed priori, and the introduction of predefined "sentinels" prevents the realization of dynamic data updates. In addition to this public auditability is not supported in their scheme. Shacham and Waters [4] designed an improved PoR scheme with full-proof security in the security model defined. They use publicly verifiable homomorphic authenticators built from *Boneh–Lynn–Schann s*ignatures, based on which the proofs can be aggregated into a small authenticator value, thus public retrievability is achieved. Still, the researchers only consider static data files. Erway et al. was the first to explore constructions for dynamic provable data possession. They extend the PDP model to support provable updates of stored data files using authenticated rank-based skip lists. This scheme is essentially a fully dynamic version of the PDP model. To support updates, like block insertion, they delete the index information in the "tag" computation in Ateniese's PDP model and employ authenticated rank-based skip list data structure to authenticate the tag information of updated blocks before the verification procedure. However, the performance efficiency of their scheme remains unclear. Although the discussed schemes aim at providing data integrity verification for different data storage systems, the challenge of supporting both public auditability and data dynamics simultaneously has not been fully addressed.

3. PROBLEM

3.1. System Model

Three different 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 storage and maintenance they can be either individuals or organizations; Cloud Storage Server (CSS): an entity, managed by Cloud Service Provider (CSP), has significant storage space and computational resource to maintain the client's data; Third Party Auditor: an entity, who have expertise and capabilities that clients do not have in the field of remote data auditing, it is trusted to expose and correct errors of cloud storage services on behalf of the clients on request. In cloud paradigm, by putting the large data files on the remote cloud servers, the clients can be relieved of the burden of own storage facilities and computation. As clients no longer possess local copy of their data , thus it is of critical importance for the clients to ensure that their data are being correctly stored and secure. That is, clients should be equipped with certain auditing means so that they can periodically check the correctness of the data stored remotely even without the existence of local copies. In case those clients do not have

the time or resources to monitor their data stored in the cloud they can delegate the monitoring task to a trusted TPA. In this paper, we only consider the verification schemes with public auditability: any TPA in possession of the public key can act as a auditor. We assume that Third party Auditor is unbiased while the server is un trusted. For application necessities, the clients may interact with the cloud servers to access or retrieve or modify their already stored data. More importantly, in practical cases, the client may randomly and frequently perform block-level verifications on the remotely stored data files. The most general forms of these operations we consider in this paper are basic functions like inserting, deleting or modifying

3.2Security Model

Following the security model , we say that the auditing scheme is secure if 1) there exists no other algorithm that can cheat the verifier with non negligible probability; and 2) there exists a poly nomialtime extractor that can recover the original files by extensive work. The client or TPA can periodically request the storage server to check the correctness of the data in the cloud, and the original files can be recovered by interacting with the cloud server. Note that in the conceptual "game" between the adversary and the client, the conceptual adversary has full access to the information stored in the server. The goal of the adversary is to successfully cheat the verifier, trying to generate valid responses and pass the authentication verification without being detected. Our security model has subtle some differences from that of the already existing PDP or PoR models of verification process. As mentioned before, these schemes do not support operations on dynamic data and the block insertions cannot be supported at all in this system. This is because the construction of the signatures is linked with the file index information i. Therefore, once a block of the file is inserted, the computation overhead is unacceptably high since the signatures of all the following file blocks should be recomputed with the new indexes. To deal with this drawback, 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) or h(v||i), so individual data operation on any file block will not affect the others. Recall that in existing s models where H(name||i) or h(v||i) should be generated by the client in the verification process. However, in our new scheme of things the client has no capability to calculate H(mi) without the data information. In an attempt to achieve this blockless verification, the server must take over the job of computing H(mi) and then return it to the cloud archive. The consequence of this variance will lead to a serious problem: it will give the adversary more opportunities to cheat the cloud archive by manipulating H(mi) or mi. Due to this new 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 new scheme should be authenticated during each protocol step execution other than calculated or prestored by the verifier. In the following descriptions, we will use server and the cloud interchangeably.

3.3. Design Goals

Our design goals can be summarized as the following: Public auditability for storage correctness assurance: to allow anybody, not just the clients who originally stored the file on cloud servers, to have the ability to verify the

correctness of the stored data on demand. Next is 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 of the model should be as efficient as possible so as to ensure the seamless integration of public auditability and dynamic data operation support. Blockless verification: no corrupt file blocks should be retrieved by the verifier (e.g., TPA) during verification process for efficiency concern.

4. DATA INTEGRITY MODEL

In this section, we present our security protocols for cloud data storage service with the already mentioned research aims in mind. Starting with some basic solutions aimed to provide integrity assurance of the cloud data and discuss their disadvantages. Thereafter, we present our protocol which supports public auditability and data dynamics. We also showed how our main scheme can be extended which supports batch auditing for TPA upon delegations from multi users. Notation and Preliminaries Bilinear map. 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: 1) Computable: there exists an efficiently computable algorithm for computing e; 2) Bilinear: for all h1, h2 G and a, b ZZp, e(ah 1 ,ah 2) = e(h1, h2)ab; 3) Non degenerate: e(g, g) 1, where g is a generator of G. Merkle hash tree. A Merkle Hash Tree (MHT) is a well studied authentication structure, which is intended to prove that a set of elements are undamaged and unaltered. It is constructed like a binary tree where the leaves in the MHT are the hashes of authentic data values. The verifier with the authentic hr requests for {x2,x7} and requires the authentication of the received blocks. The cloud archive provides the verifier with the auxiliary authentication information (AAI) 2 =< h(x1), hd > and 7 = <h(x8), he>. The verifier can then verify x2 and x7 by first computing h(x2), h(x7), hc = h(h(x1)||(x2))), hf = h(h(x7)||(x8))); ha = h(hc||d), hb = h(he||f) and hr = h(ha||hb), and then checking if the calculated hr is the same as the authentic one. Merkle Hash Tree is commonly used to authenticate the values of data blocks. However, in this journal we further employ MHT to authenticate both the values and the positions of data blocks. Leaf nodes are treated as the left-to- right sequence, so that any leaf node can be uniquely determined by following this sequence and the way of computing the root in MHT.

4.1 Basic Solutions

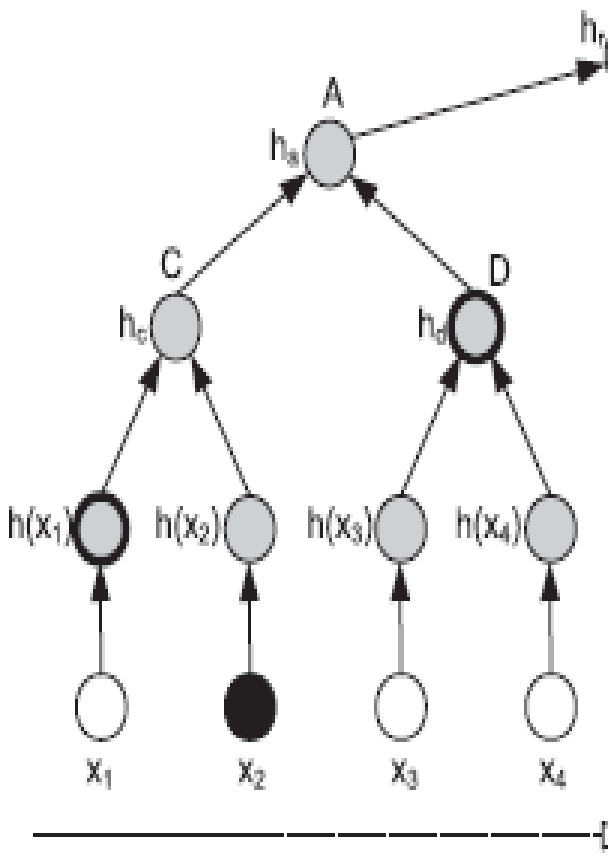
Assume the outsourced data file F consists of a finite ordered set of blocks m1,m2 m3,… . . .,mn. One straightforward way to ensure the data integrity is to pre compute MACs for the entire data file, before data outsourcing, the data owner pre computes MACs of F with a set of secret keys and stores them locally. At the time of auditing, the data owner each time reveals a secret key to the cloud server and asks for a fresh keyed MAC for verification. The above approach provides deterministic data integrity assurance straightforwardly as the verification covers all the data blocks. However, the number of verification checks allowed to be performed in this solution is limited by the number of secret keys. Once all the keys are finished, the data owner has to retrieve the entire file of F from the server in order to compute new MACs, which is by large impractical due to the huge communication overhead. In addition public auditability is also

not supported as the private keys are required for verification. Another simple solution is to use signatures instead of MACs to obtain public auditability. The data owner have to pre computes the signature of each block mi (i [1, n]) and then send both F and the signatures to the cloud server for storage. To check the correctness of F, the data owner can take up a spot-checking approach, i.e., asking for a number of randomly selected blocks and their corresponding signatures to be returned. This solution can provide probabilistic assurance of the data correctness and support public auditability. However, it also badly suffers from the fact that a considerable number of original data blocks should be retrieved to ensure a reasonable detection probability, which again can result in a large communication overhead and greatly affects system efficiency. Note that the above solutions can only support the case of static data, and they can deal with dynamic data updates.

4.2.Construction Scheme

To effectively support public auditability without having to retrieve the data blocks themselves, we go back to the homomorphic authenticator technique. Homomorphic authenticators cannot be forged Metadata generated from individual data blocks, which can be safely 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 scheme, we propose to use PKC-based homomorphic authenticator to equip the verification protocol with public auditability. Through following description, we will present the BLS-based scheme to illustrate our design with data dynamics support. As will be proved, the schemes designed under BLS construction can also be implemented in RSA construction. We show that direct extensions of previous work have security problems, and we believe that the design of protocol for supporting dynamic data operation is a major challenging task for cloud storage systems. Now we will present the main idea behind our scheme. We assume that file F (potentially encoded using Reed-Solomon codes) is divided into n blocks m1,m2, . . .,1 nm where mi ZZp and p is a large prime. Let e : G G GT be a bilinear map, with a hash function H : {0, 1}* G, viewed as a random oracle. Let g be the generator of G. let h be a cryptographic hash function. The main procedure of our protocol execution is as follows:

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. Notice that in the following descriptions, we assume the file F and the signature have already been generated and properly stored at the server. The root metadata R has been signed by the client and stored at the cloud server, so that anybody who has the client's public key can challenge the correctness of data storage. Data Modification start from data modification, which is the most frequently used operations in cloud data storage. A simple or basic data modification operation refers to the replacement of specified blocks with new ones.



4.4 Discussion on Design

Considerations Instantiations based on BLS and RSA algorithms. As discussed above, we present a BLS-based construction that offers both public auditability and data dynamics. In fact, our proposed model can also be constructed based on RSA signatures. Compared with RSA construction, as a desirable benefit, the BLS construction can offer shorter homomorphic signatures than those that use RSA techniques. In addition, the BLS construction has the shortest query and response: 20 and 40 bytes. However, while BLS construction is not suitable to use variable sized blocks, the RSA construction can support variable sized blocks, Because in RSA construction the order of QRN is unknown to the server, thus it is impossible to find distinct m1 and m2 such that gm1 mod N = gm2 mod N according to the factoring assumption. But block size cannot increase without limit, as the verification block = sc si 1 vimi grows linearly with the block size. Recall that h(H(mi)) are used as MHT leafs, upon receiving the challenge the server can calculate these tags on-the-fly or pre store them for fast proof computation. In fact, one can directly use h(gmi) as the MHT leaves instead of h(H(mi)). Like this, at the verifier side the job of computing the aggregated signature should be accomplished after authentication of gmi. Now the computation of aggregated signature is eliminated at the server side, as a trade-off, additional computation overhead may be introduced at the verifier side.

Our design goals are as follows:

1. Public auditability for storage correctness assurance: to allow anybody, not just the clients who originally stored the file on cloud servers, to have the ability 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 of the model should be as efficient as possible so as to ensure the seamless integration of public auditability and dynamic data operation support.

3. Block less verification: no challenged file blocks should be retrieved by the verifier (e.g., TPA) during verification process for efficiency concern.

5. IMPROVED STORAGE SECURITY MODEL

The system is designed to perform public data integrity analysis on cloud environment. Multi user based data file management is used in the system. Authentication and integrity schemes are improved in the system. Bilinear aggregate signature model is enhanced to manage multi user based data dynamic. The system includes confidentiality for data security. The system is designed to manage shared storage with security. Data providers share the storage space under the data centers. Third party auditor (TPA) verifies the data integrity for the data providers. The system is divided into five major modules. Data center, data provider, third party auditor, data distribution process and security management. The data center provides the shared storage for the data providers. The data provider module is designed to share data sources. Third party auditor handles the public auditability process. Data distribution module is designed to manage data update and delivery process. Integrity analysis is performed in security management module.

5.1. Data Center The data center application is designed to allocate storage space for the data providers. One or more data centers can be used in the cloud environment. Different sized storage area is allocated for the data providers. Data files are delivered to the clients.

5.2. Data Provider Shared data files are provided by the data providers. Data providers are managed by multiple users. Each user is assigned with separate authentication code. Data update operations are handled by the users.

5.3. Third party Auditor The third party auditor module is designed to provide security for the cloud. Data files and their signatures are maintained under the TPA. Block based signature model is used in the system. Data dynamics initiates the signature update process.

5.4. Data Distribution Process The data values are requested by the clients. The client requests are processed by the data centers. Shared data file contents are delivered to the clients. All the access information is updated to the data centers.

5.5. Security Management Authentication and integrity analysis are carried out under the security management process. Block signature model is used in the integrity analysis. TPA performs the security process. Bilinear aggregate signature is used for multi user based parallel verification process. The Rivest Cipher (RC4) algorithm is used for security.

6. CONCLUSION

Client data sources are managed under the cloud resources. Third Party Auditor (TPA) performs the public data verification process for the clients. The TPA model is enhanced to manage data sources for multi user environment. Data dynamic is enhanced with multi user based file operations. The system support multi user data verification process. Client resource consumption is reduced by the system. Data dynamic is supported for multi user environment. Simultaneous data verification is performed.

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 (CCS '07), 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 (CCS '07), pp. 584-597, 2007.

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

[5] K.D. Bowers, A. Juels, and A. Oprea, "Proofs of Retrievability: Theory and Implementation," Report 2008/175, Cryptology ePrint Archive, 2008.

[6] M. Naor and G.N. Rothblum, "The Complexity of Online Memory Checking," Proc. 46th Ann. IEEE Symp.Foundations of Computer Science (FOCS '05), pp. 573-584,2005