© 2015 IJSRSET | Volume 1 | Issue 2 | Print ISSN : 2395-1990 | Online ISSN : 2394-4099
Themed Section: Engineering and Technology

Fine Grained Updates in Cloud Using Third Party Auditing

M Paventhan, C Murugavel, S Rajadurai
Dhanalakshmi College of Engineering, Chennai, Tamilnadu, India

ABSTRACT

Users of cloud storage services no longer physically maintain direct control over their data, which makes data security one of the major concerns of using cloud. Existing research work already allows data integrity to be verified without possession of the actual data file. When the verification is done by a trusted third party, this verification process is also called data auditing and this third party is called an auditor. We call coarse-grained updates. As a result, every small update will cause re-computation and updating of the authenticator for an entire file block, which in turn causes higher storage and communication overheads. We provide a formal analysis for possible types of fine-grained data updates and propose a scheme that can fully support authorized auditing and fine-grained update requests. Based on our scheme, we also propose an enhancement that can dramatically reduce communication overheads for verifying small updates. Theoretical analysis and experimental results demonstrate that our scheme can offer not only enhanced security and flexibility, but also significantly lower overhead for big data applications with a large number of frequent small updates.

Keywords: Cloud Computing, Data Security, Provable Data Possession Authorized Auditing, Fine-Grained Dynamic Data Update

I. INTRODUCTION

Cloud computing is being intensively referred as one of the most influential innovations in information technology in recent years. With resource virtualization, cloud can deliver computing resources and services in a pay-as-you-go mode, which is envisioned to become as convenient to use similar to daily-life utilities such as electricity, gas, and water and telephone in the near future .These services, can be categorized into Infrastructure-as-a-Service, Platform-as-a-Service and Software-as-a-Service. Many international IT corporations now offer powerful public cloud services to users on a scale from individual to enterprise all over the world; for examples are Amazon AWS, Microsoft Azure, and IBM Smart Cloud.

Although current development and proliferation of cloud computing is rapid, debates and hesitations on the usage of cloud still exist. Data security/privacy is one of the major concerns in the adoption of cloud computing. Compared to conventional systems, users will lose their direct control over their data. In this paper, we will Investigate the problem of integrity verification for storage in cloud. This problem can also be called data auditing. When the verification is conducted by a trusted third party (TPA). From cloud user's perspective, it may also be called 'auditing-as-a-service'. To date, extensive research is carried out to address this problem. In a remote verification scheme, the cloud storage server cannot provide a valid integrity proof of a given proportion of data to a verifier unless all this data is intact. To ensure integrity of user data stored on cloud service provider, this support is of no less importance than any data protection mechanism deployed by the cloud service provider (CSP), no matter how secure they seem to be, in that it will provide the verifier a piece of direct, trustworthy and real-timed intelligence of the integrity of the cloud user's data through a challenge request. It is especially recommended that data auditing is to be conducted on a regular basis for the users who have high-level security demands over their data .Although existing data auditing schemes already have various properties, potential risks and inefficiency such as security risks in unauthorized auditing requests and inefficiency in processing small updates still exist. In this paper, we will focus on better support for small dynamic updates, which benefits the scalability and efficiency of a cloud storage server. To achieve this, our scheme utilizes a flexible data segmentation strategy and

a ranked Merkle hash tree (RMHT). Meanwhile, we will address a potential security problem in supporting public verifiability to make the scheme more secure and robust, which is achieved by adding an additional authorization process among the three participating parties of client, CSS and a third-party auditor (TPA).

II. METHODS AND MATERIAL

A. Related Work

In traditional systems, scalability and elasticity are key advantages of cloud. As such, efficiency in supporting dynamic data is of great importance. Security and privacy protection on dynamic data has been studied extensively in the past. In this paper, we will focus on small and frequent data updates, which is important because these updates exist in many cloud applications such as business transactions and online social networks (e.g. twitter). Cloud users may also need to split big datasets into smaller datasets and store them in different physical servers for reliability, privacy-preserving or efficient processing purposes .Among the most pressing problems related to cloud is data security/privacy. It has been one of the most frequently raised concerns. There is a lot of work trying to enhance cloud data security/privacy with technological approaches on CSP side. As they are of equal importance as our focus of external verifications .Integrity verification for outsourced data storage has attracted extensive research interest. The concept of proofs of retrieve ability (POR) and its first model was proposed. Unfortunately, their scheme can only be applied to static data storage such as archive or library. In the same year, Ateniese, et al. proposed a similar model named 'provable data possession' (PDP). Their schemes offer 'block less verification' which means the verifier can verify the integrity of a proportion of the outsourced file through verifying a combination of pre-computed file tags which they call homo morphic verifiable tags (HVTs) or holomorphic linear authenticators (HLAs). Work by Shacham, et al provided an improved POR model with stateless verification. They also proposed a MAC-based private verification scheme and the first public verification scheme in the literature that based on BLS signature scheme. In their second scheme, the generation and verification of integrity proofs are similar to signing and verification of BLS signatures. When wielding the same security strength (say, 80-bit security), a BLS signature (160 bit) is much shorter than an RSA signature (1024 bit), which is a desired benefit for a POR scheme. They also proved the security of both their schemes and the PDP scheme by Ateniese. From then on, the concepts of PDP and POR were in fact unified under this new compact POR model. Ateniese, extended their scheme for enhanced scalability, but only partial data dynamics and a predefined number of Challenges are supported. In 2009, Erway, et al. proposed the first PDP scheme based on skip list that can support full dynamic data updates [12]. However, public auditability and variable-sized file blocks are not supported by default .Wang, proposed a scheme based on BLS signature that can support public auditing (especially from a third party auditor, TPA) and full data dynamics, which is one of the latest works on public data auditing with dynamics support. However, their scheme lacks support for fine grained update and authorized auditing which are the main focuses of our work. Latest work by Wang. Added a random masking technology on top of to ensure the TPA cannot infer the raw data file from a series of integrity proofs. In their scheme, they also incorporated a strategy first proposed in to segment file blocks into multiple 'sectors'. However, the use of this strategy was limited to trading-off storage cost with communication cost. Other lines of research in this area include the work of Ateniese, on how to transform a mutual identification protocol to a PDP scheme; scheme by Zhu. That allows different service providers in a hybrid cloud to cooperatively prove data integrity to data owner; and the MR-PDP Scheme based on PDP proposed by Curtmola, that can efficiently prove the integrity of multiple replicas along with the original data file.

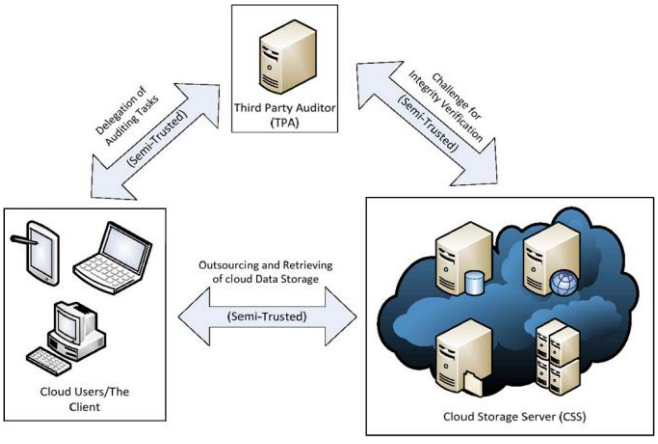


Figure 1: Relationship between the participating parties

B. Problem Statement

Many big data applications will keep user data stored on the cloud for small-sized but very frequent updates. A most typical example is Twitter, where each tweet is restricted to140 characters long (which equal 140 bytes in ASCII code).They can add up to a total of 12 terabytes of data per day. Storage of transaction records in banking or securities markets is a similar and more security-heavy example. Moreover, cloud users may need to split large-scale datasets into smaller chunks before uploading to the cloud for privacy-preserving [17] or efficient scheduling [19]. In this regard, efficiency in processing small updates is always essential in big data applications.

To better support scalability and elasticity of cloud computing, some recent public data auditing schemes do support data dynamics. However, types of updates supported are limited. Therefore previous schemes may not be suitable for some practical scenarios. Besides, there is a potential security threat in the existing schemes. The necessary authorization and authentication process between the auditor and cloud service provider in the existing system support dynamic data updates over fixed size data blocks, as a result every small update will cause re-computation and updating of the entire file block.

III. RESULTS AND DISCUSSION

A. Proposed Scheme

Proposed scheme can fully support authorized auditing and fine-grained update requests. Based on our scheme, we also propose an enhancement that can dramatically reduce communication overheads for verifying small updates Cloud storage fine grained data updates and to implement a scheme that fully support authorized auditing and fine grained update requests. In verification process the main adversary is the untrustworthy server did not carry out the data update successfully.

B. Roles of the Participating Parties

Most PDP and POR schemes can support public data verification. In such schemes, there are three participating parties: client, CSS and TPA. In brief, both CSS and TPA are only semi-trusted to the client. In the old model, the challenge message is very simple so that everyone can send a challenge to CSS for the proof of a certain set of file blocks, which can enable malicious exploits in practice. First, a malicious party can launch distributed denial-of-service attacks by sending multiple challenges from multiple clients at a time to cause additional overhead on CSS and congestion to its network connections, thereby causing degeneration of service qualities. Second, an adversary may get privacy-sensitive information from the integrity proofs returned by CSS. By challenging the CSS multiple times, an adversary can either get considerable information about user data or gather statistical information about cloud service status. To this end, traditional PDP models cannot quite meet the security requirements of 'auditing-as-a-service', even though they support public verifiability.

i. Verifiable Fine-Grained Data Operations

Some of the existing public auditing schemes can already support full data dynamics in their models, only insertions, deletions and modifications on fixed-sized blocks are discussed. Particularly, in BLS-signature-based schemes with 80-bit security, size of each data block is either restricted by the 160-bit prime group order p, as each block is segmented into a fixed number of 160-bit sectors. This design is inherently unsuitable to support variable-sized blocks, despite their remarkable advantage of shorter integrity proofs. In fact, as described existing schemes can only support insertion, deletion or modification of one or multiple fixed-sized blocks, which we call 'coarse-grained' updates.

ii. Ranked Merkle Hash Tree (RMHT)

The Merkle Hash Tree (MHT) has been studied in the past. In this paper we utilize an extended MHT with ranks which we named RMHT. Similar to a binary tree, each node N will have a maximum of 2 child nodes. In fact, according to the update algorithm, every non-leaf node will constantly have 2 child nodes. Information contained in one node N in an RMHT T is represented as fH; rNg where H is a hash value and rN is the rank of this node. T is constructed as follows. For a leaf node LN based on a message mi, we have H ¼ hðmiÞ, rLN ¼ si; A parent node of N1 ¼ fH1; rN1g and N2 ¼ fH2; rN2g is constructed as NP ¼ fhðH1kH2Þ; ðrN1 þ rN2Þg where k is a concatenation operator. A leaf node mi's AAI Wi is a set of hash values chosen from every of its upper level so that the root value R can be computed

through fmi;Wig. For example, for the RMHT , m1's AAI W1 ¼ fhðm2Þ; hðeÞ; hðdÞg. According to the property of RMHT, we know that the number of hash values included in Wi

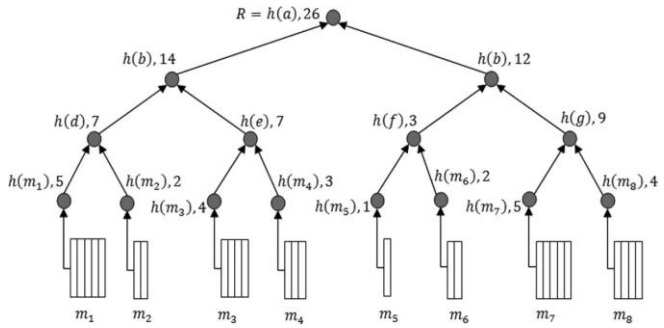


Figure 2: RHMT Tree

iii. Our Scheme

We now describe our proposed scheme in the aim of supporting variable-sized data blocks, authorized third party auditing and fine-grained dynamic data updates.

Our scheme is described in three parts:

1. Setup: the client will generate keying materials via KeyGen and FileProc, then upload the data to CSS. Different from previous schemes, the client will store a RMHT instead of a MHT as metadata. Moreover, the client will authorize the TPA by sharing a value sigAUTH.

2. Verifiable Data Updating: the CSS performs the client's fine-grained update requests via Perform Update, then the client runs Verify Update to check whether CSS has performed the updates on both the data blocks and their corresponding authenticators (usedfor auditing) honestly.

3. Challenge, Proof Generation and Verification: Describes how the integrity of the data stored on CSS is verified by TPA via GenChallenge, GenProof and Verify.

iv. Prepare for Authorization

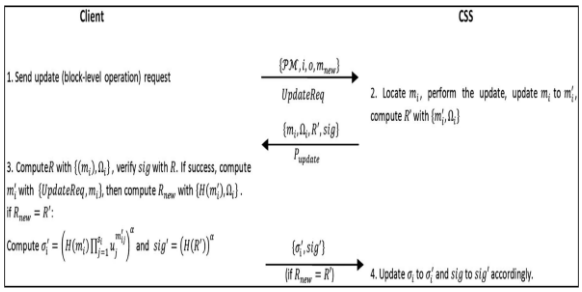
The client asks (her choice of) TPA for its ID VID (for security, VID is used for authorization only). TPA will then return its ID, encrypted with the client's public key. The client will then compute sigAUTH ¼ SigsskðAUTHktkVIDÞ and sends sigAUTH along with the auditing delegation request to TPA for it to compose a challenge later on. Different from existing schemes, after the execution of the above two algorithms, the client will keep the RMHT 'skeleton' with only ranks of each node and indices of each file block to reduce fine-grained update requests to block level operations. The client then sends fF; t;F; sig;AUTHg to CSS and deletes fF;F; t;F; sigg from its local storage. The CSS will construct an RMHT T based on mi and keep stored with fF; t;F; sig; AUTHg for later verification, which should be identical to the tree spawned at clientside

v. Verifiable Data Updating

Same as Setup, this process will also be between client and CSS. We discuss 5 types of block-level updates (operations) that will affect T: PM, M, D, J and SP (see Definition 1). We will discuss how these requests can form fine-grained update requests in general. The verifiable data update process for a PM-typed update is as follows :

1. The client composes an update quest UpdateReq defined in Section 4.2 and sends it to CSS.
2. CSS executes the following algorithm: PerformUpdateðUpdateReq;FÞ: CSS parses UpdateReq and get fPM; i;o;mnewg. When Type ¼ PM, CSS will update mi and T accordingly, then output Pupdate ¼ fmi;Wi;R0; sigg (note that Wi stays the same during the update) and the updated file F0. Upon finishing of this algorithm, CSS will send Pupdate to the client.
3. After receiving Pupdate, the client executes the following algorithm: VerifyUpdateðpk; PupdateÞ: The client computes m0 i using fmi; UpdateReqg, then parse Pupdate to fmi;Wi; R0; sigg, compute R (and HðRÞ) and Rnew use fmi;Wig and fm0i;Wig respectively. It verifies sig use HðRÞ, and check if Rnew ¼ R0. If either of these two verifications fails, then output FALSE and return to CSS, otherwise. Due to their similarity to the process described above, other types of operations are only briefly discussed as follows. For whole-block operations M, D, and J, as in model in the existing work , the client can directly compute _0 i without retrieving data from the original file F stored on CSS, thus the client can send _0 i along with UpdateReq in the first phase. For responding to an update request, CSS only needs to send back HðmiÞ instead of mi. Other operations will be similar to where Type ¼ PM. For an SP-typed update, in addition to updating mi to m i, a new block m_ needs to be inserted to T after m0i.

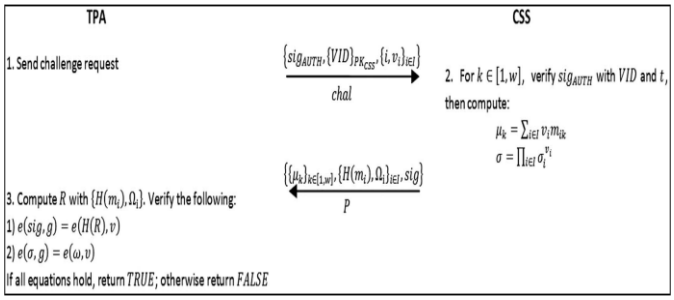
Nonetheless, as the contents in m_ are a part of the old mi, the CSS still needs to send mi back to the client. The process afterwards will be just similar to a PM-typed upgrade, with an only exception that the client will compute Rnew using fm0 i; hðm_Þ;Wig to compare to R0, instead of using fm0 i;Wig as in the PM-typed update.



vi. Analysis on Fine-Grained Dynamic

Data Updates

Following the settings in our proposed scheme, we now define a fine-grained update request for an outsourced file divided into l variable-sized blocks, where each block is consisted of si ½1; smax_ segments of a fixed size _ each. Assume an RMHT T is built upon fmigi2½1;l_ for authentication, which means T must keep updated with each RMHT operation for CSS to send back the root R for the client to verify the correctness of this operation .We now try to define and categorize all types of fine-grained updates, and then analyze the RMHT operations with Type ¼ PM;M;D; J or SP that will be invoked along with
The update of the data file.



IV. CONCLUSION

In this paper, we have provided a formal analysis on possible types of fine-grained data updates and proposed a scheme that can fully support authorized auditing and fine-grained update requests. Based on our scheme, we have also proposed a modification that can dramatically reduce communication overheads for verifications of small updates. Theoretical analysis and experimental results have demonstrated that our scheme can offer not only enhanced security and flexibility, but also significantly lower overheads for big data applications with a large number of frequent small updates such as applications in social media and business transactions. Based on the contributions of this paper on improved data auditing, we plan to further investigate the next step on how to improve other server-side protection methods for efficient data security with effective data confidentiality and availability. Besides, we also plan to investigate auditability-aware data scheduling in cloud computing. As data security is also considered as a metric of quality-of service (QoS) along with other metrics such as storage and computation, a highly efficient security-aware scheduling scheme will play an essential role under most cloud computing contexts.

V. REFERENCES

[1] R. Buyya, C.S. Yeo, S. Venugopal, J. Broberg, and I. Brandic,‘‘Cloud Computing and Emerging IT Platforms: Vision, Hype,Reality for Delivering Computing as the 5th Utility,’’ Future Gen.Comput. Syst., vol. 25, no. 6, pp. 599-616, June 2009.

[2] M.Armbrust, A. Fox, R. Griffith, A.D. Joseph, R.Katz, A.Konwinski,G. Lee, D. Patterson, A. Rabkin, I. Stoica, andM. Zaharia, ‘‘AViewof Cloud Computing,’’ Commun. ACM, vol. 53, no.4, pp. 50-58,Apr. 2010.

[3] R. Curtmola, O. Khan, R.C. Burns, and G. Ateniese, ‘‘MR-PDP:Multiple-Replica Provable Data Possession,’’ in Proc. 28th IEEEConf. on Distrib. Comput. Syst. (ICDCS), 2008, pp. 411-420.

[4] C. Erway, A. Ku¨ pc¸u¨ , C. Papamanthou, and R. Tamassia,‘‘Dynamic Provable Data Possession,’’ in Proc. 16th ACM Conf.on Comput. and Commun. Security (CCS), 2009, pp. 213-222.

[5] S.E. Schmidt, ‘‘Security and Privacy in the AWS Cloud,’’presented at the Presentation Amazon Summit Australia,Sydney,Australia,May 2012, accessed on: March 25, 2013.