Volume 3, Issue 2 (2015) 344-347 ISSN 2347 - 3258
International Journal of Advance Research and Innovation

A Fine-Grained Auditing and Verification Technique for Big Data Stored on Cloud

A. Thenmozhi, P. D. Sheba Kezia Malar Chelvi
Department of Computer Science Engineering, J. J. College of Engineering and Technology, Trichy, Tamil Nadu, India

Article Info
Article history:
Received 20 March 2015
Received in revised form
10 April 2015
Accepted 20 May 2015
Available online 15 June 2015

Keywords
Provable Data Possession,
Proof of Knowledge,
Ranked Merkle Hash Tree

Abstract
Currently, cloud computing is playing a major role in the information technology infrastructure. Cloud storage enables remote storage of user data and provides applications and services without burdening the user of local data storage and maintenance. In this scenario, the users do not maintain their data and hence data security becomes a major concern. Data stored in cloud storage need to be verified to ensure that the data has not undergone any unauthorized changes. Public auditing and verification techniques help ensure the integrity of data retrieved from cloud storage. A thorough survey of the existing techniques related to public auditing and verification is presented in this paper. The paper also presents an efficient solution to the data integrity issue that will arise while storing data in the cloud storage. The prior works related to data integrity in cloud can only support coarse grained updates. But this work supports authorized public auditing and fine-grained updates through the use of secure hash algorithm and the Ranked Merkle Hash Tree (RMHT) which can provide additional flexibility and improved efficiency.

1. Introduction

Cloud computing security is an evolving sub-domain of computer security, network security, and, more broadly, information security. It refers to a broad set of policies, technologies, and controls deployed to protect data, applications, and the associated infrastructure of cloud computing. With cloud computing, users can remotely store their data into the cloud and use on-demand high-quality applications. Using a shared pool of configurable computing resources. Data outsourcing users are relieved from the burden of data storage and maintenance. When users put their data (of large size) on the cloud, the data integrity protection is challenging. Enabling public audit for cloud data storage security is important. For example, a cloud computer facility, serves European users during European business hours with a specific application (e.g. email) while the same resources are getting reallocated and serve North American users during North America's business hours with another application (e.g. web server). This approach should maximize the use of computing powers thus reducing environmental damage as well of functions.

In section 2 of this paper we present a thorough survey of the existing auditing and verification system available in the literature. In section 3 we present the proposed fine grained auditing and verification technique and in section 4 we present the conclusion.

2. Related Work

In [1], Jules et al. express and discover evidences of retrievability. A POR scheme allows a store or back-up service (prover) to produce a short proof that a user (verifier) can retrieve a target file F, that is, that the collection holds and consistently transmits file data sufficient for the user to recover F in its entirety. A POR may be viewed as a kind of cryptographic proof of knowledge (POK), but one specially designed to handle a large file (or bit string) F. They explore POR protocols here in which the communication costs, number of memory accesses for the prover, and storage requirements of the user (verifier) are small parameters essentially independent of the length of F. In a POR, unlike a POK, neither the prover nor the verifier need actually have knowledge of F. PORs give rise to a new and unusual security definition whose formulation is another contribution of this work. PORs equally an important tool for semi-trusted online archives. The cryptographic techniques help users ensure the privacy and integrity of files they retrieve. It is also natural, however, for users to want to verify that archives do not delete or modify files prior to retrieval. The goal of a POR is to complete these checks without users having to download the files themselves. A POR can also provide quality-of-service guarantees, i.e., show that a file is retrievable within a certain time bound.

With POR protocol the verifier stores only a single cryptographic key-irrespective of the size and number of the files whose retrievability it seeks to verify—as well as a small amount of dynamic state for each file. (One simple variant of this protocol allows for the storage of no dynamic state, but yields weaker security). POR protocol encrypts F and randomly embeds a set of randomly-valued check blocks called sentinels. The use of encryption here renders the sentinels indistinguishable from other file blocks. The verifier challenges the prover by specifying the positions of a collection of sentinels and asking the prover to return the associated sentinel values. If the prover has modified or deleted a substantial portion of F, then with high probability it will also have suppressed a number of sentinels. It is therefore unlikely to respond correctly to the verifier. To protect against corruption by the prover of a small portion of F, they also employ error-correcting codes.

Practical POR systems produce a problem like they do not provide assurances about the state of individual

Corresponding Author,
E-mail address: honey3491@gmail.com

repositories. In [1], the authors did not directly address error- correction for the single-server case, and have not established formal definitions about the complete progress.

In [2], Wang et al. consider the fact that users no longer have physical possession of the outsourced data. This makes the data integrity protection in Cloud Computing a formidable task, especially for users with constrained computing resources. Moreover, users should be able to just use the cloud storage as if it is local, without worrying about the need to verify its integrity. Thus, enabling public auditability for cloud storage is of critical importance so that users can resort to a third party auditor to check the integrity of outsourced data and be worry-free. They argued that to securely introduce an effective third party auditor, the auditing process should bring in no new vulnerabilities towards user data privacy, and introduce no additional online burden to user. Thus they enabled the third party auditor to perform audits for multiple users simultaneously and efficiently. They utilize the technique of public key based homomorphic linear authenticator which enables third party auditor to perform the auditing without demanding the local copy of data and thus drastically reduced the communication and computation overhead as compared to the straightforward data auditing approaches. By integrating the homomorphic linear authenticator with random masking, the protocol guarantees that the third party auditor could not learn any knowledge about the data content stored in the cloud server during the efficient auditing process. The aggregation and algebraic properties of the authenticator further benefit this design for the batch auditing.

Specifically, their contribution can be summarized in the following three aspects: They motivate the public auditing system of data storage security in cloud computing and provide a privacy-preserving auditing protocol, i.e., this scheme enables an external auditor to audit user's outsourced data in the cloud without learning the data content. This scheme is the first to support scalable and efficient public auditing in the cloud computing. This scheme achieves batch auditing where multiple delegated auditing tasks from different users can be performed simultaneously by the third party auditor. They prove the security and justify the performance of proposed schemes through concrete experiments and comparisons with the state-of-the-art. But they did not enable secure access to storage on cloud. They did not enable the third party auditor to efficiently perform multiple auditing tasks.

In [3], Ateniese et al. have introduced a provable data possession technique that can be used for isolated data inspection. The model creates probabilistic proofs of possession by sample random sets of chunks from the server, which decreases I/O costs. The client maintains constant quantity of metadata to confirm the proof. The test or reply procedure conveys a small, continuous quantity of statistics, which reduces net communication. The authors propose a generic transformation that adds to any distant data examination arrangement based on spot checking. They present a demonstrable data possession that lets remote data examination, that is, delivers proof that a third party stores in a file. The model is lightweight, that is, by using spot examination it allows the server to check small portion of the file to make the proof; all preceding methods must access the entire file. Within this model, they give the first provably-secure scheme for remote data checking. The client supplies a minor quantity of metadata to confirm the server proof. Also, the scheme uses network bandwidth. The trial and the reply is each a little more than 1 Kilobit. They also present a more effective version of this structure that proves data possession consuming a single sectional exponentiation at the server. Both their constructions use homomorphic provable tags because of the homomorphic property, tags detected for many file blocks can be combined into a single value. The server saves chunks and their reliable labels, using them to make a proof of ownership. The client is thus convinced of data possession, without actually having to retrieve the file chunks. It can be used to minimize the file block accesses and the computation on the server. Every small update cause recomputation and updating of the authenticator for an entire file block, which in turn causes higher storage and communication overheads. They have computational burden on the server. They are not more secure in the storage process in the servers.

In [4], R.D. Pietro et al. have introduced the provable data possession. In this, they consider the issue of frequent access, efficiently and securely verifying the data in the storage server that was faithfully storing their client's outsourced data. The storage server is assumed to be untrusted in terms of both security and reliability. The problem is degraded by the clients with incomplete capitals. Prior works have handled this problem by either community key cryptography or need for the client to subcontract its data in encrypted form. In tis work the authors have developed a provably safe provable data possession method based completely on symmetric key cryptography, while eliminates the need for unpackaged encryption. Also, in difference with its precursors, their provable data possession method lets subcontracting of lively data.

Two very different approaches have been suggested in this work. The first is a public- key-based technique allowing any verifier (not just the client) to query the server and obtain an interactive proof of data possession. This property is called public verifiability. The interaction can be repeated any number of times, each time resulting in a fresh proof. The POR scheme uses special blocks (called sentinels) hidden among other blocks in the data. During the verification phase, the client asks for randomly select sentinels and checks whether they are intact. The server modifies or deletes parts of the data, then sentinels would also be affected with a certain probability. However, sentinels should be indistinguishable from other regular blocks; this implies that blocks must be encrypted. Thus POR cannot be used for public databases, such as libraries, repositories, or archives. In other words, its use is limited to confidential data. In addition, the number of queries is limited and fixed a priori. This is because sentinels, and their position within the database, must be revealed to the server at each query and a revealed sentinel cannot be reused.

In [5], Q. Wang et al. have considered the task of letting a third party auditor, by the cloud client, to authenticate the honesty of the lively data stored in the cloud. The outline of third party auditor removes the participation of the customer is the checking of whether his data stored in the cloud is certainly intact, which can be significant in attaining frugalities of scale for cloud computing. The provision for data subtleties via the most overall forms of data process, such as block alteration, supplement and removal, is also an important step toward realism. The schemes with private auditability can achieve higher scheme efficiency, public auditability allows any-one, not just the client (data owner), to challenge the cloud server for correctness of data storage while keeping no private information. The prior works on safeguarding remote data integrity often lacks the support of public auditability or lively data processes. Wang et al., scheme attains both. They first classify the problems and possible safety glitches of straight postponements by fully lively data updates from prior works. In their proposed system they have introduced a technique to provision climbable and well-organized public auditing in cloud. In specific, their scheme attains batch checking where manifold secondary checking something from diverse users can be done instantaneously by the third party auditor. They also prove the security of their proposed complex system and justify the performance of their scheme. They used PKC (Public Key Cryptography) based homomorphic authenticator for the verification protocol with public auditability. The technique can simultaneously verify in the multiple-client environment. The technique did not support the data insertion.

In [6], Peterson et al. have proposed provable data possession (PDP) that provides probabilistic proof that can be verified when a third party stores a file. The model is unique in that it allows the server to access small portions of the file in generating the proof; all other techniques must access the entire file. Within this model, they give the first provably-secure scheme for remote data checking. Using PDP technique allows outsourcing of dynamic data that is, it efficiently supports operations such as block modification deletion and append.

In [7], Shah et al. also proposed methods for auditing storage services. In their approach, a third-party auditor verifies a storage provider’s possession of an encrypted file via a challenge-response MAC over the full (encrypted) file; the auditor also verifies the storage provider’s possession of a previously committed decryption key via a conventional proof-of-knowledge protocol.

In [8], Erway et al. proposed the first PDP scheme based on skip list that can support full dynamic data updates. However, public auditability and variable-sized file blocks are not supported by default.

In [9], the authors have proposed a a public auditing scheme based on BLS signature and Merkle hash tree (MHT) that can support fine-grained update requests and our proposed is based on this scheme.

Clusters, leading to a huge number of distance computations. So, k-means++ initialization becomes inefficient. Even though scalable k-means++ presented in [4] chooses more than one centers in each pass and is proven as a good approximation of the original k-means, it still needs too many passes in practice, which incurs huge communication and I/O costs.

In the following section we present the proposed technique.

3. Proposed Public Audit Technique

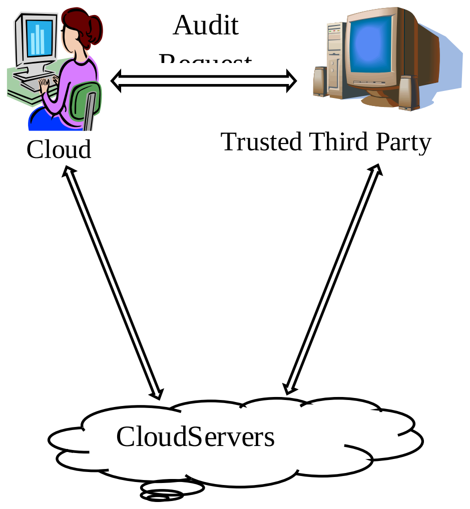

Fig: 1. Proposed Public Audit System

In this section, we present a scheme that can fully support authorized auditing and fine-grained update requests and also propose an enhancement that can dramatically reduce communication overheads for verifying small updates. Figure 1 depicts the proposed public audit system. With this proposed technique, the cloud user first decomposes the file to be stored in the cloud into smaller blocks and computes the hash code for each block. The hash codes are sent to the Trusted Third Party (TTP) and they are stored along with block numbers, file identifiers and chunking metadata. The hash on the file blocks are computed with Secure Hash Algorithm and Ranked Merkle Hash Tree.

Merkle hash tree [10] is a tree in which every non-leaf node is labeled with the hash of the labels of its children nodes. Hash trees are useful because they allow efficient and secure verification of the contents of larger data structures. The nodes in a Ranked Merkle Hash Tree store the number of blocks and the hash value computed over those blocks as shown in Figure 2 [10]. This implies that the number of blocks at each node can vary whereas MHTs use fixed number of blocks at each node. This is the key merit of RMHT compared to MHT.

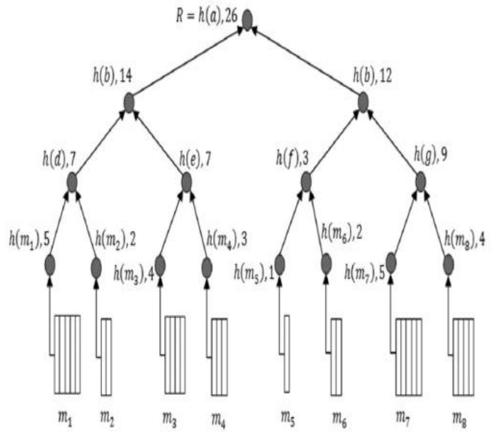

Fig: 2. A Ranked Merkle Hash Tree [10]

The files containing the actual data of the user are then stored cloud server. At a later time when a user wishes to check the integrity of the information retrieved, before actually retrieving the data the user sends an audit request to the trusted third party mentioning the file id and the block number of the block containing the data to be verified. Now the TTP sends a challenge to the cloud server asking for the file. The cloud server returns the file to the TTP. The TTP segments the file as per the chunking metadata. It then computes the hash over the block for which the user requested an audit. The TTP retrieves the hash stored for this block and compares it against the computed hash. If both match the TTP sends a positive reply to the user confirming the integrity of the data stored in the cloud. Otherwise, a negative reply is sent to the user indicating that the data stored has undergone some change and not to be trusted. This fine grained approach will well suit big data applications because the hash need not be calculated for the entire file to verify a portion of its content. Moreover when updates happen the hash need not be calculated for the entire file. Instead the hash code of the particular block which contains the updated information needs to be recalculated by the user performing updates and this alone needs to be conveyed to the TTP which can be updated in the RMHT maintained at the TTP.

4. Conclusion

In this paper an auditing technique to verify the integrity of data stored in cloud has been proposed. This proposed technique employs Secure Hash Algorithm and Ranked Merkle Hash Tree. The proposed technique allows fine grained updates to be performed without imposing much overhead on hash computation. Hence this technique will be very apt for big data applications as hash re-computation over the entire dataset will be very time consuming for such applications.

References

[1] A. Juels, J. B. S. Kaliski, PORs: Proofs of Retrievability for Large Files, 14th ACM Conference on Computer and Communications Security (CCS '07), 2007, 584-597

[2] Lou W. K. Ren, Q. Wang, Privacy- Preserving Public Auditing for Data Storage Security in Cloud Computing, In Proceedings of the 30st Annual IEEE International Conference on Computer Communications (INFOCOM'10), , 2010, 1 - 9

[3] G. Ateniese, R. Burns, R. Curtmola, J. Herring, O. Khan, L. Kissner, Z. Peterson, D. Song, Remote Data Checking Using Provable Data Possession, ACM Transactions on Information and System Security, 14(1), 2011

[4] G. Ateniese, R. D. Pietro, L.V. Mancini, G. Tsudik, Scalable and Efficient Provable Data Possession, 4th International Conference on Security and Privacy in Communication Netowrks (Secure Comm '08), 2008, 1-10

[5] W. Lou, K. Ren, J. Li .Q. Wang, C. Wang, Enabling Public Auditability and Data Dynamics for Storage Security in Cloud Computing, IEEE Transactions on Parallel and Distributed Systems, 22(5), 2011, 847 - 859

[6] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, D. Song, Provable data possession at untrusted stores. 14th ACM Conference on Computer and Communications Security (CCS'07), ACM, New York, 2007

[7] M. Baker, J. C. Mogul, M. A. Shah, R. Swaminathan, Auditing to keep online storage services honest, Presented at HotOS XI, 2007. Available at: http://www.hpl.hp.com/personal/Mehul shah/papers/hotos112007 shah.pdf.

[8] C. Erway, A. Küpçü, C. Papamanthou, R. Tamassia, "Dynamic Provable Data Possession, 16th ACM Conference on Computer and Communications Security (CCS'09), 2009, 213-222

[9] Chang Liu, Jinjun Chen, Laurence T. Yang, Xuyun Zhang, Chi Yang, Rajiv Ranjan, Ramamohanarao Kotagiri, Authorized Public Auditing of Dynamic Big Data Storage on Cloud with Efficient Verifiable Fine-Grained Updates, IEEE Transactions on Parallel & Distributed Systems, 25(9), 2014, 2234-2244

[10] R.C. Merkle, A Digital Signature Based on a Conventional Encryption Function, Int'l Cryptol. Conf. on Adv. in Cryptol. (CRYPTO), 1987, 369-378