IJCSN International Journal of Computer Science and Network, Volume 4, Issue 6, December 2015
ISSN (Online) : 2277-5420 www.IJCSN.org
Impact Factor: 0.417

Preserving Data Privacy in Third Party Cloud Audit

[1] L. Gayathri , [2] R. Ranjitha, [3] S. Thiruchadai Pandeeswari , [4] P.T. Kanmani

[1,2,3,4] Department of Information Technology
Thiagarajar College of Engineering,
Madurai-15, Tamil Nadu

Abstract - Cloud technology provides many benefits in terms of computing and storage. Accessibility from anywhere and on-demand service provision maximizing the available resource utilization are the major features of Cloud storage. However, one of the major problems in cloud storage is 'Data Integrity'. In order to ensure data integrity, it is in practice to have the cloud data audited by a third party auditor. However, this poses a threat to data privacy as the data has to be exposed for carrying out the audit process. The proposed work suggests an approach such that the Third party auditor will be able to audit the data without accessing the actual user data. Hence, privacy of the data is preserved. This is achieved by dividing data into blocks and generating hash using the latest of MAC algorithms, Secure Hash Algorithm (SHA-3). The hash values are stored in a Merkle tree for efficient organization of the hash values generated. This process is carried out on both the client end and the cloud service provider end. The top root hash value is sent to the TP auditor and individual hash values are stored in cloud storage. While performing the audit, the TP auditor requests hash values from the cloud owner. Data Integrity is confirmed by comparing the hash values obtained from the cloud owner and cloud service provider. If the values match, integrity is confirmed. The proposed approach is implemented in Hadoop and the results are analyzed.

Keywords - *Third-Party Audit, SHA-3, Merkle Tree, Data Privacy, Data Integrity.*

1. Introduction

Cloud Computing is the one of the long awaited possibility, where users has the ability to store their data remotely so that they can utilize the quality applications and services that has high resource demand, from a shared pool of configurable computing resources. Data outsourcing helps the users to get relieved from the burden of local data storage and maintenance. As the user no longer has the physical possession of the data, the integrity protection of the data has become a complicated task, especially for users with constrained computing resources and capabilities. Thus, it is important to enable public auditability for cloud data storage security so that when needed users can use an external audit party to check the integrity of outsourced data. Many solutions exist for the problem of integrity maintenance of data. The auditing can be performed in two ways either as Private or Public. In Private Auditability, the client is very important. He is responsible to verify the data. No one else other than the client can question the server regarding the data integrity. But, more than Private Auditability Public Auditability is preferred for its convenience. Because public auditability allows a third party to perform integrity verification on behalf of client. As the client here is not solely responsible for it, it largely reduces client's burden. We refer to this third party as the Third Party Auditor (TPA).

Public auditing service for cloud data storage should be enabled to ensure the full data integrity and also to reduce the computation resources of users and online burden, so that when needed users may resort to an independent third- party auditor (TPA) for auditing the outsourced data. On behalf of the users, the TPA periodically checks the integrity of the data stored in the cloud. For the users, this is a more easier and affordable way to ensure their storage correctness in the cloud.

Furthermore, in addition to help users to evaluate the risk of their subscribed cloud data services, the audit result from TPA would be beneficial for the cloud service providers to improve their cloud-based service platform, and even help for independent arbitration purposes .Hence, public auditing services will help to establish a cloud environment, where users will have ways to assess risk and gain trust in the cloud. For secure introduction of an effective third party auditor (TPA), the following are the two fundamental requirements have to be met: 1) TPA should be able to audit the data in the cloud storage without demanding the local copy of data, and also should not introduce an additional on-line burden to the cloud

user; 2) The third party auditing process should not bring any new vulnerabilities for user's data privacy. In this, we utilize secure hash algorithms and tree data structure to achieve the privacy-preserving public auditing system, which meets all above requirements. Widespread security and performance analysis shows that the schemes proposed are provably secure and highly efficient.

2. Motivation

Data security in cloud is one of the serious issues with cloud storage facility. 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. For this, auditing of the data is necessary to assure user safety of their data. In order to preserve the privacy of the user data during the process of auditing, we introduce a method in which TPA need not access the actual user data. Hence this work attempts to ensure the integrity of the data by performing third party auditing while preserving the privacy of the user data.

3. Related Work

[1] Proposed auditing of outsourced data, the scheme utilizes RSA based homomorphic tags, thus achieving public auditing. In this, the client need to verify if the server has retained file data without actually retrieving the data from server and without having the server access the entire file. By sampling random sets of blocks from the server, the model generates probabilistic proofs of possession by sampling random sets of blocks. This reduces I/O cost drastically. The Provable Data Possession [PDP] model for remote data checking supports large data sets in widely distributed storage systems. The system is subject to freeloading in which partners attempt to use storage resources and contribute none of their own. The location and physical implementation of these replicas are managed independently by each partner and will evolve over time. Partners may even outsource storage to third-party storage server providers. Efficient PDP schemes will ensure that the computational requirements of remote data checking do not unduly burden the remote storage sites. Major advantages of this approach are i) It allows the server to access small portions of the file in generating the proof, but other techniques must access the entire file. ii) It supports exchange storage capacity to achieve reliability and scalability. iii) PDP generates proofs as quickly as the disk produces data. While the advantages are discusses above, there are certain drawbacks like i) An overhead of generating metadata is imposed on client. ii) No support provided for dynamic auditing. iii) Requires more than 1kilo-byte of data for a single verification.

In [2] ,the main issue is how to frequently, efficiently and securely verify that a storage server is faithfully storing its client's (potentially very large) outsourced data. The storage server is assumed to be untrusted in terms of both security and reliability. (In other words, it might maliciously or accidentally erase hosted data; it might also relegate it to slow or off-line storage.) The problem is exacerbated by the client being a small computing device with limited resources. Prior work has addressed this problem using either public key cryptography or requiring the client to outsource its data in encrypted form. To construct a highly efficient and provably secure PDP technique based entirely on symmetric key cryptography, while not requiring any bulk encryption. Also, in contrast with its predecessors, PDP technique allows outsourcing of dynamic data, i.e, it efficiently supports operations, such as block modification, deletion and append.

Advantages of this approach are i) It makes use of less storage space (size of challenge and response is significantly less, less than a single data block). ii) It uses less bandwidth. iii) No bulk encryption of outsourced data is required, the scheme delivers better performance on client side. The disadvantages observed in this approach are i) The number of queries which can be answered is fixed priori. ii) Not applicable for dynamic data operations, supports only basic block operation with limited functionality. iii) It is a partially dynamic scheme, not fully dynamic.

[5] Proposed the scheme, dynamic auditing protocol that can support the dynamic operations of the data on the cloud servers. This scheme requires the linear combination of data blocks to be sent to the auditor for verification. The scheme makes use of a TPA for integrity verification. It also supports data dynamics via the most general forms of data operation, such as block modification, insertion and deletion. Given a file F consisting of n blocks, we define an update as either insertion of a new block (anywhere in the file, not only append), or modification of an existing block, or deletion of any block. DPDP solution is based on a new variant of authenticated dictionaries, where we use rank information to organize dictionary entries. Thus we are able to support efficient authenticated operations on files at the block level, such as authenticated insert and delete .This enables efficient proofs of a whole file system, enabling verification at different levels for different users and at the same time not having to download the whole data. This approach has advantages as follows i) It can extract the whole file with the help of error-correcting codes. ii) It efficiently maintains and update the authentication information. iii) It recomputes bottom-up the ranks of the

affected nodes in constant time per node. The disadvantages of this approach are i) The scheme may leak data content to the auditor because it requires the server to send linear combinations of data blocks to the auditor for verification. ii) The efficiency of this scheme is not clear.

[7] proposed the Proofs of Retrievability(POR) protocol verifies 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. This scheme requires that the prover access only a small portion of a (large) file F in the course of a POR. Briefly, 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 ensure data possession and retrievability, it makes use of spot checking and error correcting codes. Advantages of the approach proposed in this paper are : i) It guarantees quality-of-service. ii) File is retrievable within a certain time bound. iii) Improving network bandwidth and reliability are reducing user reliance on local resources. The drawbacks are i) Number of queries clients used is fixed priori. ii) Preprocessing of each file is needed prior to storage at the server. iii) The scheme cannot be used for public databases and can only be used for the confidential data.

Scheme proposed in [8] uses Advanced Encryption Standard(AES) along with Merkle Hash Tree to maintain the integrity of the data. AES being faster in encryption decryption and the buffer-space requirement being less as compared to RSA and try to improve the performance by making use of AES algorithm. The cloud must not impose on user the responsibility to verify his/ her stored data. Taking this into consideration and relieve client from the overhead of data integrity verification, we introduce an entity called the Third Party Auditor (TPA), which acts on behalf of client for data integrity checking and send an alert to notify the status of the stored data. The storage security scheme also assures recovery of data, in case of data loss or corruption, by providing a recovery system. Thus the scheme aims at keeping the user data integrated and support data restore. The system also reduces the server computation time when compared with previous systems. Advantages of this approach are : i) Encrypting and decrypting symmetric key data is relatively easy to do and extremely very fast. ii) Due to the use of Merkle Hash Tree, there is no loss of keys. iii) The Merkle tree allows verifying that a transaction exists in the block without having the entire block. Disadvantages are i) AES is a symmetric key cryptosystem, so it is easily breakable. ii) It has too simple algebraic structure.

To overcome the disadvantages of above schemes ,we proposed a system consisting of SHA-3 for generating the hash values and stored it in Merkle Hash Tree. After a check is performed, a notification is sent to the client about the status of user's data indicating whether the data is in its actual form or if its integrity is lost. By this data integrity is ensured.

4. Proposed System

The proposed system uses SHA-3 algorithm to generate the hash values of the data stored in the cloud. Initially before generating the hash values, the data is divided into blocks of fixed size. Size of each block is maintained in such a way that on the whole the data is divided into 8 blocks. In order to maintain uniform block size, bits are padded wherever required. Hash Value is generated for each block of data using SHA-3 algorithm. In this work, the SHA-3 hash values are generated by using Keccak function, in which the logical operations such as XOR, NOT,AND and ROT are performed on the data block twice. The resulting hash values are stored in Merkle hash tree. Block diagram of the operations involved in generating hash values are given in figure 1 below.

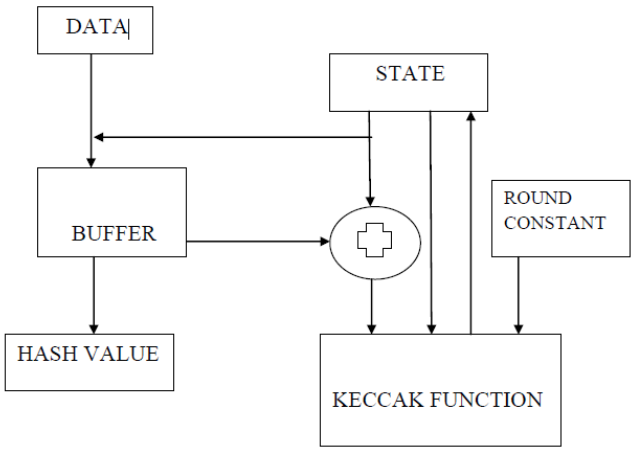

Fig 1. Hash Value Generation

The hash values are organized using Merkle hash tree (MHT) because MHT is a well-studied authentication

structure. It can be efficiently used to prove that a set of elements are undamaged and unaltered. It helps greatly in reduction of server time. The leaf nodes of the MHT are the hash values of the original file blocks.

The hash values at the leaf nodes are rehashed again repeatedly in a tree-like fashion until a single root node is obtained. This phenomenon is shown in figure 2 below.

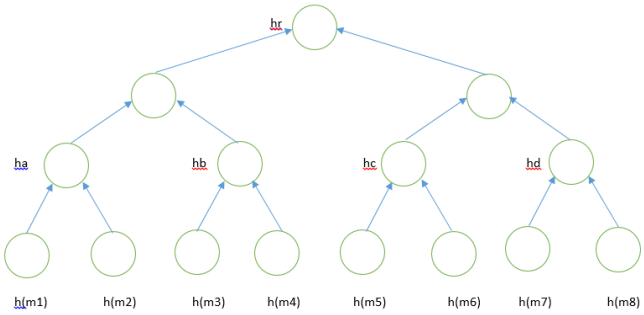

Fig 2. Merkle Hash Tree Generation

Considering the first half of the MHT shown in figure 2 above m1,m2,m3 and m4 are the data blocks of original file. h(m1), h(m2), h(m3) and h(m4) denotes the hash values of the data blocks 1,2,3,and 4 of the original file. These values are rehashed again. h(m1) and h(m2) are rehashed to generate h_a and h(m2) and h(m4) are rehashed to generate h_b . Further, h_a and h_b are rehashed once again to generate the root node h_r.

The integrity verification process, is where client initiates by sending a request to TPA for auditing the desired file or data. This is done by sending some metadata such as FileId and ClientId. The TPA generates a challenge, sends it to the Cloud Service Provider(CSP) and in response, the server generates a proof for the corresponding challenge. In the proof, the server generates the proof. The proof contains the signature of the root and the root of the MHT generated for the respective file. The verification process is done in two stages. First is file authentication and second is integrity checking. For authentication of the file, the signature of the root is checked. If it matches with signature stored during file upload, the output is given as True otherwise emits False. If the output is True, the integrity is checked by checking the value of the root with previously stored root. Any changes made to the file blocks are reflected in the value of the root. If the root does not match, it means that some changes are made to the file and the file has lost its integrity.

Integrity verification is done by checking the value of only Tags. TPA does not need to access the actual data for it. Due to this, TPA cannot view client‟s data and it makes the process Privacy-Preserving. The TPA is an entity that acts in behalf of the client. It has the expertise, capabilities, knowledge and professional skills that client does not have. It handles the work of integrity verification and reduces the overhead of the client to which, client no longer needs to verify the integrity of the data at the server on its own. The three network entities viz. the client, cloud CSP and TPA are present in the cloud environment. The client stores data on the storage server provided by the CSP. TPA keeps a check on client‟s data by periodically verifying integrity of data on-demand and notifies client if any variation of fault is found in client data. The design of the proposed system is shown in the figure 3 below.

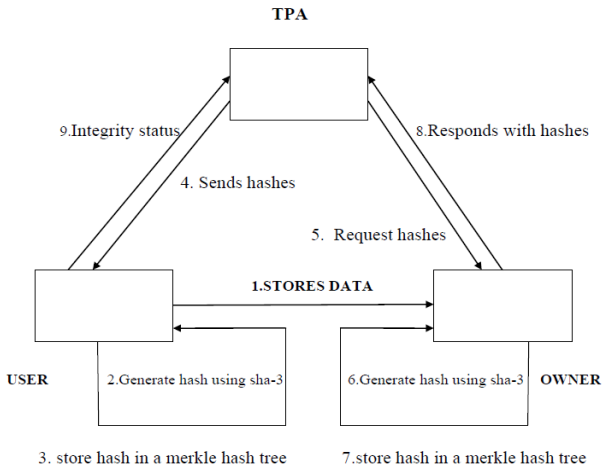

Fig 3. Design of the proposed system

The function of the system is to ensure the integrity of the data that is stored in the cloud by the user. First the user stores the data in the Hadoop as it provides a distributed environment and that is how a cloud environment looks from owner‟s percpective. This data stored is going to be checked of its integrity. But integrity cannot be checked by a person belonging to either of the party as the result may be biased. So, a third party is introduced to check this integrity, who is known as Third Party Auditor. Third party performs the auditing which is known as Third Party Auditing(TPA).But the original data cannot not be given as it is for auditing as it damages the privacy property of the data.

Hence the data is converted into hash values using hashing algorithm, in particular SHA-3.SHA-3 is better than other hashing algorithms such as SHA-2 SHA-1 and

MD5 in which collision attacks have been found. Finding two messages such that they produce same hash values is known as collision attack.But,SHA-3 resists attacks of complexity upto 2n where n is the length of the hash value generated.SHA-3 performs only simple operations like XOR and rotate. Before feeding the data into SHA-3 the data is split up into eight parts and converted into equal length by adding „0" at the end. So, after giving each part as input into the SHA-3 algorithm eight hash values are obtained. These values are then stored in the Merkle hash tree. Storing of the hash values in this tree makes the access of hashes easier. In merkle hash tree, the eight hash values are stored as leaf nodes.

The neighbouring roots are added to give its parent root. The neighbouring hashes are added until the root hash is found. This root hash is sent to the Third Party Auditor as a file. The file is transferred to the Third Party using TCP/UDP socket connection. Then the Third Party sends the request to the cloud owner to send the hashes produced from the data stored in the cloud. The same steps as done in the user side is followed i.e.

First the file is split into eight parts, then it is fed into SHA-3,then it is stored in merkle hash tree as root nodes and then the nodes are added to find the root node. This root is sent to the Third party as response to its earlier request. Hence ,now Third party is in possession of the root nodes from both the user side data and owner side data. These two node values are a string values from the file and are compared using java"s inbuilt string function. If they are equal then the integrity is maintained and if not then the data has been manipulated in the owner side and it is not the same data which the user stored in the cloud.

5. Performance Evaluation

Instead of exposing the user data for audit, the work recommends to apply some cryptographic technique and present the data in a form that does not explicitly exposes the data. In order to achieve this, SHA-3 algorithm is used in the proposed approach. However, this security measure will add upto performance overhead. In order to optimize the performance overhead and to maintain the trade-off between performance and security optimum, SHA-3 algorithm is used in the proposed work. In case, if some encryption algorithm such as AES is used in place of SHA, it will add upto the performance ooverhead heavily. Compile time comparison of AES and SHA-3 is given in the figure 4 below.

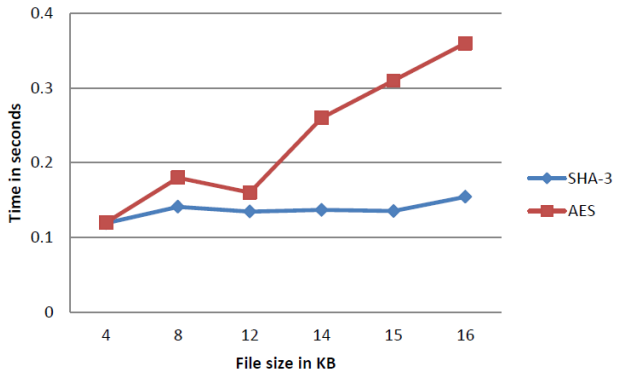

Fig 4. Compile Time Comparison

Figure 4 compares the compile time for similar file size for AES and SHA-3(Keccak).The difference in compile time is much smaller up to file size of 12KB.But it changes drastically as the input file size is increased to 14KB,15KB and so on. As seen evidently, the difference in compile time increases between AES and SHA-3 as the input file size increases. This ensures that SHA-3 is effective than AES in terms of compile time.

6. Conclusion

The proposed system guarantees the achievement of data integrity. This system also supports Public Auditing by making use of TPA and Privacy Preserving by not exposing the data to TPA during integrity verification process. Since we are using SHA-3, it resists attacks with complexity 2^n where n is the hash size. This will completely remove the burden of client and helps to keep the data safe.

References

[1] Remote integrity checking,Y. Deswarte, J. Quisquater, and A. Saidane in Proc. of Conference on Integrity and Internal Control in Information Systems (IICIS'03), November 2003.

[2] Store, forget, and check: Using algebraic signatures to check remotely administered storage,T. Schwarz and E.L. Miller, in Proceedings of ICDCS '06. IEEE Computer Society, 2006.

[3] Provable Data Possession at Untrusted Stores,G.Ateniese, Proc.14th ACM Conf. Computer and Comm. Security (CCS' 07), 2007.

[4] Proofs of Retrievability for Large Files,A. Juels, Pors ,Proc. 14th ACM Conf. Computer and Comm. Security (CCS '07), pp. 584-597,2007.

[5] Scalable and Efficient Provable Data Possession, G.Ateniese, Proc.Fourth Int'l Conf. Security and

Privacy in Comm. Networks (SecureComm '08), 2008.

[6] Dynamic ProvableData Possession,C,Erway, A.Kuocu, C. Pamanthou, R.Tamassia, Proc. 16th ACM Conf. Computer and Comm. Security (CCS'09),2009.

[7] Enabling Public Auditability and Data Dynamics for Storage Security in Cloud Computing,Cong Wang, IEEE Transactions onParallel and Distributed Systems, May 2011.

[8] Ensuring Dynamic DataStorage Security in Cloud Computing,C.Wang, Q.Wang, Kui Ren, Wenjing Lou,Proc. 17th Int'1 WorkshopQuality of Service (IWQos'09),2009.

[9] Cryptographic primitivesenforcing communication and storage complexity.P. Golle, S. Jarecki, and I. Mironov, In Financial Cryptography, pages 120-135, 2002.

[10] Ensuring Dyanmic Data Integrity with PublicAuditing for Cloud Storage",L. Chen and H. Chen,In Proc. Of International Conference on Computer Science and Service System (ICSSS' 2012), 2012.

[11] Poonam M. Pardeshi et al, / (IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 5 (5) , 2014, 6680-6685

[12] "Study on cloud computingsecurity",D.G.Feng, M. Zang, Y. Zang and Z. Xu, Journal of Software, vol.22 (1), pp. 71-83, 2011.

[13] "Data Security in the world of cloud computing",L.M. Kunfam, IEEE Security and Privacy, vol.7 (4),pp.61-64,2009.

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

[15] Data Possession inData Storage Security for Cloud Computing,M. Venkatesh, Improving Public Auditability,ICRTIT-IEEE 2012

[16] Performance Evaluation of Symmetric Encryption Algorithms, Elminaam, DiaaSalama Abdul, Hatem Mohamed Abdul Kader, and Mohie Mohamed Hadhoud. IJCSNS International Journal of Computer Science and Network Security 8.12 (2008): 280-286.

[17] "comparison of dataencryption algorithms",Simar Preet Singh, and Raman Maini, International Journal of Computer Science and Communication (IJCSC), Vol. 2, No. 1, January-June 2011, pp. 125-127

[18] A Privacy-Preserving Remote DataIntegrity Checking Protocol with Data Dynamics and PublicVerifiability ,Z. Hao, S. Zhong and N. Yu, IEEE Transactions on Knowledge and DataEngineering, Vol. 23, No. 9, September 2011