An Efficient Remote Data Integrity Checking Protocol in the Cloud Storage

Vidya Chilapur[1] Ranjana Nadagoudar[2]
[1]PG Scholar [2]Assistant Professor
[1,2]Visvesvaraya Technological University Belagavi, India

Abstract— The important application of cloud computing is cloud storage which provides flexible, scalable and high quality of data storage and computational resources to the users. Data owners choose to store their data files into cloud storage servers to reduce the storage cost. But Cloud Storage Servers are not trustworthy; hence data owner needs to check the integrity of the data files stored on the cloud servers. To address this problem a Remote Data Integrity Checking Protocol has been implemented. This Remote Data Integrity Checking Protocol allows data owner to check the integrity of their data files stored in the cloud servers and is secured against forgery attack, Replace attack and Replay attack. This protocol also allows data owner to perform data dynamic operations on the data files stored on the cloud servers.

Key words: Data Integrity, Cloud Storage

I. Introduction

By managing an enormous number of distributed computing resources in Internet, it is provides huge virtualized computing ability and storage space. Thus, cloud computing is extensively accepted and used in many real applications. Cloud service provider enables reliable, scalable, and low-cost outsourced storage service to the end-users. It provides the users with a more flexible way called pay-as-you-go model to get rich computation and storage resources on-demand. Under this pay-as-you-go model, the users can rent necessary IT infrastructures according to their requirement instead of buying them. Thus, the up-front investment of the users will be reduced greatly. In addition, it is convenient for them to adjust the capability of the borrowed resource while the scale of their applications changes.

Cloud service provider provides promising services for data storage, which saves the users costs of investment and resource. Cloud storage also provides various security issues for the outsourced data. Accident disk error or hardware failure of the cloud storage server (CSS) may cause the unexpected corruption of outsourced files. On the other side, the CSS is not fully trustworthy from the perspective of the data owner; it may actively delete or modify files for tremendous economic benefits. At the same time, CSS may hide the misbehaviours and data loss accidents from data owner to maintain a good reputation. Therefore, it is important for data owners to check the integrity for the data stored on CSS before using it.

For example, a big inter-national trading company stores all the imports and exports record files on CSS. According to these files, the company can get the key information such as the logistics quantity, the trade volume etc. If any record file is discarded or tampered, the company will suffer from a big loss which may cause bad influence on its business and development. To avoid this kind of

Circumstances, it is mandatory to check the integrity for outsourced data files. Furthermore, since these files may refer to business secret, any information exposure is unacceptable. If the company competitor can execute the file integrity checking, by frequently checking the files they may obtain some useful information such as when the file changes, the growth rate of the file etc, by which they can guess the development of the company. Thus, to avoid this situation, we consider the private verification type in our scheme, that is, the data owner is the unique verifier. In fact, the current research direction of RDPC focuses on the public verification, in which anyone can perform the task of file integrity checking with the system public key. Although RDPC with public verification seems better than that with private verification, but it is unsuitable to the scenario mentioned above.

II. Related work

Deswarte et al. [1] proposed has proposed the first Remote Data Possession Checking Protocol which was based on hash function. The disadvantage of this scheme was that it requires to access the complete file blocks for each challenge.

Ateniese et al. [2] proposed the provable data possession (PDP) model which was based on probabilistic proof technique for integrity checking without accessing the complete file. They provided two concrete schemes (S-PDP, E-PDP) which were based on RSA. These two protocols had good performance, but they didn't support dynamic operations. To overcome this problem, in 2008, they presented a dynamic PDP scheme using symmetric encryption. But this scheme still did not support block insert operation.

Sebé et al. [3] provided a RDPC protocol for critical information infrastructures based on the problem to factor large integers, which is easily adapted to support data dynamics.

Erway et al. [4] first presented a fully dynamic PDP scheme (DPDP) by using authenticated skip list, which allowed data owner to append, delete, insert and update file blocks at anytime.

Wang et al. [5] used Merkle hash tree (MHT) to propose another dynamic method for remote data checking, in which each block was hashed to be a leaf node of MHT. By sorting all leaf nodes from left to right, the MHT implicitly identified the block position which is essential for dynamic operations. However, using MHT caused heavy computation cost.

In 2013, Yang and Jia [6] presented an efficient scheme, in which an index table was utilized to support dynamic operations. By the index table, the data owner recorded the logical location and version number for each block for the outsourced file. However, to delete or insert one data block, the verifier had to find the position of the block and shift the remaining entries to insert or delete a row in the index table, which still incurred high computation cost.

[7], Chen et al. provided a dynamic RDPC scheme by using homomorphic hash function defined in [8]. Unfortunately, their scheme was proved insecure by Yu et al. [9]. To overcome the drawback, Yu et al. presented a new RDPC protocol based on RDPC scheme and proved the security. They also used MHT to achieve data dynamic operations, which caused the same shortcoming of inefficient as in [11].

In 2008, Curtmola et al. [10] first considered the remote integrity checking for multiple replicas in cloud setting. They assumed a scenario that the data owner stored certain replicas of an important file on the server, it is necessary to verify whether all these replicas are kept intact. To achieve this goal, they presented a provable secure multiple replicas PDP scheme.

III. PROPOSED METHODOLOGY

Here cloud storage system has two participants: CSS and data owner. The CSS has huge storage ability and rich computation resources; it accepts the data owner's requests to store the outsourced data files and provides access service. The data owner enjoys CSS's service and stores large amount of files to CSS without keeping backup copies in local disk. As the CSS is not trustworthy and occasionally misbehave, for example, modifying or deleting partial data files, the data owner can check the integrity for the remotely stored data efficiently.

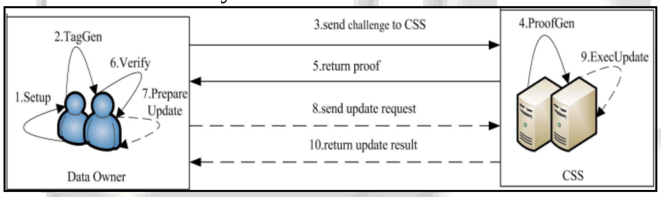

Fig. 1: Block Diagram

The RDPC has seven Algorithms

A. *KeyGen(1^k, λ_p , λ_q , m, s) → (K, sk)*

Data owner runs this algorithm to initialize the system and to generate keys. It takes inputs k, λ_p, λ_q, message sector number m and random seed s and produces homomorphic key K and private key sk as output.

B. *TagGen (K, sk, F) → T*

Data owner executes this algorithm to produce tags of the file. This algorithm takes inputs the homomorphic key, private key sk and file F and produces tag set T as output where tag set T is a sequential collection of each file block.

C. *Challenge(c) → chal*

Data owner executes this algorithm to generate challenge information to the CSS. It takes inputs as challenged block count c and produces output as challenge.

D. *ProofGen(F, T, chal) → P*

CSS runs this algorithm to generate the integrity proof P. It takes file F, tag set T and the challenge chal as input and produces the output as proof P

E. *Verify(K, sk, chal, P) → {1, 0}*

This algorithm is performed by the data owner to check the integrity of the file. To check the integrity, algorithm uses proof P returned from CSS and algorithm takes inputs homomorphic key K, private key sk and challenge chal and proof P and produces output 1 if the if the P is correct otherwise 0.

F. *PrepareUpdate(F'_i, i, UT) → URI*

This algorithm is executed by the data owner to prepare data dynamic operations on file blocks. It takes input as new file block F_i, the block position i and update type UT and produces the update request information URI. The update type has three optional elements: insert, delete, modify.

G. *ExecUpdate(URI) → {Success, Fail}*

This algorithm is executed by the CSS to execute the update operation. It takes inputs URI and produces the result. If the update operation is completed successfully, then algorithm returns success, otherwise returns Fail

IV. CONCLUSION

Presently, cloud storage has become important storage pattern and users can outsource their data files there.In this paper, I have studied the issue for integrity checking of data files outsourced to remote server and proposed an efficient Remote Data Integrity Checking protocol with data dynamic operations. This scheme uses a homomorphic hash function to verify the integrity for the data files outsourced on remote cloud server, and it reduces the storage costs and computation costs of the data owner. This scheme also enables the data owner to perform data dynamic operations such as insert, modify or delete operation on file blocks with high efficiency.

REFERENCES

[1] Y. Deswarte, J.-J. Quisquater, and A. Saïdane, "Remote integrity checking," in Proc. 6th Work. Conf. Integr. Int. Control Inf. Syst. (IICIS), 2003, pp. 1–11.
[2] G. Ateniese et al., "Provable data possession at untrusted stores," in Proc. 14th ACM Conf. Comput. Commun. Secur. (CCS), 2007, pp. 598–609.
[3] G. Ateniese et al., "Provable data possession at untrusted stores," in Proc. 14th ACM Conf. Comput. Commun. Secur. (CCS), 2007, pp. 598–609.
[4] G. Ateniese, R. Di Pietro, L. V. Mancini, and G. Tsudik, "Scalable and efficient provable data possession," in Proc. 4th Int. Conf. Secur. Privacy Commun. Netw. (SecureComm), 2008, Art. no. 9.
[5] F. Sebé, J. Domingo-Ferrer, A. Martinez-balleste, Y. Deswarte, and J. Quisquater, "Efficient remote data possession checking in critical information infrastructures," IEEE Trans. Knowl. Data Eng., vol. 20, no. 8, pp. 1034–1038, Aug. 2008.
[6] C. Erway, A. Küpçü, C. Papamanthou, and R. Tamassia, "Dynamic provable data possession," in Proc. 16th ACM Conf. Comput. Commun. Secur. (CCS), 2009, pp. 213–222.
[7] Q. Wang, C. Wang, K. Ren, W. Lou, and J. Li, "Enabling public auditability and data dynamics for

storage security in cloud computing," IEEE Trans. Parallel Distrib. Syst., vol. 22, no. 5, pp. 847–859, May 2011.

[8] K. Yang and X. Jia, "An efficient and secure dynamic auditing protocol for data storage in cloud computing," IEEE Trans. Parallel Distrib. Syst., vol. 24, no. 9, pp. 1717–1726, Sep. 2013.

[9] L. Chen, S. Zhou, X. Huang, and L. Xu, "Data dynamics for remote data possession checking in cloud storage," Comput. Electr. Eng., vol. 39, no. 7, pp. 2413–2424, 2013.

[10] M. N. Krohn, M. J. Freedman, and D. Mazieres, "On-the-fly verification of rateless erasure codes for efficient content distribution," in Proc. IEEE Symp. Secur. Privacy (S&P), May 2004, pp. 226–240.