An improved data integrity validation model for cloud storage

Zhijian Qin[1], *Lin* Huo[1,*], and *Shicong* Zhang[1]

[1]Guangxi University, Nanning, Guangxi Zhuang Autonomous Region, China

Abstract. Data integrity validation is considered to be an important tool to solve the problem that cloud subscribers cannot accurately know whether there are non-subjective changes in the data they upload to cloud servers. In this paper, a data integrity verification model based on dynamic successor tree index structure, Bloom filter and Merkle tree is proposed. The block labels generated according to the features of the dynamic successor tree index structure can sense whether changes have been made to the user's data, while the Merkle tree can track the cha[*]nged data blocks, enabling the user to effectively verify the integrity of the data stored in the cloud server and provide more effective protection for data.

1 Introduction

In recent years, researchers have proposed a variety of data integrity verification schemes based on different system and security models, mainly focusing on improving verification efficiency, reducing communication overhead, supporting blockless verification, dynamic verification and data retrievability. Although all the proposed PDP model can verify the integrity of cloud-stored data, there are still areas that can be improved in terms of security and verification efficiency. Literature [1, 2] solves this problem by using homomorphic key random mask technique, Literature [4] uses MHT to realize dynamic data operation for data integrity, Literature [3] improves MHT based on Literature [4] to realize the function of supporting fine-grained update data, and Literature [5] merges multiple copies of MHT into one MHT to improve efficiency, However, these models suffer from the security problem that CSS can falsify response evidence to spoof audits even when the data is incomplete. The literature [6] strengthens the auditing process with MHT-based coverage trees to avoid forgery attacks launched by semi-trustworthy CSS. However, the generation of data block labels in the literature requires power operations and the user-side signature is computationally intensive. In this paper, based on [6], we construct a more lightweight way of generating data block labels to reduce the computational overhead and improve the validation efficiency.

[*] Corresponding author: 15871440817@163.com

2 Preliminaries

2.1 Dynamic successor tree index structure

The dynamic successor tree index structure [7] is a tree-based index structure whose index entries contain the following information.

(1) Document ID: describes the document to which the index entry belongs.

(2) Root: Participle at the root of a successor tree.

(3) Leaf: a participle located in a leaf in a successor tree, indicating the succession of roots.

(4) Leaf Information: A list of index information pointed to by leaf nodes.

(5) Current Position Information SP: Information about the position in the document of the participle located at the index leaf node.

(6) Related Position Information RP: Information about the subsequent positions of the clauses located in the index tree nodes in the document.

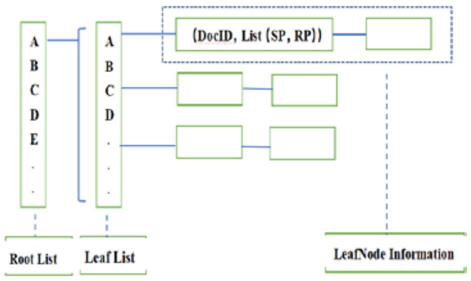

Fig. 1. Refined model of dynamic successor tree index structure.

Compared with the inverted file indexing model, the dynamic successor tree indexing structure has a slight advantage in creation efficiency and a significant advantage in retrieval efficiency, especially in large-scale text information retrieval.

2.2 Bloom filter

A Bloom filter is a probabilistic data structure which consists of a very long binary vector and a set of random mapping functions. The binary vector represents a set, which is used to determine whether an element belongs to the set or not, and the Bloom filter maps the set elements into the binary vector via a series of random mapping functions. The advantage of the Bloom filter is that its spatial efficiency and query time far exceed that of general algorithms, while the disadvantage is that it has some false positives and deletion difficulties.

2.3 Merkle tree

Merkle tree (also called Merkle Hash tree) is a type of binary tree or multi-tree based on hash value, where the values on the leaf nodes are usually the hash values of the data blocks, not the values of the leaf nodes, which are the hash values of the result of combining all the children of that node. In the computer domain, Merkle trees are mostly used for integrity processing. In scenarios dealing with integrity verification, especially when such verification is performed in a distributed environment, Merkle trees significantly reduce the amount of data to be transferred and the complexity of the computation.

3 Data integrity verification model

The system model consists of three entities: the User, the Third Party Auditor (TPA), and the Cloud Server Server (CSS).

The user in the system is the data owner. At present, the users of cloud storage involve individuals and enterprises, who have a large number of data files to be stored, but the local equipment is not enough to meet the demand for large amounts of data storage, so they relieve the pressure of local storage by storing data to the cloud server.

The cloud storage server has a large number of storage, computing, network and other broadband resources, and the user data uploaded to the cloud server is managed and maintained by the CSS, while the CSS also has to meet the user's query and dynamic update data needs, while also ensuring data security.

In order to ensure the integrity and verifiability of data, users need to check it periodically, but since neither the user nor the server can be convinced of the results obtained by the verifier, a third-party verifier. TPA, is introduced to perform the verification. However, TPA is curious about the contents of the user's data, so the user generally expects the TPA to perform only audits and does not want the contents of the data to be disclosed to the TPA.

The system model can be described as follows:

(1) The user constructs a security index for the file set $\mathrm{F} = \{f_1, f_2, f_3, \dots f_n\}$ according to the dynamic successor tree index structure, and obtains the corresponding index file I.

(2) Then divide the index file into several data blocks $\mathrm{K} = \{K_1, K_2, K_3, \dots K_n\}$, and extract n different index identifiers K_i ={keyword$_1$_LNIL$_1$, keyword$_2$_LNIL$_2$, keyword$_3$_LNIL$_3$, … keyword$_n$_LNIL$_n$ }, where keyword = RootName + LeafName, and LNIL is the length of the leaf node information table of the LeafNode under the corresponding RootNode.

(3) In order to further reduce the storage space of the block label and construct a lightweight data integrity verification scheme, this paper introduces a Bloom filter to compress the keyword set, and maps the index identifier K_i of the data block to a Bloom filter In B_i, the data block label set $\mathrm{B} = \{B_1, B_2, B_3, \dots B_n\}$ is finally obtained.

(4) At the same time, the Merkle tree structure is introduced, and the block label is used as the leaf node to create a merkle tree for each index file.

(5) After completing the operation of creating the merkle tree, the user uploads the index file and data block label collection to the cloud server for storage, and at the same time uploads the information of the root node of the merkle tree to TPA and the local file Store and delete.

(6) When the user needs to verify the integrity of the data in the cloud server, the user authorizes TPA. the TPA establishes communication with the CSS, initiates an integrity verification request to the CSS, and the CSS generates a verification response and returns it to the TPA. After that, TPA determines the verification result and sends the result to the user to complete the data integrity verification.

4 Performance analysis

In order to evaluate the performance of the proposed model in this paper, a theoretical analysis of the computational overhead related to the model is performed.

It can be seen from the analysis that the advantage of this model over the literature [6] is that the computational overhead of the TPA in this paper is reduced, and the computational overhead of the client side is the same in order of magnitude, but the client side mainly does linear computation in this paper. It can also be seen that the computational overhead of CSS in this model is higher than that in [6].

Table 1. Comparison of efficiency and function of different schemes.

Model	Public Audit	Dynamic Audit	Data Privacy Protection	Anti-Replacement Attack	User Computing Costs	CSS Computing Costs	TPA Computing Costs
MHT-PA model	√	√	×	×	O(n) mainly do exponentiation	cO(logn)	cO(logn)
Model in [6]	√	√	√	√	O(n) mainly do exponentiation	cO(logn)	cO(logn)
Our model	√	√	√	√	O(n) mainly do mapping operation	O(1)	O(n)

5 Summary

This paper is based on the index file formed by the dynamic successor tree index structure, combined with Bloom filter and merkle tree for data integrity verification, which can realize public verification and reduce computational overhead to a certain extent. Finally, the performance of the proposed model is compared and it is proved that the proposed model achieves data privacy protection and also improves the verification efficiency to some extent.

References

1. Wang C, Wang Q, Ren K, et al. Privacy-preserving public auditing for data storage security in cloud computing [C]. Proceedings of the 29th IEEE International Conference on Computer Communications. Piscataway, NJ: IEEE,2010:1－9
2. Zhu Y, Ahn G J, Hu H, et al. Dynamic audit services for out-sourced storages in clouds [J］. IEEE Transactions on Services Computing, 2013, 6(2):227－238
3. Liu C, Chen J, Yang L T, et al. Authorized public auditing of dynamic big data storage on cloud with efficient verifiable fine-grained updates [J]. IEEE Transactions on Parallel & Distributed Systems, 2014, 25(9):2234－2244
4. Wang Q, Wang C, Ren K, et al. Enabling public auditability and data dynamics for storage security in cloud computing [J]. IEEE Transactions on Parallel & Distributed Systems, 2011, 22(5):847－859
5. LIU C, RANJAN R, YANG C, et al. MuＲ-DPA: top-down levelled multi-replica Merkle hash tree based secure public auditing for dynamic big data storage on cloud [J]. IEEE Transactions on Computers, 2015, 64(9):2609－2622
6. Miao J, Feng C, Li M, Liu X. Public auditing scheme of data integrity for public cloud [J]. Computer Applications, 2018, 38(10):2892-2898
7. Huang J, Lu Z, et al. Research on Streamline Inter-relevant Successive Trees [J]. Journal of Chinese Computer Systems, 2011, 32(02):286-290.