SCALABLE AND EFFICIENT PROVABLE DATA POSSESSION

ABSTRACT: ***Data integrity is very important when a clients store their data into the cloud. Clients must be assured that the files which are stored in the cloud are secured. Mean while cloud server storage space should not be occupied by duplicate content.Most of the existing dynamic Poss, a tag used for integrity verification is generated by the secret key of the uploader.Thus, other owners who have the ownership of the file but have not uploaded it due to the cross-user deduplication on the client-side cannot generate a new tag when they update the file. The dynamic Poss would fail when someone is modified the file content, that user's ownership will be removed from common file and store the altered file in separate space. When someone is removed the file content then whose ownership is detached.***

key word: HAT tool,secret key,upload.

INTRODUCTION

Dynamic Proof of Storage (PoS) is a useful cryptographic primitive that enables a user to check the integrity of outsourced files and to efficiently update the files in a cloud server. Although researchers have proposed many dynamic PoS schemes in single user environments, the problem in multi-user environments has not been investigated sufficiently. A practical multi-user cloud storage system needs the secure client-side cross-user deduplication technique, which allows a user to skip the uploading process and obtain the ownership of the files immediately, when other owners of the same files have uploaded them to the cloud server. To the best of our knowledge, none of the existing dynamic PoSs can support this technique. In this paper, we introduce the concept of deduplicatable dynamic proof of storage and propose an efficient construction called DeyPoS, to achieve dynamic PoS and secure cross-user deduplication, simultaneously. Considering the challenges of structure diversity and private tag generation, we exploit a novel tool called Homomorphic Authenticated Tree (HAT). results show that our construction is efficient in practice.

EXISTING SYSTEM

In most of the existing dynamic PoSs, a tag used for integrity verification is generated by the secret key of the uploader. Thus, other owners who have the ownership of the file but have not uploaded it due to the cross-user deduplication on the client-side, cannot generate a new tag when they update the file. In this situation, the dynamic PoSs would fail.In most of the existing dynamic Proof of Storage, a tag used for integrity verification is generated by the secret key of the uploader. Thus, other owners who have the ownership of the file but have not uploaded it due to the cross-user deduplication on the client-side cannot generate a new tag when they update the file. In this situation, the dynamic Proof of Storage would fail. Once the files are updated the cloud server has to regenerate the complete authenticated structures for these files, which causes heavy computation cost on the server-side.

PROPOSED SYSTEM

To the best of our knowledge, this is the first work to introduce a primitive called **deduplicatable dynamic Proof of Storage** (deduplicatable dynamic PoS), which solves the structure diversity and private tag generation challenges.In contrast to the existing authenticated structures, such as skip list and Merkle tree, we design a novel authenticated structure called **Homomorphic Authenticated Tree (HAT),** to reduce the communication cost in both the proof of storage phase and the deduplication phase with similar computation cost.Note that HAT can support integrity verification, dynamic.We implemented both proof of storage and deduplication process with low communication cost. When more than one client are uploading the same file content, then that file is stored in one storage location and providing ownership to the users who are uploading same file content and we also providing proof of storage. We also presented new deduplication constructions supporting authorized duplicate check in cloud architecture, in which the duplicate-check tokens of files are generated by the private cloud server with private keys. Security analysis demonstrates that our schemes are secure in terms of insider and outsider attacks specified in the proposed security model.

MODULES

- System Construction
- Block Generation
- Deduplicatable Dynamic POS
- Homomorphic Authenticated Tree

MODULES DESCSRIPTION

System Construction:

In the first module we develop the System Construction module, to evaluate and implement a deduplicatable dynamic proof of storage and propose an efficient construction called DeyPoS. For this purpose we develop User and Cloud entities. In User entity, a user can upload a new File, Update uploaded File blocks and a user can deduplicate other users File by using deduplicatable dynamic proof of storageOur system model considers two types of entities: the cloud server and users. For each file, *original user* is the user who uploaded the file to the cloud server, while *subsequent user* is the user who proved the ownership of the file but did not actually upload the file to Cloud entity, the cloud first check login authentication of users and then it gives permission for deduplication process for authenticated users and users datas are stored in blocks. the cloud server.In the

Block Generation

In this module, we develop the Block Generation process. In the *update* phase, users may modify, insert, or delete some blocks of the files. Then, they update the corresponding parts of the encoded files and the authenticated structures in the cloud server, even the original files were not uploaded by themselves. Note that, users can update the files only if they have the ownerships of the files, which means that the users should upload the files in the upload phase or pass the verification in the Deduplication phase. Though we can

create n-blocks in this module, we split the files into 3 Blocks. The Blocks for files are divided equally accordingly and then the blocks are uploaded in the Cloud Server too.

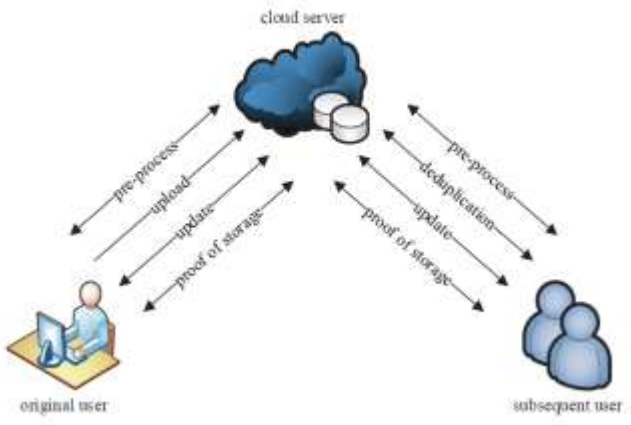
Deduplicatable Dynamic POS:

In this module we focus on a Deduplicatable Dynamic PoS scheme in multiuser environments. Deduplicatable Dynamic Proof of Storage is used to deduplicate the other users file with proper authentication but without uploading the same file. Deduplicatable Dynamic Proof of Storage (deduplicatable dynamic PoS), which solves the structure diversity and private tag generation challenges. In the deduplication phase, the files to be uploaded already exist in the cloud server. The subsequent users possess the files locally and the cloud server stores the authenticated structures of the files. Subsequent users need to convince the cloud server that they own the files without uploading them to the cloud serve.In the update phase, users may modify, insert, or delete some blocks of the files. Then, they update the corresponding parts of the encoded files and the authenticated structures in the cloud server, even the original files were not uploaded by themselves. Note that, users can update the files only if they have the ownerships of the files, which means that the users should upload the files in the upload phase or pass the verification in the deduplication phase. For each update, the cloud server has to reserve the original file and the authenticated structure if there exist other owners, and record the updated part of the file and the authenticated structure. This enables users to

Homomorphic Authenticated Tree:

(HAT).For In this module we design a novel authenticated structure called homomorphic authenticated tree reduce the communication cost in both the proof of storage phase and the deduplication phase with similar computation cost. And also HAT can support integrity verification, dynamic operations, and cross-user deduplication with good consistency.A HAT is a binary tree in which each leaf node corresponds to a data block. Though HAT does not have any limitation on the number of data blocks, for the sake of description simplicity, we assume that the number of data blocks n is equal to the number of leaf nodes in a full binary tree.Thus, for a file F = (m1, m2, m3, m4) where mı represents the ı-th block of the file. Each node in HAT consists of a four-tuple νi = (i, li, vi, ti). i is the unique index of the node. The index of the root node is 1, and the indexes increases from top to bottom and from left to right. li denotes the number of leaf nodes that can be reached from the i-th node. vi is the version number of the i-th node. ti represents the tag of the i-th node.

SYSTEM ARCHITECTURE:



SYSTEM REQUIREMENTS:

HARDWARE REQUIREMENTS:

System : Pentium Dual Core.
Hard Disk : 120 GB.
Monitor : 15'' LED
Input Devices : Keyboard, Mouse
Ram : 1GB.

SOFTWARE REQUIREMENTS:

Operating system : Windows 7.
Coding Language : JAVA/J2EE
Tool : Netbeans 7.2.1
Database : MYSQL

REFFERENCE

[1] Z. Xia, X. Wang, X. Sun, and Q. Wang, "A Secure and Dynamic Multi-Keyword Ranked Search Scheme over Encrypted Cloud Data," IEEE Transactions on Parallel and Distributed Systems, vol. 27, no. 2, pp. 340–352, 2016.

[2] Z. Xiao and Y. Xiao, "Security and privacy in cloud computing," IEEE Communications Surveys Tutorials, vol. 15, no. 2, pp. 843–859, 2013.

[3] C. A. Ardagna, R. Asal, E. Damiani, and Q. H. Vu, "From Security to Assurance in the Cloud: A Survey," ACM Comput. Surv., vol. 48, no. 1, pp. 2:1–2:50, 2015.

[4] G. Ateniese, R. Di Pietro, L. V. Mancini, and G. Tsudik, "Scalable and Efficient Provable Data Possession," in Proc. of SecureComm, pp. 1–10, 2008