RESEARCHERID
THOMSON REUTERS

ISSN: 2277-9655
[Shubhangi* *et al.*, 6(8): August, 2017] Impact Factor: 4.116
IC™ Value: 3.00 CODEN: IJESS7

IJESRT

INTERNATIONAL JOURNAL OF ENGINEERING SCIENCES & RESEARCH TECHNOLOGY

DE-DUPLICABLE EFFECTIVE VALIDATION of CAPACITY for DYNAMIC USER ENVIRONMENT

Dr. Shubhangi D C[*1] & Pooja[2]

[*1]HOD, Department of Computer Science & Engineering, VTU PG Centre, Kalaburgi, Karnataka, India

[2]P.G. Student, Department of Computer Science & Engineering, VTU PG Centre, Kalaburgi, Karnataka, India

DOI: 10.5281/zenodo.839115

ABSTRACT

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). We prove the security of our construction, and the theoretical analysis and experimental results show that our construction is efficient in practice.

KEYWORDS: Cloud Storage, Dynamic Proof of Storage, De-duplication.

I. INTRODUCTION

Cloud storage gains lots of attention now days due to its advantages of low cost, high accessibility, and easy sharing. Lots of companies such as Microsoft, Amazon, and Google started to provide their own cloud services, where users are able to upload their files to the servers, users can access the data through using various devices, and they can also share that data with other users. But despite this advantages cloud storage services still has many security issues and threats.

Data integrity is among the most important feature when a user outsourced his files to cloud storage for storing. User must get convinced that as the files stored in cloud server will not be modified. Previous techniques for preserving data integrity such as message authentication codes (MACs) and one more technique by name digital signatures need the users to download all the files from the cloud server for validation process which requires lots of communication cost. For this reason these techniques are not at all suitable for storage services where users have to check the integrity frequently mostly in each hour. Due to these researches developed proof of storage (PoS).for checking the integrity without the need of downloading the files from cloud server. Moreover, users also need several dynamic operations such as modification, insertion, and deletion, to update their files while maintaining the capability of PoS.

To perform dynamic operations Dynamic PoS is proposed. This employs authenticated structures known as Markel tree. Which means when dynamic operations are executed, users reconstruct tags (which may be used for integrity checking, such as MACs and signatures) for the updated blocks only, instead of all blocks.

II. RELATED WORK

The main idea of PoS is to randomly choose a few data blocks as the challenge. Then, the cloud server returns the challenged data blocks and their tags as the response. Since the data blocks and the tags can be combined via

homomorphic functions, the communication costs are reduced. Introducing the concept of proof of ownership which is a solution of cross-user deduplication on the client-side. It requires that the user can generate the Merkle tree without the help from the cloud server, which is a big challenge in dynamic PoS. System proposed another proof of ownership scheme which improves the efficiency and a client-side deduplication scheme for encrypted data, but the scheme employs a deterministic proof algorithm which indicates that every file has a deterministic short proof. Thus, anyone who obtains this proof can pass the verification without possessing the file locally. Other deduplication schemes for encrypted data were proposed for enhancing the security and efficiency. Note that, all existing techniques for cross-user deduplication on the client-side were designed for static files. 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.

Our System proposed a solution called proof of storage with deduplication, which is the first attempt to design a PoS scheme with deduplication. Introducing proofs of ownership and retrievability, which are more efficient in terms of computation cost. Considering proof of storage for multi-user updates, but those schemes focus on the problem of sharing files in a group. Deduplication in these scenarios is to deduplicate files amon different groups. Unfortunately, these schemes cannot support deduplication due to structure diversity and private tag generation. In this paper, we consider a more general situation that every user has its own files separately. Hence, we focus on a deduplicatable dynamic PoS scheme in multiuser environments.

The major techniques used in PoS and dynamic PoS schemes are homomorphic message authentication codes and homomorphic signatures. With the help of homomorphism, the messages and MACs/signatures in these schemes can be compressed into a single message and a single MAC/signature. Therefore, the communication cost can be dramatically reduced. These techniques have been used in PoS and secure network coding.

The main contributions of this paper are as follows.

- 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 operations, and cross-user deduplication with good consistency.

- We propose and implement the first efficient construction of deduplicatable dynamic PoS called DeyPoS, which supports unlimited number of verification and update operations. The security of this construction is proved in the random oracle model, and the performance is analyzed theoretically and experimentally.

III. METHODOLOGY

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 storage.
- Our 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 the cloud server.
- In the 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 asymptotic performance of our scheme in comparison with related schemes, where n denotes the number of blocks, b denotes the number of the challenged blocks, and |m| denotes the size of one block. From the table, we observe that our scheme is the only one satisfying the cross-user deduplication on

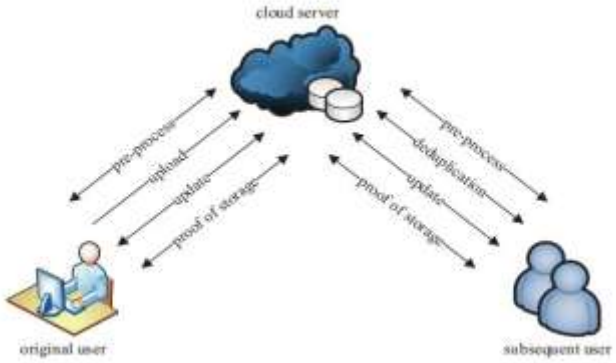
the client-side and dynamic proof of storage simultaneously. Furthermore, the asymptotic performance of our scheme is better than the other schemes except which only provides weak security guarantee.

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 DynamicProof of Storage(deduplicatable dynamic PoS), which solves the structure diversity and private tag generation challenges.
- The main process of this module is 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 the cloud server. There are five phases in a deduplicatable dynamic PoS system: pre-process, upload, deduplication, update, and proof of storage as shown in below figure.



- In the pre-process phase, users intend to upload their local files. The cloud server decides whether these files should be uploaded. If the upload process is granted, go into the upload phase; otherwise, go into the deduplication phase.
- In the upload phase, the files to be uploaded do not exist in the cloud server . The original users encodes the local files and upload them to the cloud server.
- 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 server.
- 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 update a file concurrently in our model, since each update is only "attached" to the original file and authenticated structure.
- In the proof of storage phase, users only possess a small constant size metadata locally and they want to check whether the files are faithfully stored in the cloud server without downloading them. The files

ISSN: 2277-9655
Impact Factor: 4.116
CODEN: IJESS7

may not be uploaded by these users, but they pass the deduplication phase and prove that they have the ownerships of the files.

Homomorphic Authenticated Tree:

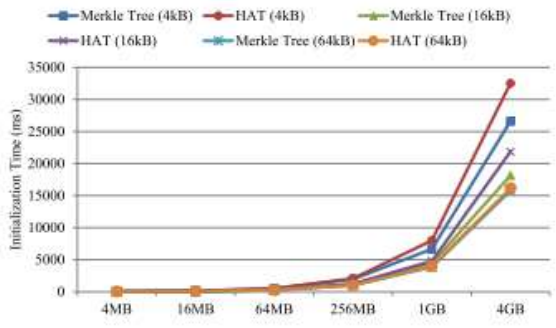
- In this module we design a novel authenticated structure called homomorphic authenticated tree(HAT).For reduce the communication cost in both the proof of storage phase and the deduplication phase with similar computation cost. And also HAT can support integrity verification, dynamic operations, and 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 vi = (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.
- When a HAT is initialized, the version number of each leaf is 1, and the version number of each non-leaf node is the sum of that of its two children. For the i-th node, mi denotes the combination of the blocks corresponding to its leaves. The tag ti is computed from F(mi), where F denotes a tag generation function.

.

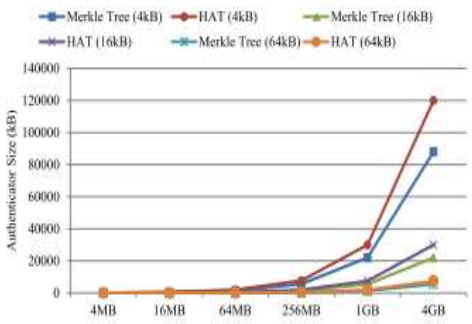
IV. EXPERIMENTAL RESULTS

We first evaluate the cost in the upload phase.

- Fig. 1 presents the initialization time for constructing Merkle trees and HATs with different sizes of files and blocks. The initialization time is similar in all schemes.
- The storage cost of the client is O(1), and the storage cost of the server is shown in Fig. 2. The authenticator size of HAT is lager than that of the Merkle tree. When the block size is 4kb, the authenticator size is less than 3 percent of the file size in our scheme.
- Fig. 3 shows the communication cost of different file sizes, where the block size is fixed on 4 kB. When the number of challenged file blocks are fixed, the communication cost stays at a steady level in Merkle tree based schemes since the major cost is to transmit the corresponding file blocks. However, the communication cost grows logarithmically with respect to the file size because the number of nodes in the sibling set grows logarithmically.
- The computation cost on the server-side in the deduplication phase are shown in Figs.4. The file size is 1 GB. The computation cost of Merkle tree based schemes on both the server-side and the client-side have better performances.

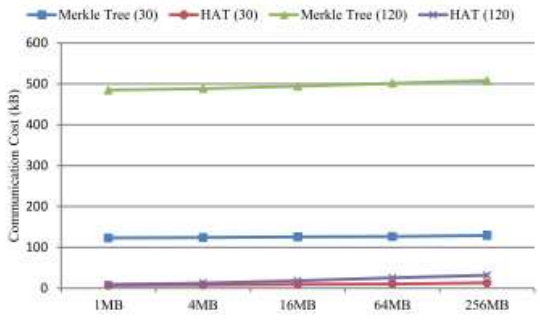


(1)

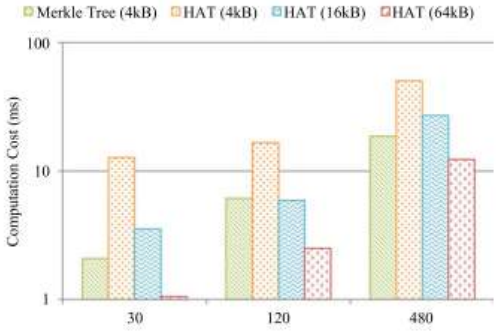


(2)

ISSN: 2277-9655
Impact Factor: 4.116
CODEN: IJESS7

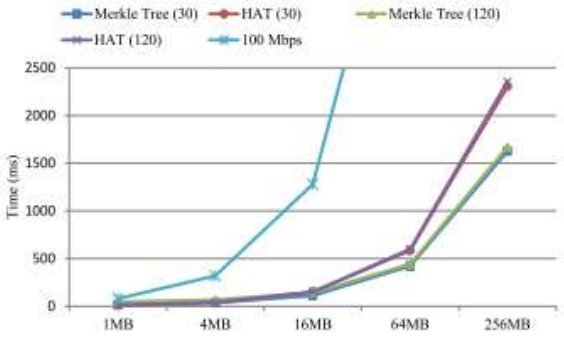


(3)

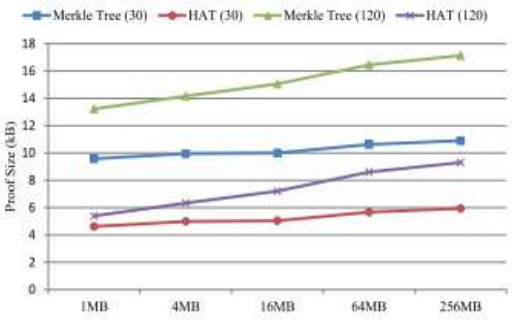


(4)

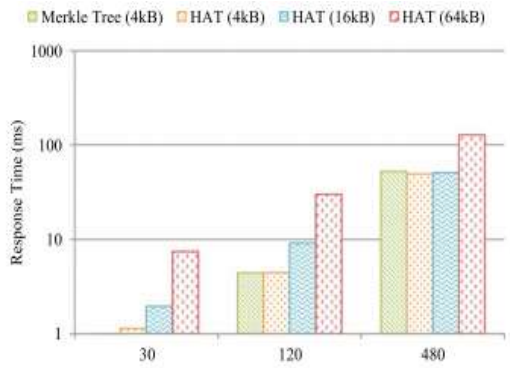
- Fig.5 presents the comparison of the performance among DeyPoS, Merkle tree based schemes, and directly transmitting the file with 100 Mbps upload speed. The overall performance is calculated with Tc þ Ts þ Tn, where T c is the computation cost on the client-side, Ts is the computation cost on the server-side, and Tn is the transmission delay on the 100 Mbps channel.
- Fig.6 presents the proof size of different file sizes, where the block size is fixed on 4 kB. Obviously, DeyPoS requires less bandwidths in all situations. When the block size is 4 kB , the block size is less than 10 kB.
- Figs.7 and 8 show the computation cost in the proof of storage phase for the cloud server and users, respectively. The computation cost on the server-side are almost the same in DeyPoS and Merkle tree based solutions.



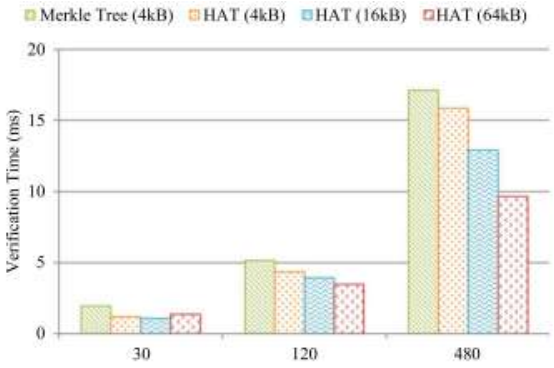
(5)



(6)



(7)



(8)

V. CONCLUSION

We proposed the comprehensive requirements in multiuser cloud storage systems and introduced the model of deduplicatable dynamic PoS. We designed a novel tool called HAT which is an efficient authenticated structure. Based on HAT, we proposed the first practical deduplicatable dynamic PoS scheme called DeyPoS and proved

its security in the random oracle model. The theoretical an experimental results show that our DeyPoS implementation is efficient, especially when the file size and the number of the challenged blocks are large..

VI. REFERENCES

[1] S. Kamara and K. Lauter, "Cryptographic cloud storage," in Proc. 14th Int. Conf. Financial Cryptography Data Security, 2010, pp. 136–149.
[2] Z. Xia, X. Wang, X. Sun, and Q. Wang, "A secure and dynamic multi-keyword ranked search scheme over encrypted cloud data," IEEE Trans. Parallel Distrib. Syst., vol. 27, no. 2, pp. 340–352, Feb. 2016.
[3] Z. Xiao and Y. Xiao, "Security and privacy in cloud computing," IEEE Commun. Surveys Tuts., vol. 15, no. 2, pp. 843–859, Jul. 2013.
[4] 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.
[5] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, and D. Song, "Provable data possession at untrusted stores," in Proc. 14th ACM Conf. Comput. Commun. Security, 2007,pp. 598–609.
[6] G. Ateniese, R. Di Pietro, L. V. Mancini, and G. Tsudik, "Scalable and efficient provable data possession," in Proc. 4th Int. Conf. Security Privacy Commun. Netow., 2008, pp. 1–10.
[7] G. Ateniese, S. Kamara, and J. Katz, "Proofs of storage from homomorphic identification protocols," in Proc. 15th Int. Conf. Theory Appl. Cryptology Inf. Security: Adv. Cryptol., 2009, pp. 319–333.
[8] C. Erway, A. Kupc€ u, C. Papamanthou, and R. Tamassia,€ "Dynamic provable data possession," in Proc. 16th ACM Conf. Comput. Commun. Security, 2009, pp. 213–222.
[9] R. Tamassia, "Authenticated data structures," in Proc. 11th Annu. Eur. Symp. Algorithm, 2003, pp. 2–5.
[10] Q. Wang, C. Wang, J. Li, K. Ren, and W. Lou, "Enabling public verifiability and data dynamics for storage security in cloud computing," in Proc. 14th Eur. Conf. Res. Comput. Security, 2009, pp. 355–370.
[11] F. Armknecht, J.-M. Bohli, G. O. Karame, Z. Liu, and C. A. Reuter, "Outsourced proofs of retrievability," in Proc. 21st ACM Conf. Comput. Commun. Security, 2014, pp. 831–843.
[12] H. Shachamand B. Waters,"Compact proofs of retrievability," J. Cryptol., vol. 26, no. 3, pp. 442–483, 2013.

CITE AN ARTICLE

C, Shubhangi D., Dr, and Pooja. "DE-DUPLICABLE EFFECTIVE VALIDATION of CAPACITY for DYNAMIC USER ENVIRONMENT." *INTERNATIONAL JOURNAL OF ENGINEERING SCIENCES & RESEARCH TECHNOLOGY* 6.8 (2017): 51-56. Web. 5 Aug. 2017.