IJMTARC – VOLUME – V – ISSUE - 18 – JUNE , 2017

ISSN: 2320-1363

# AN EFICIENT CONSTRUCTION CALLED DEYPOS , TO ACHIEVE DYNAMIC POS AND SECURE CROSS-USER DEDUPLICATION

*RAGHUVEER REDDY

*PG SCHOLAR, DEPARTMENT OF CSE, BV RAJU INSTITUTE OF TECHNOLOGY

## 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 singleuser 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.

**Index Terms**—Cloud storage, dynamic proof of storage, deduplication.

## INTRODUCTION

STORAGE outsourcing is becoming more and more attractive to both industry and academia due to the advantages of low cost, high accessibility, and easy sharing. As one of the storage outsourcing forms, cloud storage gains wide attention in recent years [1] [2]. Many companies, such as Amazon,

IJMTARC – VOLUME – V – ISSUE - 18 – JUNE , 2017

ISSN: 2320-1363

Google, and Microsoft, provide their own cloud storage services, where users can upload their files to the servers, access them from various devices, and share them with the others. Although cloud storage services are widely adopted in current days, there still remain many security issues and potential threats [3] [4]. Data integrity is one of the most important properties when a user outsources its files to cloud storage. Users should be convinced that the files stored in the server are not tampered. Traditional techniques for protecting data integrity, such as message authentication codes (MACs) and digital signatures, require users to download all of the files from the cloud server for verification, which incurs a heavy communication cost [5]. These techniques are not suitable for cloud storage services

where users may check the integrity frequently, such as every hour [6]. Thus, researchers introduced Proof of Storage (PoS) [7] for checking the integrity without downloading files from the cloud server. Furthermore, users may also require several dynamic operations, such as modification, insertion, and deletion, to update their files, while maintaining the capability of PoS. Dynamic PoS [8] is proPoSed for such dynamic operations. In contrast with PoS, dynamic PoS employs authenticated structures [9], such as the Merkle tree [10]. Thus, when dynamic operations are executed, users regenerate tags (which are used for integrity checking, such as MACs and signatures) for the updated blocks only, instead of regenerating for all blocks. To better understand the following contents, we present more details about PoS and dynamic PoS. In these schemes [5] [8] [11], each block of a file is attached a (cryptographic) tag which is used for verifying the integrity of that block. When a verifier wants to check the integrity of a file, it randomly selects some block indexes of the file, and sends them to the cloud server. According to these challenged indexes, the cloud server returns the corresponding blocks along with their tags. The verifier checks the block integrity and index correctness. The former can be directly guaranteed by cryptographic tags. How to deal with the latter is the major difference between PoS and dynamic PoS. In most of the PoS schemes [5] [11] [12], the block index is "encoded" into its tag, which means the verifier can check the block integrity and index correctness simultaneously. However, dynamic PoS cannot encode the block indexes into tags, since the dynamic operations may change many indexes of non-updated blocks, which incurs unnecessary computation and communication cost. For example, there is a file consisting of 1000 blocks, and a new block is inserted behind the second block of the file. Then, 998 block indexes of the original file are changed, which means the user has to generate and send 999 tags for this update. Authenticated structures are introduced in dynamic PoSs [8] [13] [14] to solve this challenge. As a result, the tags are attached to the authenticated structure rather than the block indexes. Taking the Merkle tree in Fig. 1a as an example (Merkle tree is one of the most efficient authenticated structures in dynamic PoS [14]), the tag corresponding to the second file block involves the index of the Merkle tree node v5, that is 5, rather than 2. When a new block is inserted behind the second file block, the authenticated structure turns into thestructure in Fig. 1b. Then, the index in the tag corresponding to the second file block changes, and the user only has to generate 2 tags for this update. This figure provides an instance that authenticated structure used in dynamic PoS reduces the computation cost in the update process. However, dynamic PoSremains to be improved in a multi-user environment, due to the requirement of cross-user deduplication on the client-side [15]. This indicates that users can skip the uploading process and obtain the ownership of files immediately, as long as the uploaded files already exist in the cloud server. This technique can reduce storage space for the cloud server [16], and save transmission bandwidth for users. To the best of our knowledge, there is no dynamic PoS that can support secure cross-user deduplication. There are two challenges in order to solve this problem. On one hand, the authenticated structures used in dynamic PoSs, such as skip list [8] and Merkle tree [14], are not suitable for deduplication. We call this challenge structure diversity, which means the authenticated structure of a file in dynamic PoS may have some conflicts. For instance, the authenticatedstructure of a file F is shown in Fig. 1a. When the file is updated to F′, the authenticated structure stored on the server-side may turn into the structure in Fig. 1b. However, an owner who intends to upload F′ usually generates a structure as shown in Fig. 1c, which is different from the structure stored in the cloud server. Thus, the owner cannot execute deduplicationunlessthe owner and the cloud server synchronize the authenticated structure. On the other hand, even if cross-user deduplication is achieved (for example, the cloud server sends the entire authenticated structure to the owner), private tag generation is still a challenge for dynamic operations. 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. If we take dynamic PoS and cross-user deduplication on the client-side as orthogonal issues, we may simply combine the existing dynamic PoS schemes and deduplication techniques. Then, structure diversity is solved via deduplication scheme. For solving private tag generation, each owner can generate its own authenticated structure and upload the structure to the cloud server, which means that the cloud server stores multiple authenticated structures for each file. Also, when a file is updated by a user, the cloud server has to update the corresponding authenticated structure in dynamic PoS, and construct a new authenticated structure for deduplication. As a result, this trivial combination in

troduces unnecessary computation and storage cost to the cloud server. Taking the combination of [10] and [15] as example, [10] is a dynamic PoS scheme which employs Merkle tree as its authenticated structure, and [15] is a crossuserdeduplication scheme which also employs Merkle tree as its authenticated structure. SupPoSe Alice and Bob independently own a file F, a Merkle tree TF is generated and stored by the cloud server for deduplication, and two Merkle trees TA and TB are generated by Alice and Bob respectively, and stored in the cloud server for PoS. When Alice updates F to F′, the cloud server updates TA to T′ A for PoS and generates a new Merkle tree TF ′ for deduplication. Thus, the number of Merkle trees grows with the version numbers and the number of owners, which is 4 (TF,T′ A,TB, and TF ′) in the above example. Also, the cloud server has to generate two Merkle trees in the above example which is more time-consuming than update the Merkle trees. As a summary, existing dynamic PoSs cannot be extended to the multi-user environment.

## RELATED WORK

The concept of proof of storage was introduced by Ateniese et al. [5], and Juels and Kaliski [17], respectively. 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. The subsequent works [11] [18] [19] [20] [21] [22] [23] [24] [25] extended the research of PoS, but those works did not take dynamic operations into account. Erway et al. [8] and later works [13] [14] [26] [27] [28] [29] focused on the dynamic data. Among them, the scheme in [14] is the most efficient solution in practice. However, the scheme is stateful, which requires users to maintain some state information of their own files locally. Hence, it is not appropriate for a multiuser environment. Halevi et al. [15] introduced 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.Pietro and Sorniotti [30] proPoSed another proof of ownership scheme which improves the efficiency. Xu et al. [31] proPoSed 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 [32] [33] [34] 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. Zheng and Xu [35] proPoSed a solution called proof of storage with deduplication, which is the first attempt to design a PoS scheme with deduplication. Du et al. [36] introduced proofs of ownership and retrievability, which aresimilar to [35] but more efficient in terms of computation cost. Note that neither [35] nor [36] can support dynamic operations. Due to the problem of structure diversity and private tag generation, [35] and [36] cannot be extended to dynamic PoS. Wang et al. [37] [38], and Yuan and Yu [39] considered 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 among 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 [40] and homomorphic signatures [41] [42]. 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 [7] [14] [18] and secure network coding [43] [44] [45]. A brief survey of homomorphic MACs and signatures could be referred in [46].

## 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.

- ❖ Haleviet *al.* introduced the concept of proof ofownership which is a solution of cross-user deduplicationon the client-side. It requires that the user can generate theMerkle tree without the help from the cloud server, which isa big challenge in dynamic PoS.
- ❖ Pietro and SorniottiproPoSed another proof of ownership scheme which improvesthe efficiency.
- ❖ Xu*et al.*proPoSed a client-sidededuplication scheme for encrypted data, but the schemeemploys a deterministic proof algorithm which indicatesthat every file has a deterministic short proof. Thus, anyonewho obtains this proof can pass the verification withoutPoSsessing the file locally.

**DISADVANTAGES OF EXISTING SYSTEM:**

- ❖ Existing dynamic PoSs cannot be extended to the multi-user environment.
- ❖ 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.
- ❖ Due to the problem of structure diversity and private tag generation, existing system cannot be extended to dynamic PoS.
- ❖ Unfortunately, these schemes cannot support deduplication due to structure diversity and private tag generation.

**PROPOS ED 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*

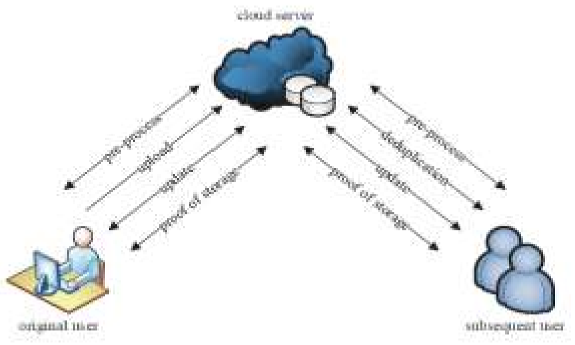
(HAT), to reduce the communication cost in both the proof of storage phase and the deduplication phase with similar computation cost.

- ❖ Note that HAT can support integrity verification, dynamic operations, and cross-user deduplication with good consistency.
- ❖ We proPoSe and implement the first efficient construction of deduplicatable dynamic PoS called Dey-PoS, 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.

## ADVANTAGES OF PROPOS ED SYSTEM:

- ❖ It is an efficient authenticated structure.
- ❖ It is the first practical deduplicatable dynamic PoSscheme called DeyPoS and proved its security in the randomoracle model.
- ❖ The theoretical and experimental resultsshow that our DeyPoS implementation is efficient,
- ❖ Performs better especiallywhen the file size and the number of the challenged blocksare large.

## SYSTEM ARCHITECTURE:



## CONCLUSION

We proPoSed the comprehensive requirements in multi-user 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 and experimental results show that ourDeyPoSimplementationis efficient, especially when the file size and the number of the challenged blocks are large.

## REFERENCES

[1] S. Kamara and K. Lauter, "Cryptographic cloud storage," in Proc. of FC, pp. 136–149, 2010. [2] 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.

[3] Z. Xiao and Y. Xiao, "Security and privacy in cloud computing," IEEE Communications Surveys Tutorials, vol. 15, no. 2, pp. 843–859, 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. of CCS, pp. 598–609, 2007.

[6] 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.

[7] G. Ateniese, S. Kamara, and J. Katz, "Proofs of storage from homomorphic identification protocols," in Proc. of ASIACRYPT, pp. 319–333, 2009.

[8] C. Erway, A. Ku¨pcu¨, C. Papamanthou, and R. Tamassia, "Dynamic provable data PoSsession," in Proc. of CCS, pp. 213–222, 2009.

[9] R. Tamassia, "Authenticated Data Structures," in Proc. of ESA, pp. 2–5, 2003.

[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. of ESORICS, pp. 355–370, 2009. [11] F. Armknecht, J.-M.Bohli, G. O. Karame, Z. Liu, and C. A. Reuter, "Outsourced proofs of retrievability," in Proc. of CCS, pp. 831–843, 2014.

[12] H. Shacham and B. Waters, "Compact Proofs of Retrievability," Journal of Cryptology, vol. 26, no. 3, pp. 442–483, 2013.

[13] Z. Mo, Y. Zhou, and S. Chen, "A dynamic proof of retrievability (PoR) scheme with o(logn) complexity," in Proc. of ICC, pp. 912– 916, 2012.

[14] E. Shi, E. Stefanov, and C. Papamanthou, "Practical dynamic proofs of retrievability," in Proc. of CCS, pp. 325–336, 2013.

[15] S. Halevi, D. Harnik, B. Pinkas, and A. Shulman-Peleg, "Proofs of ownership in remote storage systems," in Proc. of CCS, pp. 491– 500, 2011.

[16] J. Douceur, A. Adya, W. Bolosky, P. Simon, and M. Theimer, "Reclaiming space from duplicate files in a serverless distributed file system," in Proc. of ICDCS, pp. 617–624, 2002. [17] A. Juels and B. S. Kaliski, Jr., "PORs: Proofs of retrievability for large files," in Proc. of CCS, pp. 584–597, 2007.

[18] H. Shacham and B. Waters, "Compact proofs of retrievability," in Proc. of ASIACRYPT, pp. 90–107, 2008.

[19] Y. Dodis, S. Vadhan, and D. Wichs, "Proofs of retrievability via hardness amplification," in Proc. of TCC, pp. 109–127, 2009.

[20] K. D. Bowers, A. Juels, and A. Oprea, "HAIL: A high-availability and integrity layer for cloud storage," in Proc. of CCS, pp. 187–198, 2009.

[21] C. Wang,Q. Wang, K. Ren,andW.Lou, "Privacy-preserving public auditing for data storage security in cloud computing," in Proc. of INFOCOM, pp. 1–9, 2010.

[22] G. Ateniese, R. Burns, R. Curtmola, J. Herring, O. Khan, L. Kissner, Z. Peterson, and D. Song, "Remote data checking using provable data PoSsession," ACM Transactions on Information System Security, vol. 14, no. 1, pp. 1–34, 2011.

[23] Y. Zhu, H. Hu, G.-J.Ahn, and M. Yu, "Cooperative provable data PoSsession for integrity verification in multicloud storage," IEEE Transactions on Parallel and

Distributed Systems, vol. 23, no. 12, pp. 2231–2244, 2012.

[24] J. Xu and E.-C.Chang, “Towards efficient proofs of retrievability,” in Proc. of ASIACCS, pp. 79–80, 2012.

[25] J. Chen, L. Zhang, K. He, R. Du, and L. Wang, “Message-locked proof of ownership and retrievability with remote reparing in cloud,” Security and Communication Networks, 2016.