Optimal Data Deduplication in Cloud with Certificate Authority for Multiuser Environment

PINJARI MASOOM BASHA[1], A. EMMANUEL RAJU[2]
[1]PG Scholar, Dept of CSE, Dr. K.V. Subba Reddy Institute of Technology, Kurnool, AP, India,
[2]Assistant Professor, Dept of CSE, Dr. K.V. Subba Reddy Institute of Technology, Kurnool, AP, India.

ABSTRACT

Cloud computing has become an advanced platform for personal computing. Cloud computing will provide high performance computing resources and makes it possible to store and port large amounts of data across computers, servers and within IT environment. In computing, data deduplication is a special data compression technique for removing duplicate copies of repeated data without compromising its uniqueness. A multi-user cloud storage system needs the secure cross-user client-side Deduplication process, which allows a user to skip the upload process and get the rights of the document immediately, when other owners of the same document who tries to upload the document to the cloud. To the best of knowledge, existing dynamic PoS cannot support this technique. In this paper, introduced the theory of Deduplicatable dynamic proof of storage and proposed an effective system called DeyPoS, to accomplish dynamic PoS and secure cross-user Deduplication, synchronously. Considering the problems of structure, diversity and private tag generation, and make full use of a structure called Homomorphic Authenticated Tree (HAT). Furthermore, we extend the system by adding the concept of Certificate Authority for secure authentication.

I. INTRODUCTION

Cloud storage is a model of networked enterprise storage where data is stored in logical pools of storage, which are hosted by third party companies. Cloud storage provides customers with benefits, ranging from cost saving and simplified convenience, to mobility opportunities and scalable service. These great features attract more customers make use and store their personal information to the cloud storage: according to the report, the volume of data in cloud is expected to achieve 40 trillion gigabytes in 2020. Even though cloud storage system has been extensively accepted, it fails to hold some important developing needs such as the abilities of auditing virtue of cloud documents by clients and detecting cloned documents by servers. We exhibit both problems below. The first problem is virtue auditing. The cloud server is able to mitigate clients from the overburden of storage management and maintenance. The variation between cloud storage and conventional in-house storage is that the data is transferred through Internet and stored in a dubious domain, not under control of the clients, which surely raises client's great concerns on the virtue of their data. These concerns arise from the fact that the cloud storage is vulnerable to security threats from outside and inside of the cloud storage, and the undisciplined cloud servers may calmly suppress some data loss events from the clients to maintain their prestige. What is more serious is that for saving money and space, the cloud servers might even actively and knowingly remove hardly accessed data documents belonging to an ordinary client. Considering the large size of the outsourced data documents and the clients forced resource capabilities, the first problem is generalized as how can the client efficiently perform periodical integrity verifications even without the local copy of data document. However, dynamic PoS remain to be improved in a multi-user environment, due to the requirement of cross user deduplication on the client-side. This indicates that users can skip the upload process and get the ownership of documents, as long as the uploaded document already exists in the cloud. This process can reduce storage space for the cloud, and save transmission bandwidth for users. To the best of knowledge, there is no such dynamic PoS that can support secure cross-user deduplication. There are two challenges in order to solve this problem. the authenticated structures used in dynamic PoSs, such as skip list and Merkle tree, are not convenient for deduplication. This is called as challenge structure diversity, which means the authenticated structure of a file in dynamic PoS may have some conflicts.

II. 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] extended the research of PoS, but those works did not take dynamic operations into account. Erway *et al.* [8] and later works [29] focused on the dynamic data. Among them, the scheme in [14] is the most efficient solution in practice. However, the scheme is state full, 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 Employes 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 [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 are similar 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 [18] and secure network coding. A brief survey of Homomorphic MACs and signatures could be referred in [46].

III. PROBLEM STATEMENT

A multi-user cloud storage system needs the secure client side cross user deduplication technique, which allows a user to stop the uploading process and gain the ownership of the files immediately, when other owners of the same files have uploaded them to the cloud server. As we know, none of the existing dynamic PoSs can support this technique. In most of the existing dynamic PoSs, a tag used for integrity verification is generated by the secret key of the uploaded. 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 PoS would fail. Existing dynamic PoS 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, secure authentication and private tag generation existing system cannot be extended to dynamic PoS.

IV. IMPLEMENTATION

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. 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 . 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 user encodes the local files and uploads 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 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 may not be uploaded by these users, but they pass the deduplication phase and prove that they have the Ownerships of the files. Furthermore, we extend the system by adding the concept of Certificate Authority for secure authentication.

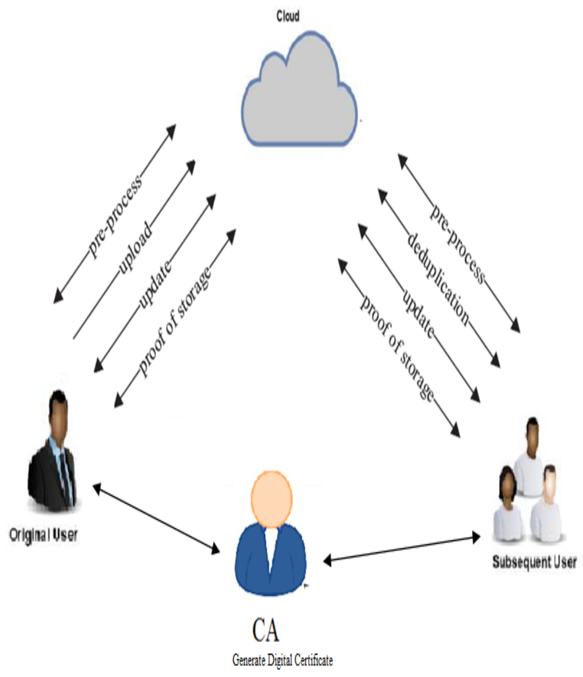

Figure: Architecture

V. IMPLEMENTATION MODULES:

1. System Construction
2. Block Generation
3. Deduplicatable Dynamic POS
4. Homomorphic Authenticated Tree

1. 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 data's 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.

2. Block Generation

In this module, we develop the Block Generation process. In the *update* phase, users may modify, insert 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.

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

4. 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 mι represents the ι-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. 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.

VI. CONCLUSION

We proposed the first realistic Deduplicatable dynamic PoS scheme which makes use of complete necessities in multi consumer cloud storage systems and proved its security within the random oracle model. The theoretical and experimental results show that the procedure is efficient, peculiarly when the file dimension and the number of the challenged blocks are large. Furthermore, we extend the system by adding the concept of Certificate Authority for secure authentication.

VII. 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. K¨upc ¨u, 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 distributedfile 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.

[26] E. Stefanov, M. van Dijk, A. Juels, and A. Oprea, "Iris: A scalable cloud file system with efficient integrity checks," in *Proc. of AC- SAC*, pp. 229–238, 2012.

[27] D. Cash, A. K¨upc¸ ¨u, and D. Wichs, "Dynamic proofs of retrievability via oblivious RAM," in *Proc. of EUROCRYPT*, pp. 279–295, 2013.

[28] M. Azraoui, K. Elkhiyaoui, R.Molva, andM. ¨Onen, "StealthGuard: Proofs of Retrievability with Hidden Watchdogs," in *Proc. Of ESORICS*, pp. 239–256, 2014.

[29] Z. Ren, L. Wang, Q. Wang, and M. Xu, "Dynamic Proofs of Retrievability for Coded Cloud Storage Systems," *IEEE Transactions on Services Computing*, vol. PP, no. 99, pp. 1–1, 2015.

[30] R. Di Pietro and A. Sorniotti, "Boosting Efficiency and Security in Proof of Ownership for Deduplication," in *Proc. of ASIACCS*, pp. 81–90, 2012.

[31] J. Xu, E.-C. Chang, and J. Zhou, "Weak leakage-resilient clientside deduplication of encrypted data in cloud storage," in *Proc. Of ASIACCS*, pp. 195–206, 2013.

[32] S. Keelveedhi, M. Bellare, and T. Ristenpart, "DupLESS: Serveraided encryption for deduplicated storage," in *Proc. of USENIX Security*, pp. 179–194, 2013.

[33] J. Li, X. Chen, M. Li, J. Li, P. Lee, and W. Lou, "Secure Deduplication with Efficient and Reliable Convergent Key Management," *IEEE Transactions on*

Parallel and Distributed Systems, vol. 25, no. 6, pp. 1615–1625, 2014.

[34] J. Li, Y. K. Li, X. Chen, P. Lee, and W. Lou, "A Hybrid Cloud Approach for Secure Authorized Deduplication," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 5, pp. 1206–1216, 2015.

[35] Q. Zheng and S. Xu, "Secure and efficient proof of storage with deduplication," in *Proc. of CODASPY*, pp. 1–12, 2012.

[36] R. Du, L. Deng, J. Chen, K. He, and M. Zheng, "Proofs of ownership and retrievability in cloud storage," in *Proc. of TrustCom*, pp. 328–335, 2014.

[37] B. Wang, B. Li, and H. Li, "Public auditing for shared data with efficient user revocation in the cloud," in *Proc. of INFOCOM*, pp. 2904–2912, 2013.

[38] B. Wang, B. Li, and H. Li, "Oruta: privacy-preserving public auditing for shared data in the cloud," *IEEE Transactions on Cloud Computing*, vol. 2, no. 1, pp. 43–56, 2014.

[39] J. Yuan and S. Yu, "Efficient public integrity checking for cloud data sharing with multi-user modification," in *Proc. of INFOCOM*, pp. 2121–2129, 2014.

[40] R. Gennaro and D.Wichs, "Fully Homomorphic Message Authenticators," in *Proc. of ASIACRYPT*, pp. 301–320, 2013.

[41] D. Boneh and D. M. Freeman, "Homomorphic Signatures for Polynomial Functions," in *Proc. of EUROCRYPT*, pp. 149–168, 2011.

[42] D. Catalano, D. Fiore, and B. Warinschi, "Homomorphic Signatures with Efficient Verification for Polynomial Functions," in *Proc. of CRYPTO*, pp. 371–389, 2014.

[43] A. Yun, J. H. Cheon, and Y. Kim, "On Homomorphic Signatures for Network Coding," *IEEE Transactions on Computers*, vol. 59, no. 9, pp. 1295–1296, 2010.

[44] C. Cheng and T. Jiang, "An Efficient Homomorphic MAC with Small Key Size for Authentication in Network Coding," *IEEE Transactions on Computers*, vol. 62, no. 10, pp. 2096–2100, 2013.

[45] J. Chen, K. He, R. Du, M. Zheng, Y. Xiang, and Q. Yuan, "Dominating Set and Network Coding-Based Routing in Wireless Mesh Networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 2, pp. 423–433, 2015.

[46] D. Catalano, "Homomorphic Signatures and Message Authentication Codes," in *Proc. of SCN*, pp. 514–519, 2014.

[47] M. Bellare, S. Keelveedhi, and T. Ristenpart, "Message-locked encryption and secure deduplication," in *Proc. of EUROCRYPT*, pp. 296–312, 2013.

[48] M. Abadi, D. Boneh, I. Mironov, A. Raghunathan, and G. Segev, "Message-locked encryption for lock-dependent messages," in *Proc. of CRYPTO*, pp. 374–391, 2013.