Dynamic Proof of Storage in Cloud for Multiple User Environments

[a] Rakshitha H.N, [b] Mrs.Shruthi Prabhakar, [c] Mrs. Rachana C.R.
[a] IV Sem, M.Sc(CS)., DOS in Computer Science.
[b] Assistant Professor, DOS in Computer Science.
[c] Associate Professor and Head, DOS in Computer Science.
PG Wing of SBRR Mahajana First Grade College, Pooja Bhagavat Memorial Mahajana Education,
Metagalli, K.R.S Road, Mysuru, India

Abstract **- Dynamic Proof of Storage (DeyPoS) is a utility authority primitive that authorizes the files integrity and systematically modernizes the files in a cloud server. A sensible multi-user cloud storage system holds the solid client-side cross-user deduplication technique, which authorizes a user to hop the uploading process and secure ownership of the file instantly and when variant owner of the same file have to upload them to the same cloud server. The concept of DeyPos and secure cross-user deduplication concurrently is introduced in this paper. Regarding the problem of structure diversity and private tag generation, we utilize a novel tool named Homomorphic Authenticated Tree.**

Keywords- Cloud Server, Deduplication, Dynamic Proof Of Storage, Ownership Proof.

I. INTRODUCTION

Storage externalization is more attractive to both industry and academia due to the advantages of low cost, high accessibility and easy sharing. Data integrity is main objective of the cloud storage. User should be convinced that the files keep within the server don't seem to be tampered. Traditional techniques for safeguarding data integrity, similar Message authentication codes (MACs) and digital signatures need users to transfer all of the files from the cloud server for authentication that incurs a heavy communication. Thus techniques don't seem to be sequestrate for cloud storage services wherever users could check the integrity frequently like each hour.

Thus researchers establish proof of storage (PoS) for checking the Integrity, while not downloading files from the cloud server users need many dynamic operations like modification, insertion, and deletion and updating to maintaining the prospective of PoS, dynamic PoS enroll authenticated structures, like Merkle tree.
In these schemes, each block of a file is attached a (cryptographic) tag that is employed for substantiating the integrity of that block. Once a proponent wants to deduce the integrity of a file, it selects some block indexes of the file, and sends them to the cloud server. Even with these challenged indexes, the cloud server returns the corresponding blocks along with their tags. The proponent checks the block integrity and index correctness.

The former can be directly bounded by cryptographic tags. How to affect the latter is that major variation between PoS and dynamic PoS. In most of the PoS schemes the block index is "encoded" into its tag, which implies the proponents will examine the block integrity and index correctness concurrently. However, dynamic PoS cannot encode the block indexes into tags, since the dynamic operations could modification several indexes of non updated blocks, that incurs unwanted computation and communication cost. As an example, there is a file consisting of one thousand blocks and a replacement block is inserted behind the second block of the file. Then, 998 block indexes of the first file are modified, which implies the user should generate and send 999 tags for this update. However, dynamic PoS remain to be upgrade in multi-user environment, due to the demand of cross-user deduplication on the client-side. This suggests that users can skip the uploading methodology and acquire the possession of files currently, as long as a result of the uploaded files exists already among the cloud server. This methodology can reduce storage space for the cloud server, and save transmission bandwidth for users.

II. LITERATURE SURVEY

I carried out extensive survey of literature and consolidated the results which are as follow:
[1] A secure and dynamic Multi-keyword ranked search scheme over encrypted cloud data. In this paper, a secure, efficient and dynamic search theme is planned, which supports not only the accurate multi-keyword stratified search however conjointly the dynamic deletion and insertion of documents. A balanced binary tree as the index is constructed and "Greedy depth-first search" algorithm is preferred to obtain higher strength than linear search. In the proposed theme, the data owner is to responsibility for generating change data and creates them to cloud server. The data owner must store the unencrypted index tree and also the data.

[2] Security and privacy in cloud computing: A Survey. One of the leading interesting challenges within the area of social calculate and social media investigation is that the supposed community analysis. An accepted barrier in multiple website analysis is that the separation of those websites. This paper aims to make proof on the existence of a mapping among identities across multiple communities providing a technique for connecting these websites.

[3] From security to Assurance in the cloud: A Survey. Cloud property at lower prices, and higher performance and without having to care about infrastructure management still, cloud occupant remain worried with the cloud's level of service and the non functional properties their applications can count on. The research community has been in the focus on the nonfunctional aspect of the cloud paradigm, among which cloud safety stands out. The analysis in this article focuses on the interface between cloud security and cloud security assurance.

[4] Proofs of ownership in Remote Storage Systems. In this paper, an idea of proof-of-ownership is set, a client can prove to a server that it has a copy of a file without sending it. This authorizes to counter attacks on file deduplication systems where the attackers get a "short summary" of the file and uses it to fool the server into thinking that the attacker owns the entire file.

[5] Hybrid provable data possession at untrusted stores in cloud computing. It shows the core problem, if an untrusted server to store client info. We will obvious knowledge possession within the model, which cut back the knowledge block access, but additionally cut back the quantity of computation on the server and consumer and server traffic. Our design and development on the PDP program is in the main supported the usage of rhombohedra and uneven coding system. It exceeds what we did in the past; the improvement has brought to the bandwidth, computation and storage system. And it applied the public (third party) verification. Finally, we additionally expect our program, it supports dynamic outsourcing of information create it a additional realistic application of cloud computing surroundings

[6] Compact proofs of retrievability. In this paper, built from BLS signatures and secure in the random oracle model, features a proof-ofretrievability protocol in which the client's query and server's response are both extremely short. This scheme allows public verifiability: anyone can act as a verifier, not just the file owner. Our second scheme, which builds on pseudorandom functions (PRFs) and is secure in the standard model, allows only private verification. It features a proof-of-retrievability protocol with an even shorter server's response than our first scheme, but the client's query is long. Both schemes rely on homomorphic properties to aggregate a proof into one small authenticator value

[7] Provable data possession at untrusted stores. It shows, two provable-secure PDP schemes that are more efficient, even when compared with schemes that achieve weaker guarantees. In particular, the over head at the server is low, as opposed to liner in the size of the data. Experiments using our implementation verify the practicality of PDP and reveal that the performance of PDP is bounded by disk input output and not cryptographic computation.

[8] Scalable and Efficient Provable Data Possession. It shows how to frequently, efficiently and securely verify that a storage server is faithfully storing its client's data. The storage server is assumed to be untrusted in terms of both security and reliability. The problem is exacerbated by the client being a small computing device with limited resources. Prior work has addressed this problem using either public key cryptography or requiring the client to outsource its data in encrypted form .It efficiently supports operations like block modification, deletion and append

[9] Proof of storage from homomorphic identification protocols. In this paper proofs of storage are interactive protocols helps to client to verify that a server trusty stores a file. Previous work has proofs of storage can be constructed from any homomorphic linear authenticator (HLA). Roughly speaking, are signature/message authentication schemes where 'tags' on multiple message can be homomorphically merge to yield a 'tag' on any linear combination of these messages. We gave a frame work for building public-key HLAs from any identification protocol satisfying certain homomorphic properties. We then show how to turn any public-key HLA into publicly-verifiable PoS with communication complexity independent of the file length and supporting an unbounded number of verifications.

[10] Dynamic proof of Retrievability for coded cloud storage systems. A new dynamic proof of retrievability idea for coded cloud storage systems. Network coding and erasure codes are affect to encode data blocks to achieve within-server and cross-server data redundancy.

[11] A dynamic proof of retrievability scheme with 0(log n) complexity. It shows cloud storage conducts security concerns one major concern is about the data probity. We extend the static PoR idea to dynamic framework. We present a new authentication data structure called Merkle b+ tree. Compared to the existing dynamic PoR idea, our worst case communication complexity is 0(log n) instead of 0(n).

[12] Practical Dynamic proofs of retrievability. In this paper, a dynamic PoR scheme with constant client storage whose bandwidth cost is comparable to a Merkle hash tree, thus being very practical is proposed. The structure out performs the constructions of Stefanov et al. and Cash et al. both in theory and in practice. Specifically for n outsourced blocks of bits each writing a block requires +0(log n) server computation. Audits are also very efficient.

III. EXISTING SYSTEM

In most of the existing dynamic PoS, a tag used for integrity verification is generated by the secret key of the up loader. Thus, other owner 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. This time, the dynamic PoS would fail.

Halevi et al. 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 proposed another proof of ownership scheme which improves the efficiency.

Xu et al. 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.

The cloud server and users do not fully trust each other. A malicious user may cheat the cloud server by claiming that it has a certain file, but it actually does not have it or only possesses parts of the file. A malicious cloud server may try to convince users that it faithfully stores files and updates them, whereas the files are damaged or not up-to-date.

Existing dynamic PoS cannot be extended to the multi-user environment is a disadvantage.

Other drawback is 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 cannot support deduplication due to structure diversity and private tag generation is a disadvantage of existing system.

IV. PROPOSED SYSTEM

To institute a primitive called deduplication dynamic proof of storage, which clarify the structure diversity and private tag generation challenge?
In divergence to the live authenticated construction, like skip list and Merkle tree, we planned an authenticated construction known as Homomorphic Authenticated Tree, to diminish the communication cost in both the proof of storage and deduplication both stage along with computation cost.

HAT can support honesty verification, dynamic operations and cross-user deduplication with good stability.

We suggest and implement a systematic construction of deduplicatable dynamic PoS. it helps inexhaustible number of verification and update operations.

Some advantages are it is an efficient authenticated structure.
It supports secure cross-user deduplication.
Performs are better especially when the file size and the number of the challenged blocks are large.

V. SYSTEM ARCHITECTURE

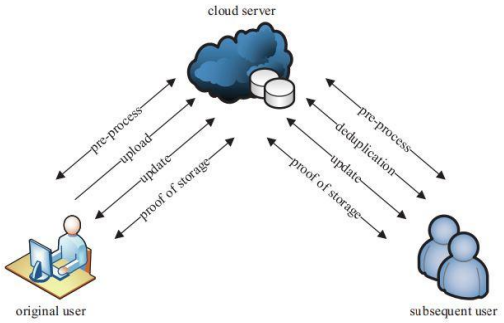


Fig.1: The system model of deduplication Dynamic PoS
In our architecture, we consider two entities like cloud server and user and it contain five phases Such as pre-process, upload, update, deduplication and proof of storage. Upload phase is a unidirectional process from the original user. Users have to prove their proof of ownership in multi-user environment.

VI MODULES

Our system construction model considers two types of entities: the cloud server and users. For each file Original user 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 checks login authentication of users and then it gives permission for deduplication process for authenticated users and user's data are stored in blocks.

In block generation model users can update the files only if they have the ownership of the files, which means that the users should upload the files in the upload phase or pass the verification in the Deduplication phase.

There are five phases in dedulicatable dynamic PoS system: pre-process, upload, deduplication, update and proof of storage.

In the pre-process phase, users intend to upload their 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 deduplication phase.

In the deduplictaion 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.

Update the cloud server has to reserve the original file and the authenticated structure if there exit other owners, and record the updated part of the authenticated structure, Since each update is only "attached" to the original file and authenticated structure.

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.

Homomorphic Authenticated tree reduce the communication cost in both the proof of storage phase and the deduplication phase with similar computation cost.

VII. IMPLEMENTATION

The project has been implemented with the following software tool JAVA/J2EE, Netbeans 7.2.1, MYSQL.

1. Building Blocks

We deploy the following tools as our building blocks:

1) Collision-resistant hash functions: A hash function H: $\{0, 1\}^* \rightarrow \{0, 1\}^*$ is collision-resistant if the probability of finding two different values x and y that satisfy H(x) = H(y) is negligible.

2) Deterministic symmetric encryption: The encryption algorithm takes a key k and a plaintext m as input, and outputs the cipher text. We use the notation Enc k(m) to denote the encryption algorithm.

3) Hash-based message authentication codes: A hash-based message authentication code HMAC: $\{0, 1\}^* \times \{0, 1\}^* \rightarrow \{0, 1\}^*$ is a deterministic function that takes a key k and an input x, and outputs a value y. We define HMAC k(x) def = HMAC (k, x).

4) Pseudorandom functions: A pseudorandom function f: $\{0, 1\}^* \times \{0, 1\}^* \rightarrow \{0, 1\}^*$ is a deterministic function that takes a key k and a value x, and outputs a value y that is indistinguishable from a truly random function of the same input x. We Define $f_k(x)$ def= f (k, x).

5) Pseudorandom permutations: A pseudorandom permutation $\pi : \{0, 1\}^* \times [1, n] \rightarrow [1, n]$ is a deterministic function that takes a key k and an integer x where $1 \leq x \leq n$, and outputs a value y where$1 \leq y \leq n$ that is indistinguishable from a truly random permutation of the same input x. We define π k(x) def= π (k, x).

The leaf node tag generation algorithm:

1: procedure LEAFTAG (α_s , k_c , α_c , c_i , l_i, l_{il}, v_{il})
2: $\tau_i \leftarrow \alpha_s c_i$
3: $l_{ii} \leftarrow f_{kc}$ ($i_i || l_{il} || v_{il}$) + $\alpha_c \tau_l$
4: return τ_l, $t_{i\,l}$

The non-leaf node tag generation algorithm:

1: procedure NONLEAFTAG (k_c, i, l_i, v_i)
2: $\tau_{2i} \leftarrow t_{2i} - f_{kc}$ ($2i || l_{2i} || v_{2i}$)
3: $\tau_{2i+1} \leftarrow t_{2i+1} - f_{kc}$ ($2i + 1 || l_{2i+} || v_{2i+1}$)
4: return $t_i \leftarrow f_{kc}$ ($i || l_i || v_i$) + τ_{2i} + τ_{2i+1}

6) Key derivation functions: A key derivation function KDF: $\{0, 1\}^* \times \{0, 1\}^* \rightarrow \{0, 1\}^*$ is a deterministic function that can derive a secret key from two secret values.

2. Pre-process Phase

In the pre-process phase, a user runs the initialization algorithm (id, e) ← Init (1, F) which computes: e ← H (F), id ← H(e). Then, the user announces that it has a certain file via id. If the file does not exist, the user goes into the upload phase. Otherwise, the user goes into the deduplication phase.

3. Upload Phase

Let us the file F = ($m_1 . . . m_n$). The user first invokes the encoding algorithm (C, T) ← Encode (e, F)

4. Deduplication Phase

If a file announced by a user in the pre-process phase exists in the cloud server, the user goes into the deduplication phase and runs the deduplication protocol res ∈ {0, 1} ← Deduplicate (U(e, F), S(T)).

5. Update Phase

A user can arbitrarily update the file, such as modify a block, insert a batch of blocks, and delete some blocks, by invoking the update protocol res ∈ {he*, (C*, T *)i,⊥ } ← Update (U(e, ι, m, OP), S(C, T)). After all operations are finished, the user uploads the updated blocks of the file and the updated nodes of the HAT to the cloud server. Then, the user computes the updated metadata e^* and verifies the updated blocks via the checking protocol.

6. Proof of Storage Phase

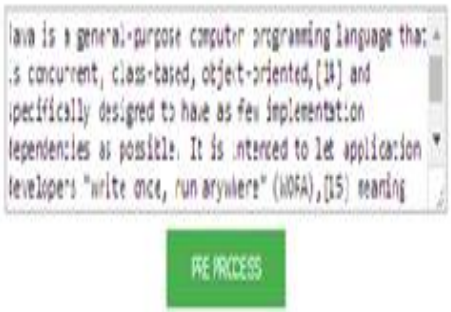
Every time, users can go into the proof of storage phase if they have the ownerships of the files. The users and the cloud server run the checking protocol res ∈ {0, 1} ← Check (S(C, T), U (e)) interactively to check the file integrity in the cloud server

VIII.EXPERIMENTAL RESULTS:

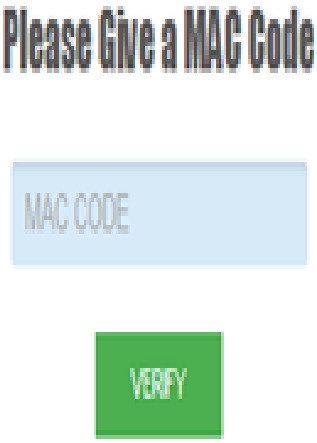
Registration Form:

User login

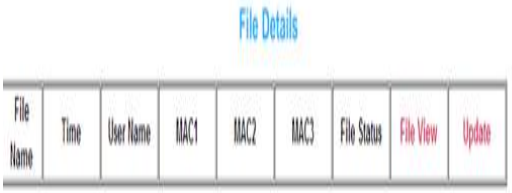
File uploading



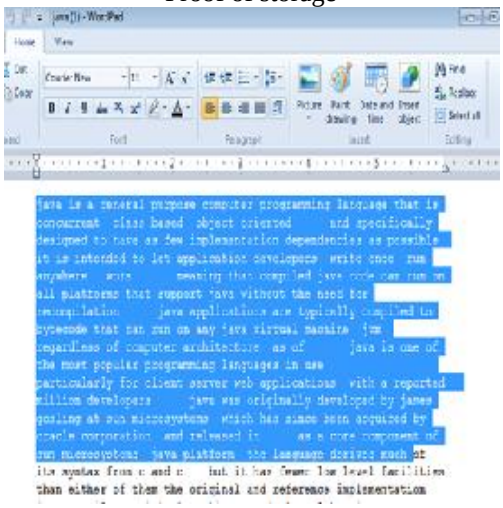
Duplication checking



File details



Proof of storage



Volume 5, Issue 21 Published by, www.ijert.org 5

VIIII. CONCLUSION

A novel tool known as HAT is used. The first practical deduplicatable dynamic PoS scheme called DeyPoS is implemented effectively. The abstract and experimental result shows that DeyPos implementation is efficient, especially once the file size and the variety of the challenged blocks area unit is massive.

REFERENCES

[1] Kun He, Jing Chen, Ruiying Du, Qianhong Wu, Guoliang Xue, and Xiang Zhang,"De-duplicatable dynamic proof of storage for multi-user environments," IEEE Transactions. 2016

[2] Z.Xia. 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: A Survey," 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] 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.

[6] Narn-Yam Lee and Yun-kunachang. "Hybrid provable data possession at untrusted stores in clod computing," In Proc, of CCS, pp.598-609, 2007.

[7] H.Shacham and B.Waters,"Compact proofs of Retrievability," journal of Cryptology, vol.26, no.3, pp. 442-483, 2013.

[8] 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.

[9] 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.

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

[11] Z.Ren, L.Wang, Q.Wang and M.Xu,"Dynamic proofs of Retrievability for coded cloud storage systems," IEEE Transaction son Services Computing vol. pp, no. 99, pp. 1-1, 2015.

[12] Z.Mo, Y.Zhou, and S.chen,"A dynamic proof of retrievability (PoR) scheme with 0(log n) complexity," in Proc.of ICC, pp. 912-916, 2012.

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