International Journal of Engineering Research & Management Technology
ISSN: 2348-4039
September-October-2020 Volume 7, Issue-5
Email: editor@ijermt.org www.ijermt.org

DEDUPLICATION OF SECURE DATA IN CLOUD STORAGE

Kamlesh Kumar,
Research Scholar, Dept. of CSE,
The Glocal University, Mirzapur Pole, Saharanpur (UP)

Dr. Manoj Kumar,
Associate Professor, Dept. of CSE,
The Glocal University, Mirzapur Pole, Saharanpur (UP)

Abstract
In today's digitally evolving world, the very thing that is of utmost importance is the security of data. Cloud Computing has emerged as a popular and effective tool to manage data for administrations. Each day, around 2.5 quintillion bytes of data is generated on internet and to store this large amount of data, we need servers which can deduplicate data efficiently so as to avoid wastage of storage thus minimizing expenses. In this paper, we will be looking onto various techniques and methods to achieve this deduplication. The results depict that redundant data is always mapped onto same hash code and thus it does not get uploaded on the cloud servers thus ensuring successful deduplication. It saves storage as well as saves bandwidth by eliminating duplicate data. In this project, data gets stored in the cloud server named drivehq and numerous efforts have been taken to ensure complete data access. With effective deduplication, ensuring data confidentiality is also very important thus data is always stored in the cloud in an encrypted format. It is achieved with the help of Advanced Encryption Standard (AES) algorithm.

Key Words: *Cloud Computing, deduplication.*

Introduction

One of the best strategies to control the rapid expansion of data is through deduplication. Many storage providers use data deduplication because it allows them to save a lot of storage space, which lowers prices. The idea of deduplication is to store only one copy of each piece of data. If a user needs to save repeated data that the cloud storage provider has previously acquired, the storage provider simply makes a pointer link to that data rather than storing a new duplicate of the data. A technique called file level deduplication stores each file in only one copy. Each file is broken into blocks using the block level deduplication approach, and each block is then saved in a single copy. By comparing the hash value of the current file with the hash value of previously stored files or blocks, indistinguishable blocks of files are detected. A block of data's newness to the system is determined by the cuckoo filters.

In-line deduplication and post-process deduplication are two methods for achieving deduplication in cloud storage. Data is initially stored and then processed for deduplication checks as part of the post-process deduplication. The benefit of this method is that the preprocessing time for hash calculations and computation is not taken into account when storing the data, but the drawback is that there is a risk of storing duplicate data in temporary servers that must be removed once the deduplication check is complete, taking up unnecessary storage space on a system that is already close to capacity.

Deduplication hash calculations can be carried out in-line while data is being stored on the servers. After the data has been divided into manageable chunks, a high-security cryptographic hash (SHA) is generated for each one.

Then, an index is created using these hashes. The present chunk is regarded as duplicate if the hash lookup table in the index returns a duplicate result. Instead of storing an entirely new block when a similar block is found that has already been stored, simply a pointer reference to the old block is stored. Convergent encryption, where the encryption key is extracted from the plain text itself so that similar data will have similar hash values, will be utilized for this form of deduplication. However, private key encryption can still be used to get around this convergent encryption. Convergent encryption offers a practical way to achieve deduplication and guarantee data confidentiality. A similar file will always encrypt to the same cipher text and produce the same plain text throughout the retrieval process thanks to convergent encryption, which generates encryption keys consistently from the data content. A convergent key is used to encrypt or decrypt a data copy, and it is generated by computing the cryptographic hash value of the data copy's content. Users keep the keys after data encryption and key generation and submit the encrypted text to the cloud. Due to the determinism of encryption, identical data copies will produce the same convergent key and cipher text. This makes it possible for the server to deduplicate the encrypted text. Only the respective data owners with their convergent keys may decrypt the encryption text. Using a hash algorithm, a file is hashed in this convergent encryption to produce a hash value that may be used as the encryption key for the file. When utilizing the AES technique to encrypt files, the output hash code or fingerprint can be utilized as the key.

Processes including hashing, key management, and comparison are used in deduplication. Tiger hash, which takes less work than other hashing methods, will be the hashing algorithm employed here. The DHT will keep the hashing keys for use in other dynamic auditing and deduplication operations. In order to prevent data duplication in cloud storage, the inline hashing approach will be applied in this case. If any new data is uploaded to the cloud storage by any user, the new hash value will be compared with the existing hash value. Tiger is a quick and powerful hash function that is made to run quickly on contemporary devices, especially the most cutting-edge ones, while remaining competitive with other suggested hash functions on 32-bit processors. Compared to MD5 or SHA-1, tiger hash is a more structured design. In order to create the tiger hash function, the Merkle-Damgard paradigm is used universally. Tiger is a one-way compression algorithm that makes use of 64-bit architecture. The hash value for a given data block is produced using the Tiger hash function. In the tiger hashing method, 512 bits of input data provide a 192-bit hash value. Tiger-160 creates 160-bit hashes with SHA-1's execution time lowered while also offering greater accuracy. Collision resistance is a feature of hash values that prevents attackers from accessing and altering them. The output of the Tiger hash function is of constant size, and the generated hash values do not collide, allowing the integrity of the data to be checked at any time and from any location. Higher data integrity is achieved as a result.

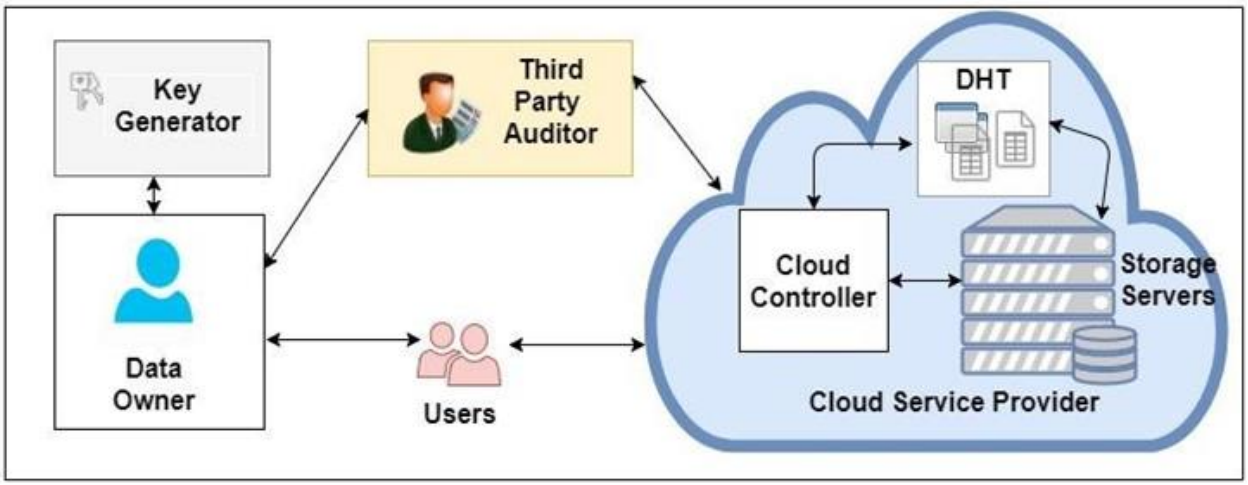

Figure-1.1 : The Ldap Protocol Architecture

The CSP also has a Cloud Controller (CC) that transports the non-duplicated data from the DO to the cloud storage servers after checking for deduplication entries. The storage index table is constructed for the data kept by the storage manager, and the index for the deduplication data is updated with changes to the address pointers. As the response to the request is generated and sent to the requester, the challenge question generator in the cloud controller is in charge of producing queries according on the scheduling strategy.The module that facilitates the development of the challenge question's random selection based on the agreed-upon audit scheduling strategy is known as the challenge query generator.

The TPA Verifier determines if the received proof is legitimate or not, and if any improper behavior is discovered, the audit report is forwarded to the DO for further action. When the data is received, the CSP Controller looks for any deduplication data entries. If any blocks have duplicate entries, the duplicated blocks should be discarded, and the unique data blocks should be stored instead. A pointer to the storage address of the original data block will be included in the replicated blocks. The file server and the server check for deduplication entry and store the data in the cloud. The storage manager generates an index for each file with a file id, hashes the exact location address for the blocks of the file document to be stored in the storage, and stores the address for the set of documents in the exact storage. The auditor will receive the produced tag and check the data tag with the server later. The data is now received by the server from the client and is encrypted before being stored in the cloud into the table. When the auditor requests any proof from the server, the server immediately generates the tag and sends it to the auditor for verification

Dynamic Auditing for Data Integrity

When generating a request inquiry using algorithm 2, the TPA checks the auditing result it receives from the CSP as a response. Prior to comparing the server's response for matches of the hash of the challenged blocks, the TPA first computes the hash values for all the challenged data blocks using the Merkle hash tree and computes the root challenge hash.

Algorithm 2:Third Party Auditor

Input : Metadata *Ti*, Response *R*

Output: query result

Begin

1. After receiving the Tag Ti from the user, create an index search table to contain the tag values.

$Ti = \{ Fid , Uid , Hv , Hc , Ts , Fv , T , k \}$

2. The TPA submits an auditing request to the CSP for a collection of random blocks in accordance with the scheduling rules with audit period T of the data (Table 1.1).

1. Create distinct request queries with challenge ID, file ID, block number, and verification key for n random blocks.

$Chal= \{Cid , Fid , bi , Ts , k\}$ for $1 \leq i \leq n$

4. Send the CSP the query for a group of blocks that have been challenged.

$S.Chal \leftarrow \{i\}\ i \in N \mid 1 \leq i \leq N$

5. Compute Merkle Hash Tree using the hash values sent by the CSP for the relevant challenged blocks and extract the hash value from the TPA for the challenged blocks after receiving the CSP's response.

6. To verify the accuracy of the data, compare the root hash value.

$h_{root}\ (S.Chal) == h_{root}\ (R.Resp)$

7. If the results are equal, the proof is successful and the output is true; otherwise, if it fails, the data blocks are incorrect and the user will be informed of the corrupted data blocks.

Deduplication

Data is expanding at an absurdly rapid rate. Deduplication is a method for storing data once and using it as many times as necessary. The pieces' hash digest is calculated by the DO. These hash values are known as a fingerprint or digest. The cuckoo filter stores the fingerprint, and different storage server nodes get the encrypted data blocks.

Deduplicated data is removed by the server, which only keeps the original data. The cuckoo filters utilized allow for a quick query validation and quick calculation of the result, allowing for high throughput. Based on the index values, the filters allow users to determine if a specific data block is present in the system or not. Each node has a hash value, a relative index, data, and the time the tree was created appended. The sensitivity-based scheduling variables serve as the foundation for the relative index. Cuckoo filters significantly improve the performance of database query operations by lowering the cost of disk look-ups for missing data. Faster set membership tests are supported by the use of cuckoo filters, which scale exponentially to contain the data. Each hash block's similarity identifier is calculated and recorded in the index file for each input file. The common blocks within the input can be effectively found using the similarity identifier.

Algorithm 3:Cloud Service Provider

Input : Encrypted data

Output: storage index, response

Begin

1. Before saving the data, the cloud controller runs a deduplication check by using cuckoo filters to compare the hash values of the blocks stored on the server with the hash blocks that were received from the data owner.

For each Fid in storage Index

If H(f) != Fid.H **then**

return True

Else

Check the blocks connected to Fid to obtain the address.

The reference pointer is used to replace the redundant block and point to the saved block.

return list()

End If

End for

2. If the block matches, the data is erased and a link to that block is stored.

1. Keep the specifically defined block.

4. The storage manager updates the storage index table to make it easier to search for and retrieve data or to generate a response.

5. Extract for each query received Cid , Fid ,bi , k

6. Calculate MAC code $MAC=H(CTi , k)$

7. Create response R.Resp={Cid , MAC, bi }

8. Send the response to TPA.

End

Reducing storage system overload and optimizing storage utilization for the suggested online cloud storage. By avoiding making duplicate copies of existing files that are being stored in the cloud, storage space may be kept efficiently. Before a file is stored on a server, it is encrypted in order to increase the security of the files that third-party users maintain. The secure and safe key maintenance protects the encrypted file. Without the correct file key, the third-party members cannot decrypt the encrypted file. The security is upheld during the download process by the two key level authentications. Overall, the planned work focuses more on security while also conserving cloud storage capacity.

A lookup service of a certain key value pair from the widely dispersed data at the storage servers is provided by the Improved Distributed Hash Table when used with the put/get interface. The improved distributed hash table (IDHT) builds a data structure from the file's hash values and the container that houses the remaining file blocks. Each block has a pointer that leads to the following block that is kept in the bucket. The containers are regarded as buckets that contain both the object id and the data. On the remote data storage server dispersed over many cloud data centers, the updating procedure can be carried out. Based on the hash value, the IDHT retrieves the

data that has to be updated. Utilizing the hash function will prevent collisions and allow for effective dynamic auditing. When a request for dynamic data updating is made, the block that the data owner wants to update at the CSP must be chosen based on the file data kept there. To execute an integrity check on the stored data after a block in the file has been updated, the entire key and value combination must be altered. This important update is completed. When there are fewer buckets available to retain the key, only one location of the (key, value) pair is updated rather than the entire pair. In order to verify the data integrity of the data saved at the CSP, it is also necessary to alter the metadata that is recorded at the TPA along with the key value. Simply updating the TPA table based on the file and block number will accomplish this.

Based on the request from TPA, the following step is to obtain the hash value from CSP. The effectiveness of the CSP in getting the hash data has a significant impact on the system's performance. The MHT (Merkle hash tree) is used by this system to retrieve hashes from the CSP. For a single auditing purpose, just the hash value of the data block can be acquired using MHT, not the entire data block. This lowers the cost of computation and communication for the CSP process.

The load factor for both a successful and unsuccessful search for a particular block in the hash list is assessed in the improved distributed hash table.

$$Hash\,List\,Load\,Factor = \frac{number\,of\,elements\,K\,in\,the\,list}{Number\,of\,physcial\,elements\,n\,allocated\,for\,the\,list}$$

Where ‘n’ is represented in percentage

$$\propto = \frac{K}{n} * 100$$

K represents number of blocks in the table

n represents the table capacity

The predicted average length of a successful search [ALOS] for a block in the hash list when employing the double hashing method is as follows

$$E[ALOS] = \begin{cases} -\frac{1}{\propto} ln(1-\propto) & for\,successful\,search \\ \frac{1}{1-\propto} & for\,unsucessful\,search \end{cases}$$

In hash tables, hash collisions are resolved using double hashing. A vacant space for the data blocks to be put in the CSP is probed using a double hash. When the proper table size and hash are supplied, double hashing for any load will find an empty slot. As a result, random hashing looks to have the best search cost. All of the components are redistributed to a new, huge table when the load exceeds the threshold of load. When compared to the linear

search method shown in Table, the average time in milliseconds required for probing an array of n is swift and ideal.

Table 1.1

Comparison of Double Hashing and Linear Hash in Milliseconds

Array Size(n)	Double Hashing (ms)	Linear HashSearch(ms)
10,000,000	2200	2500
100,000,000	2500	3000
1,000,000,000	3000	3200

Where 'n' stands for the search of n array places, the best case for finding a free position to store the data is O(1) and the worst case is O(n). The load factor and the number n determine the typical situation for a successful and unsuccessful search, respectively. The double hashing technique can gather statistics about the elements in an array and relate them using information about the distribution that was previously known. Therefore, double hashing is the best option for massive storage.

Complexity Analysis

Communication costs, computation costs, and storage costs are taken into account when evaluating the LDAP system. The time required creating the hash tables and the time required to validate the information are taken into account when determining the computation cost. The time it takes for the data to be sent between the CSP and TPA and for the TPA to communicate the outcome to the data owner will be measured as part of the communication cost. In terms of transmission costs, processing resources, and storage space, data storage auditing is a service that demands a lot of resources. This section provides a quick analysis of various performance indicators.

Computation Complexity

When adopting a third-party auditing paradigm, the computation cost of the auditing system consists of calculation costs on DO during system initialization, computation costs on both CSP and TPA for each challenge-response auditing inquiry, and computation costs on both CSP and TPA for deduplication queries.

Cost of computation for DO - DO are solely involved in system initialization during third-party auditing. As a result, the cost of computing on DO is incurred during file preprocessing, key generation, and metadata computation.

Since TPA's computational capability is less impressive than CSP's, it has a higher computational cost. According to numerous earlier studies, by moving the calculation load from TPA to CSP, the computation cost on TPA for each auditing query can be decreased. However, using the hybrid model that combines MHT and IDHT will lessen the computing required for the operation, just like with the LDAP protocol.

Cost of Computing on CSP - For RSA-based homomorphic algorithms, the cost of computing the data's exponentiation accounts for the majority of the computing expenses on the server. To lower the exponentiation computation cost, the DO first divides the data into multiple blocks and schedules batch processing for a portion of the sampled data blocks. This allows the CSP to only sum the linear combination of all the sampling data blocks whose size is equal to one data block. The use of better distributed hash tables and hash operations for data localization and deduplication verification helps to further cut down on computation costs.

Communication Cost

The main communication cost, which should be taken into account as the communication cost between CSP and TPA during each challenge-response auditing inquiry, is the cost of communication between CSP and TPA because Data Owners are not involved in each auditing query in the third-party auditing protocols. The cost of communication is being drastically reduced through the use of hash code-based tagging. Scheduled Sampling

audits and Batch audits are the two main methods utilized to further reduce communication costs. Based on the information transferred between DO, CSP, and TPA, this communication expense is calculated. Along with these metrics, challenge-response Message Passing Information, packet loss frequency, connection error rate, MPI transfer bit/byte speed, and MPI transfer delay must also be taken into account when calculating communication cost.

The usage of MHT and DHT for this scheme's functioning will result in comparatively cheap communication costs. Due to the volume of data that is transferred with each request from TPA and CSP, other schemes like DPDP-based skip lists, DAP, and IHT would incur higher communication costs. Additionally, IHT requires more metadata queries than this approach, which raises the system's communication expense.

Storage Overhead

The metadata (e.g., MAC, signature, tag, etc.), which is taken into account in the current data storage auditing protocols, plays a significant role during the auditing operation and also contributes the majority of the storage overhead. In the data storage auditing protocol, concise metadata are desired in order to reduce the storage overhead generated by metadata. However, the TPA must keep track of the audit table for the entire file, which is minimal and contains only the tags and associated information like block id, version, and date. The cost of storage mostly depends on the storage servers where the data owner stores all of his data. The Improved Hash Table data structure must be used by the server to retain all data files along with relevant addresses and file auditing information. However, since deduplication check filters have been added, the ability to quickly determine if a block of data is fresh to the system or not allows for a higher reduction in storage costs.

Table-1.2

Comparison of complexity analysis of auditing protocols

Scheme	Server computa-tion	Client computa-tion	Commun-ication	Server storage	Client Storage	Dynamic Operation	Dedupl-ication
PDP	O(n)	O(n)	O(n)	O(n)	O(n)	X	X
POR	O(t)	O(n)	O(n)	O(n)	O(n)	X	X
DPDP	O(log n)	O(log n)	O(log n)	O(n)	O(log n)	√	X
IHT	O(t+cs)	O(t+s)	O(s)	O(s)	O(Mx)	√	X
MHT	O(log n)	O(log n)	O(log n)	O(n)	O(1)	√	X
LDAP	O(n)	O(1)	O(c)	O(n)	O(1)	√	√

In Table, where 'Mx' stands for the random samples of message M, the complexity analysis of various integrity auditing methods is analyzed and contrasted with the LDAP protocol. 'n' stands for the file size, 't' for the tag size,'s' for the number of sectors, and 'c' for the quantity of challenged index blocks. The amount of time spent processing a proof or performing a dynamic update action is the computation cost at the server. Dynamic operations are not supported by PDP or POR systems. The data owner pre-processes the data on an almost continual basis. The auditing protocol determines the cost for encryption to protect privacy and tag creation to verify validity. As the server proofs are combined into a single proof to execute the batch verification, the server's computation costs are lowered.

The quantity of data exchanged between CSP and TPA is known as the communication cost, which is correlated with the file size and challenged blocks. The auditor must download all of the data from the server and is limited in how many verifications they can perform under the MAC-based protocols. Because the tag size is smaller than the data, using aggregate signatures greatly lowers the cost of communication. Low communication costs are incurred by homomorphic tags. Random sampling and batch auditing greatly reduce the communication costs associated with challenge and verification.

According to the computation costs for file uploads, a 1 MB block can be uploaded in just 1000 ms. The time required to upload and download a file is depicted in Figure 1.2. The system needs only 22 ms to get a 1 MB block, including time for decryption. The challenge and CSP's response determine how quickly a response is sent. The process executing at the CSP is what determines the random memory update rate and mean hit time. Additionally, memory bit/byte depend on how much the entity' primary memory is used. The cost of storage for DO, TPA, and CSP must be determined separately, and the total storage cost is determined by adding the three values.

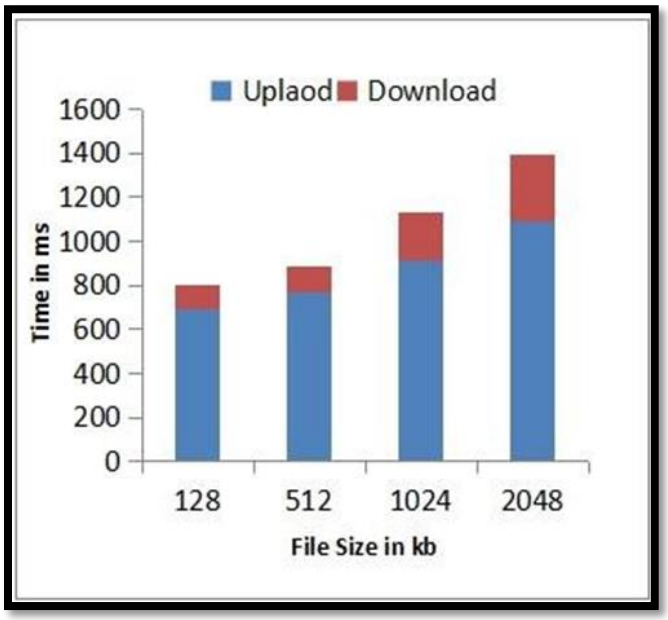

Figure-1.2: Computation Cost for File Upload and Retrieval

The usage of MHT and DHT for this scheme's functioning will result in comparatively cheap communication costs. Due to the volume of data being transferred for each request from TPA and CSP data, other systems like DPDP-based skip lists DAP, and IHT would have higher communication costs.

Additionally, IHT requires more metadata queries than this approach, which raises the system's communication expense. Sampling with batch auditing further lowers the cost of communication. At the DO, the cost of computing is minimal. Based on the amount of data transferred between CSP and TPA, this communication expense is calculated. The cost of communication is calculated based on the number of packets transmitted and received as well as the response time. The Figure displays the communication cost for each file.

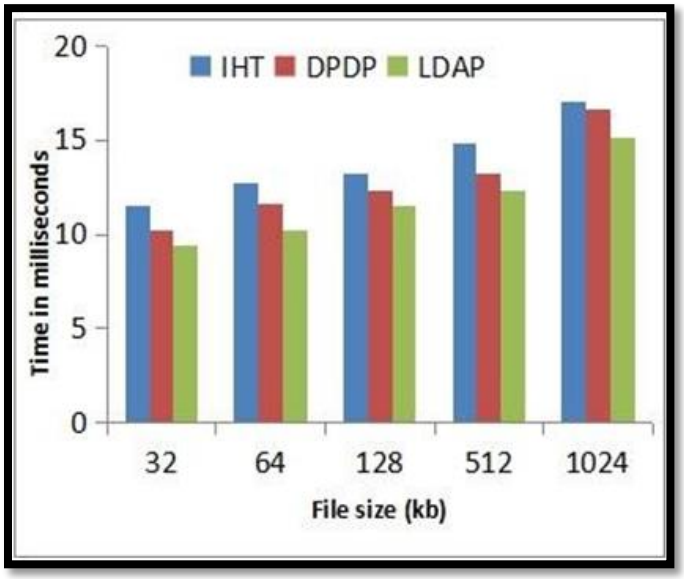

Figure-1.3: Communication Cost Between CSP and TPA

Figure tag creation and tag checking throughput demonstrates that the LDAP protocol is 50% quicker than the state-of-the-art alternatives.

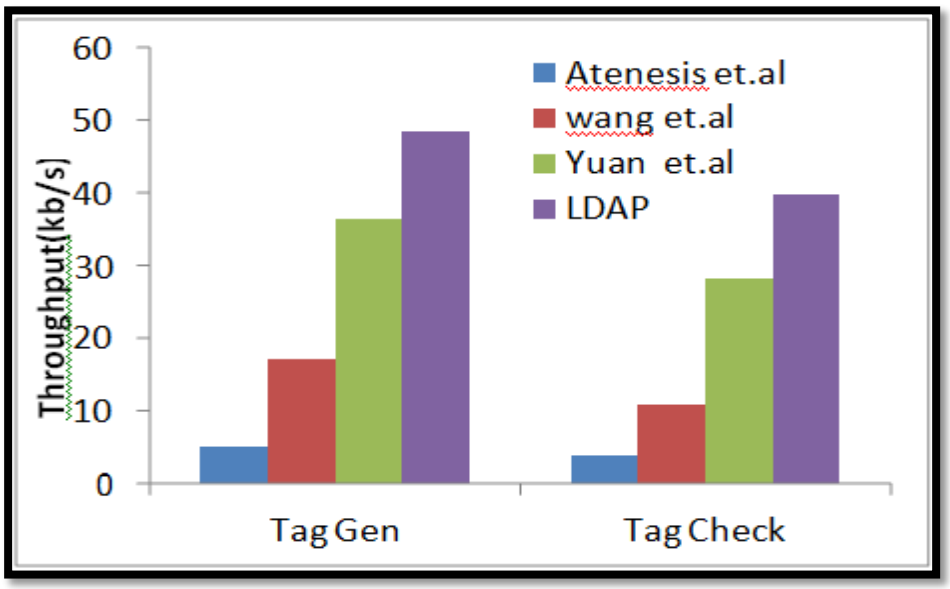

Figure-1.4: Throughput of TAG Processing

In order to prevent data duplication in cloud storage, the inline hashing approach will be applied in this case. If any new data is uploaded to the cloud storage by any user, the new hash value will be compared with the existing hash value. The operations performed at the DO side for tag generation and the activities at the TPA for proof checking are added together to determine the throughput of tag generation and tag checking. Given the servers' immense capability, CSP activities are disregarded. When compared to state-of-the-art techniques using group exponentiation, pairing, and multiplication operations, the throughput is high due to the usage of lightweight hash functions and symmetric encryptions.

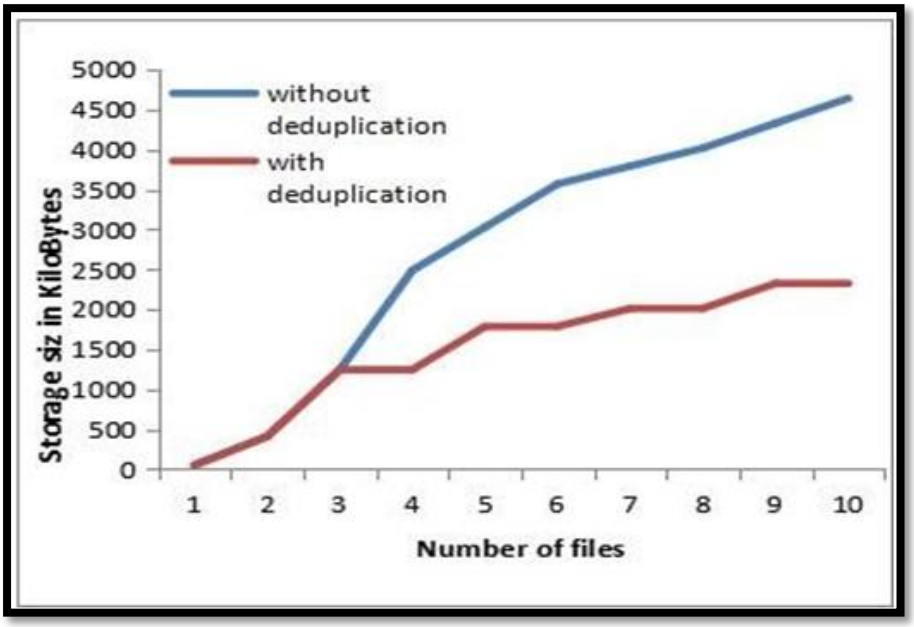

Figure-1.5: Storage Cost at the CSP

Deduplication requires processes like hashing, key management, and comparison that cost storage space. Tiger hash, which takes less work than other well-known hashing algorithms, will be the hashing algorithm employed here. The IDHT will keep the hashing keys for use in other dynamic auditing and deduplication operations. When deduplication is used, the storage cost at the CSP in Figure is cut in half.

Conclusion

Large amount of data is gathered from internet on a daily basis and this data needs to be secured from unauthorized users, criminals of cyber world. Thus encryption is necessary. This paper discusses about Advanced Encryption Standard which encrypts data before uploading it in the cloud. If duplicate data is allowed to be uploaded on the cloud on a regular basis, cloud storage will be filled with unnecessary data which need not be present there thus killing our storage and resulting in less bandwidth and bad client service. To tackle this, this paper talks about deduplication which makes use of hashing algorithm to deduplicate data. Thus this project is successful in performing secure data deduplication in cloud storage at block level which optimizes storage space and security of data. Future enhancements include production of a system which can handle large amounts of data generated everyday on cloud.

References

- Abdullah, K.A. & Al-Jafari, Mohamed. (2011). The effect of Sarbanes-Oxley Act (SOX) on corporate value and performance. European Journal of Economics, Finance and Administrative Sciences. 42-55.
- Drago, Idilio & Mellia, Marco & Munafo, Maurizio & Sperotto, Anna & Sadre, Ramin & Pras, Aiko. (2012). Inside Dropbox: Understanding personal cloud storage services. Proceedings of the ACM SIGCOMM Internet Measurement Conference, IMC. 481-494.
- Persico, Valerio& Montieri, Antonio & Pescapè,Antonio. (2016). On the Network Performance of Amazon S3 Cloud-Storage Service. 113-118.
- N. Jeber, Jalal. (2019). The Future of Cloud Computing Google Drive. 10.13140/RG.2.2.26342.06724
- Burramukku, Tirapathi & Ramya, U. & Sekhar, M.V.P.. (2016). A comparative study on data deduplication techniques in cloud storage. 8. 18521-18530
- Ku, Chan-I & Luo, Guo-Heng & Chang, Che-Pin & Yuan, Shyan-Ming. (2013). File Deduplication with Cloud Storage File System. 280-287

- N. Baracaldo, E. Androulaki, J. Glider, A. Sorniotti, “Reconciling end-to-end confidentiality and data reduction in cloud storage,” Proc. ACM Workshop on Cloud Computing Security, pp. 21–32, 2014
- Rahumed,H. C.H. Chen, Y. Tang, P. P. C. Lee, J. C. S. Lui, “A secure cloud backup system with assured deletion and version control,” Proc. International Workshop on Security in Cloud Computing, 2011
- Yang, Xue & Lu, Rongxing & Shao, Jun & Tang, Xiaohu & Ghorbani, Ali. (2018). Achieving Efficient and PrivacyPreserving Multi-Domain Big Data Deduplication in Cloud. IEEE Transactions on Services Computing. PP. 1- 1. 1
- Burramukku, Tirapathi & Rao, M.V.P.. (2017). Data deduplication in cloud storage using dynamic perfect hash functions. Journal of Advanced Research in Dynamical and Control Systems. 9. 2121-2132.
- Siddiqui, Shadab&Darbari, Manuj&Yagyasen,Diwakar. (2019). A Comprehensive Study of Challenges and Issues in Cloud Computing: Methods and Protocols.