Filter Based Data Deduplication in Cloud Storage using Dynamic Perfect Hash Functions

B. Tirapathi Reddy
ANU College of Engineering
Acharya Nagarjuna University
Andhra Pradesh, India.

M. V. P. Chandra Sekhara Rao
Department of CSE
RVR&JC College of Engineering, Guntur
Andhra Pradesh, India.

***Abstract* - Cloud storage offers cost effective storage to the users dynamically. Corporate and private users upload their sensitive data items to the cloud storage. But mostly the data stored in the cloud storage is redundant. To avoid redundant storage of the data items and to minimize the maintenance overhead, data deduplication is the most widely used mechanism. Deduplication suffers from various security weaknesses. Considering the popularity of the data items we propose a mechanism to ensure the ownership of the data items in the cloud storage. In addition, we have leveraged the advantages of perfect hash functions and made use of a probabilistic data structure to ensure the ownership of the data items. The efficiency of this mechanism is analyzed considering the popularity of data and ensuring the ownership of data by the users and further provide a detailed performance evaluation.**

Keywords - Perfect Hash Functions, Cuckoo Filter, Probabilistic Data Structure, Proof of Ownership.

I. INTRODUCTION

With the wide growth of computer users and increased production of data throughout the world, cloud storage systems are essential for storing the data. Cloud storage servers are available on demand and their pay per use mechanisms has attracted the computer user to a large extent. Generally, data uploaded to the cloud storage is the replica of already available data items, which wastes the storage capacity of the cloud storage server. Data deduplication is the mechanism, which eliminates the redundant storage of data items by deleting all the redundant copies, and maintains a single copy of the data. Data deduplication strategies based on whether data deduplication takes at the file level or block level; and whether it happens at the user end or at the server side are classified into four categories [1, 9]. Deduplication can be applied at the file level or the block level. Client side deduplication is better than server side deduplication as it saves the upload bandwidth and storage space

As the data uploaded to the cloud storage is available at a remote place, several security doubts arise and it prevents users to migrate to the cloud storage. The general approach to ensure the trust among the cloud users is by allowing them to encrypt the data prior to upload to the cloud storage. As conventional encryption and data deduplication are not compatible, convergent key encryption [4, 8] is used to encrypt the data items to be uploaded to the cloud storage. In this, the key used for encryption is derived based on the actual contents of the data item only. Data item is applied through a hash function and the resulting hash code is used as the key for encrypting the data item. When the key is generated, encryption is done; user will upload the corresponding cipher text to the cloud storage. As the encryption using the convergent key is deterministic, identical data items will produce the same convergent key and the same cipher text.

II. BACKGROUND

Client side data deduplication though is effective in terms of storage space and network bandwidth, it introduces several security concerns. Harnik et al. [5] identified that client side data deduplication suffers from various security problems [10, 19, 20 and 22]. Moreover, two users might collude and can use the Cloud Storage Provider to establish a covert channel for the exchange of information. And also, the users can use the cloud storage as a content distribution network by exchanging the hash values and can share files.

The root cause for these kinds of problems is the usage of the hash value as the proxy for the entire file. To prevent unauthorized access to the data items the concept of proof of ownership [6] is introduced, which will prove whether the user has actually owned the copy of data item or not.

A novel approach is presented to prove the ownership for the data items and eliminates the weaknesses of convergent key encryption. The scheme is proposed with the intuition that data uploaded to the cloud storage requires different levels of security. For example, a file shared among several users such as a popular video, perhaps needs less security than a file owned by an individual such as a research proposal. Accordingly, unpopular/ sensitive files require stronger security than

popular files. With this impression we construct a mechanism that provides varying levels of security for the data items and provides data deduplication based on the popularity of data items. In this only popular data items will be deduplicated whereas, all the unpopular data items are not deduplicated and are provided better security with the conventional stronger symmetric key encryption. The level of security offered to the data files depends on the popularity if the data file. The popularity detection of a file should be secure, as there is a possibility that unauthorized users might try to exploit the weaknesses of the weaker convergent key encryption and might try to compromise the cloud storage provider to gain access to the popular files. The process of detecting the popularity of files should not unveil any kind of information about the ownership of the files and the ownership of multiple users of the file. The process of checking the popularity of the file should ensure privacy for the data items and the users owning those data items.

One of the building blocks of the proposed mechanism is detecting the popularity of the data items in a secure way leveraging the advantages of Dynamic perfect hash functions [14]; it will not disclose any kind of information about the popularity of the data items and the identity of the data owners. As long as the data item is unpopular the question of proving the ownership of file will not arise because the unpopular data items are not deduplicated and are given access based on the identifier of the data items and the key used for encrypting those data items. When popular data items are deduplicated, we have to verify the ownership of the popular data items only. In order to validate the ownership of the data items we have made use of a probabilistic data structure called Cuckoo filter which is primarily used for set membership queries.

III. PROPOSED SOLUTION

A. Overview

The main instinct on which the mechanism is developed that data items do require varying levels of security based on how popular the data is. Consider an example: if a storage system is shared among several users for performing updates/access to the data items in which there are data items which are shared among several users and some data items are uploaded by very few users. The files that are uploaded to the cloud can be alienated as files uploaded/owned by several users and files uploaded/owned by very few users. Files belonging to the former group will be benefited strongly by deduplication as they are popular and may not be sensitive from the confidentiality perspective. Files belonging to the later groups i.e. owned by very few users may contain very sensitive information and require stronger security, but will not benefit a lot by deduplication.

The popularity of data items drives the process of deduplication. Initially, it is considered that all the files are unpopular and are encrypted using the symmetric key encryption. The moment a particular file is uploaded by more than a threshold number of people, we consider it as popular and will try to deduplicate. Once we start deduplicating the files the question of proving the ownership arises, as there is a possibility of unauthorized users compromising the CSP (Cloud Storage Provider) due to weaknesses in convergent key encryption. Verifying ownership process is continued only for the popular files, which are deduplicated.

The major challenge in secure design of the scheme is the secrecy in verifying the popularity of the data files. The process of verifying the popularity of the files should not disclose the identity of the file owners. If proper care is not taken, unauthorized users trying to upload the same file for number of times can mount Sybil attack [18], by which the popularity of the file will be modified. To avoid such attacks, we consider a partially trusted entity. This entity will maintain the identity of the users and prevents the Sybil attack. To prove the ownership of files by the users in secure way we are making use of cuckoo filter data structure.

B. The Proposed Scheme

It consists of users, CSP and Key server. CSP is used to hold data and will verify the ownership of the data. Key server will maintain the list of all unpopular data items and their identifiers.

Prior to uploading the file to the CSP, user will divide file into a set of blocks {b1, b2…bn}. The strategy used for dividing the file into set of block does not have any influence on the proposed mechanism. The user wishing to upload the data segment/block tries to identify the popularity for the data segment by sending the identifier of the data segment to the key server. Key server will compare the identifier with the list of unpopular identifiers it possesses. If it matches, then the cloud storage user will send the symmetric encrypted copy of the data file to the CSP. Though the user is able to verify the popularity of data item in secure way, he has to handle popularity change. I.e. the phase in which a data block reaches the threshold number of uploads with the current upload. As the user is unaware of multiple users who uploaded the copies of data items; to keep track of such information, the Key Server is used. If popularity verification results negative, the user updates the Key server by sending the Identifier of the symmetric encrypted copy of the data block. The moment a data block becomes popular, key server will delete all the information from its storage.

Once the popularity verification indicates a block to be popular, from here on the data block can be deduplicated. Convergent key encrypted copy of the data item is sent to

the CSP. As convergent key encryption suffers from various weaknesses [7][8], proof of ownership is initiated to verify whether users trying to upload/access the popular data items, really own the copy or not.

C. Popularity Checking

To verify the popularity of the data files we have made use of the dynamic perfect hash functions (DPHF). A perfect hash function will map a set of random entries to a group of integers with no collisions. As the perfect hashing can be done in linear space and time complexities, we have considered using it

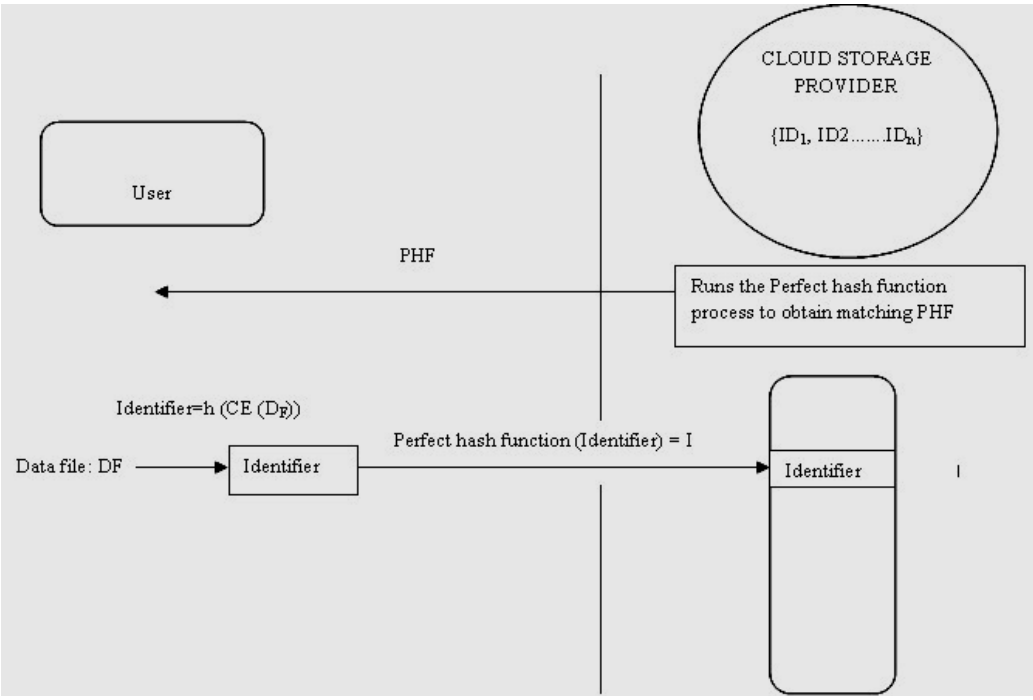

Figure 1. Process of verifying the popularity of data items.

The CSP will execute the process to produce the PHF, harmonizing the identifiers of the convergent encrypted popular blocks available at the CSP. The obtained PHF is encoded and sent to the user. The user will make use of the PHF received from the CSP, computes the identifier thereof using any un keyed hash function like secure hash algorithm (SHA), and gets the look up index I for the data block. Using the index I as an input user will send a lookup query to the CSP.

CSP will return the list of Ids of the convergent encrypted blocks stored under I. The user will verify the popularity of the data item by comparing with the list of popular data items. The whole process [11] is described in the Fig1 mentioned above. The process is secure as the hash functions used are one way

D. Proof of Ownership

In a semi trusted cloud computing environment preserving the privacy of data items is critical. The moment a file becomes popular, the proof of ownership process is initiated. In order to verify the ownership of the data items we have made use of probabilistic data structure called cuckoo filter, which helps in performing the Membership queries over a large pool of items. In the proposed mechanism for a given data item alternate bucket is calculated based on the present location and fingerprint.

For a data item A, calculates the indexes of the buckets as:

H_1 (A) = Hash (A),

H_2 (A) = H_1 (A) $\oplus$ Hash (Fingerprint)

Whenever a file becomes popular, the cloud storage server initializes the cuckoo filter and fingerprints of the data items will be placed in the cuckoo hash table as per the insertion algorithm [13]. Once the cuckoo filter is initialized, from here onwards if any user tries to access the file, the CSP will ask the user to submit the hash code of the file and compares with the existing hash values. Later CSP challenges the user by asking him to submit tokens randomly to provide evidence regarding ownership of the file.. The process by which CSP verifies the ownership is described in our earlier work [12].As cuckoo filter considers the fingerprints of the data items, rather than the original data items, it is more space efficient than other data structures.

IV. SECURITY ANALYSIS

We analyzed the security of the mechanism by considering fraudulent user C_f trying to convince the CSP that, he is a legitimate owner of the file and trying to access the file, without possessing the file in reality. The proposed mechanism will not protect against the user C_f, who takes the help of a legitimate owner of a file to have the responses of POW challenges. We assume that C_f might know a portion of a file, but not the complete file. It is assumed that fraudulent user C_f, knows a byte of the file from a random position with a probability p. If C_f does not have that particular byte, he will be able to guess the byte with a probability q.

The POW process asks the C_f to provide tokens for K blocks from random positions. Upon receiving the token, CSP process the token with a PRF (Pseudo random Function) and the resulting bit string is checked for membership in cuckoo filter. Consider a random block j out of K random blocks requested by the CSP and define the incident in_j, that C_f provides a token leading to the successful POW. It can happen when the C_f correctly guesses and provides the token or when a false positive occurs in the cuckoo filter and it happens with a probability P_f

$$P(in_j) = P(in_j \cap (tk_j \cup tk_j')) = P(in_j|tk_j)\,P(tk_j) + P(in_j|tk_j')\,P(tk_j') = P(tk_j) + P_f(P(tk_j')) \quad (1)$$

Let us analyze the probability that fraudulent user C_f can provide the bits of the J-th block successfully in the event tk_j, and the probability of this event is p. in this case the fraudulent either knows the block or may have guessed . In the later case, fraudulent user either guessed the B unknown bytes with a probability q^B or guess the output of the second hash function, used to produce the l-bit token with a probability of 0.5^l.We assume that always $q^B << 0.5^l$.

Then

$$P(tk_j) = P(tk_j \cap (knw_j \cup knw_j')) = P(tk_j \mid knw_j)\,P(knw_j) + P(tk_j \mid knw_j')\,P(knw_j') = p + (1-p)\,0.5^l \quad (2)$$

Based on equation 1 and equation 2:

$$P(in_j) = p + (1-p)\,0.5^l + p_f(1-(p+(1-p)0.5^l)) = p + (1-p)0.5^l + p_f(1-p)(1-0.5^l) = p + (I-p)(0.5^l + p_f(1-0.5^l)) \quad (3)$$

Fraudulent user C_f is challenged on K block positions. The probability of success of the fraudulent user as:

$$P(Suc) = P(in_j)^k = (p + (1-p)(0.5^l + p_f(1-0.5^l)))^k \quad (4)$$

When n data elements are inserted into the cuckoo hash table the probability of bucket being full can be calculated by:

$$P(n) = \frac{P_t^n X P_{n-t}^{(\ell-1)t}}{p_n^{\ell t}},\ n \in [0, \ell t]$$

P^n_t represents the t permutations of n.

The probability that k number of relocations took place when n^{th} data item is successfully inserted is computed by:

$$P(T = k|N = n) = p^k(n-1)[1-p(n-1)],\ k \in [0, \infty]$$

The probability that less than MNR(Maximum Number of relocations) relocations took place while inserting n^{th} data item successfully is calculated using:

$$P(T < MNR|N = n) = \sum_{k=0}^{MNR-1} P(T = k|N = n) = 1 - p^{MNR}(n-1)$$

The probability of inserting nth data item successfully but unable to insert $(n+1)^{th}$ data item is calculated by:

$$\Theta(N = n) = \{\pi_{i=1}^n P(T < MNR|N = i)\}\,XP(T \geq MNR|N = n+1) = \{\pi_{i=1}^n [1 - P^{MNR}(i-1)]\}\,XP^{MNR}(n)$$

$$E[N] = \sum_{j=0}^{\ell t} JX\Theta(N = J) = \sum_{j=0}^{\ell t} \{JX\pi_{i=1}^J [1 - P^{MNR}(i-1)]XP^{MNR}(J)\}$$

The expected number of items inserted in to the cuckoo filter is:

$$E[N] = \sum_{j=0}^{\ell t} JX\Theta(N = J) = \sum_{j=0}^{\ell t} \{JX\pi_{i=1}^J [1 - P^{MNR}(i-1)]XP^{MNR}(J)\}$$

The anticipated number of relocations while inserting n^{th} data item is:

$$E[R](k,c) = \sum_{k=0}^{MNR-1} kXP(T = k|N = n) = \sum_{k=0}^{MNR-1} kXP^k(n-1)[1-p(n-1)]$$

The total numbers of relocations are:

$$SUM_E = \sum_{c=1}^{E[N]} E[R]$$
$$= \sum_{c=1}^{E[N]} \sum_{k=0}^{MNR-1} kXP^k(c-1)[1-p(c-1)]$$

The detailed analysis describes that the proposed mechanism is secure enough to block dictionary attack and Sybil attack. In terms of computational complexity, the proposed mechanism produced better performance compared to the state of the art mechanisms

V. IMPLEMENTATION AND EXPERIMENTAL RESULTS

A. Implementation

The scheme was implemented using C++. For all the cryptographic operations, made use of Open SSL Crypto Library [16]. Client and Key server run on an Ubuntu Virtual Machine on Open stack Platform. Information related to the unpopular data items is maintained using REDIS [15] by key server. Meta data regarding the data blocks such as identifiers, encrypted keys are stored using the MySQL. To implement the perfect hash functions, made use of CMPH Library. The original Hash function is replaced with the secure one-way hash function SHA [17].

B. Experimental Results

To choose the best probabilistic data structure, evaluated the proposed mechanism by considering the bloom filter as the baseline. Compared the performance of cuckoo filter with bloom filter and blocked bloom filter. The process of using cuckoo filter gave better results than the mechanisms implemented using bloom filter in terms of insertion time, space required, average memory references needed, false positive rate and the computational speed, is shown in Table 1. The entire set of filters is configured with the same size of 128 MB. Among all the filters, cuckoo filter exhibited better false positive rate using the same amount of space of 12.40 bits/item.

Therefore, we are able to insert more elements in to the cuckoo filter. The insertion of elements into cuckoo table compared to the blooms filter is described in the chart given below. When the table is less filled, the cuckoo filter exhibits the highest number of inserts, whereas when the occupancy of the table increases insertion becomes slow.

TABLE1: SPACE EFFICIENCY AND LOOKUP PERFORMANCE

	Cuckoo Filter	Bloom Filter	Blocked Bloom Filter
Bits per item	12.40	12.80	13.00
False positive rate	0.19%	0.23%	0.43%
Number of items (millions)	127.57	123.68	123.68
Number of memory references	2	2	2
Load factor	95.5%	95.5%	95.5%

But on the whole cuckoo filter offered better insertion rate compared to other probabilistic data structures and is shown in fig 2 given below in terms of millions of operations (MOPS). We analyzed the complexity of the proposed work by considering all the probabilistic data structures and the results are given below. When performing the search operations, the time required in identifying the data element and the time needed to confirm the nonexistence of the data element is analyzed and is shown in the figure given below.

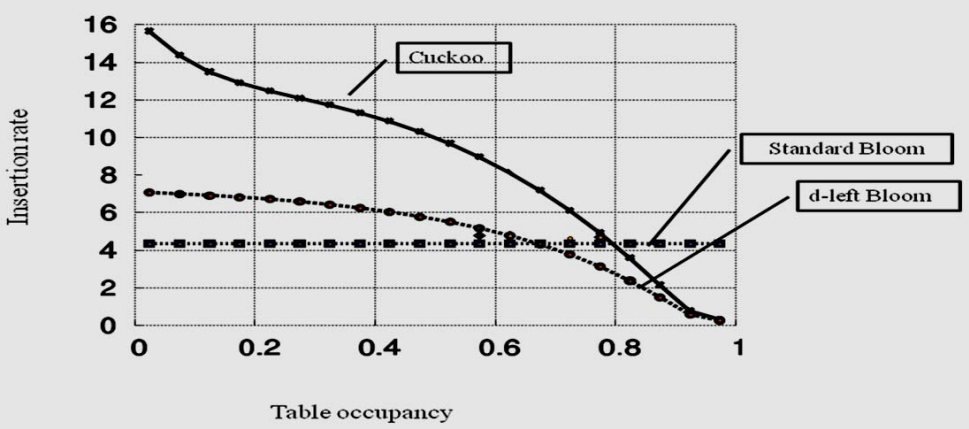

Figure2. Insertion Performance (MOPS).

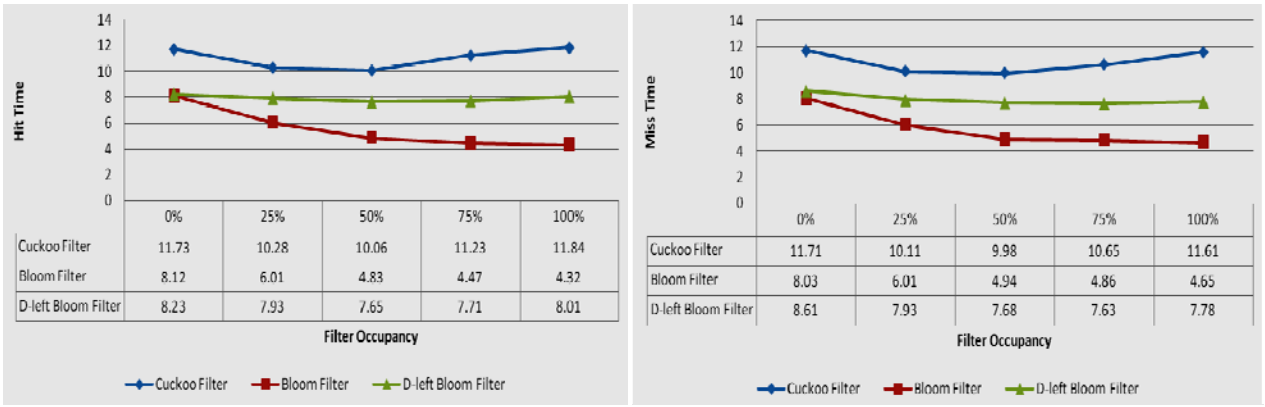

Figure3. Search Performance

Cuckoo filter offers better performance compared to the Standard Bloom Filter and D-left Bloom Filter while performing search operations. Standard bloom filter doesn't support the delete operation, whereas the same is possible with the cuckoo filter. The problem of false positive rate is also less in cuckoo filter compared to other probabilistic data structures. [3].

To compare the performance of the proposed work with the existing state of the art mechanisms, we refer the proposed work as Fpow, the work proposed by Halevi et al[6] is referred as Hpow. The work presented by DI Pietro et al[23] is referred as Dpow .

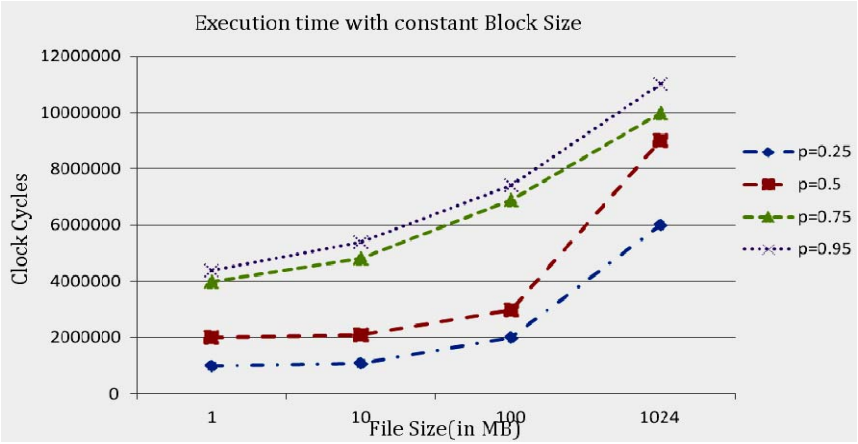

Figure 4. Performance analysis with Constant Block Size

Implemented all the mechanisms in C++, used Open SSLCrypto library for all the cryptographic operations. Considered the threshold value as 128, the probability that an unauthorized user knows the block of a file is set to {0.25, 0.50, 0.75, and 0.95} the file sizes are chosen as {1, 10, 100, and 1024}. The relative performance is shown the figure 4. The block sizes are chosen as {16, 128, 256,512, and 1024} bytes, the number of blocks to which the data owner has to produce tokens is set to {105, 210, 515, and 1020} the execution time of all the mechanisms are shown in figure 5.

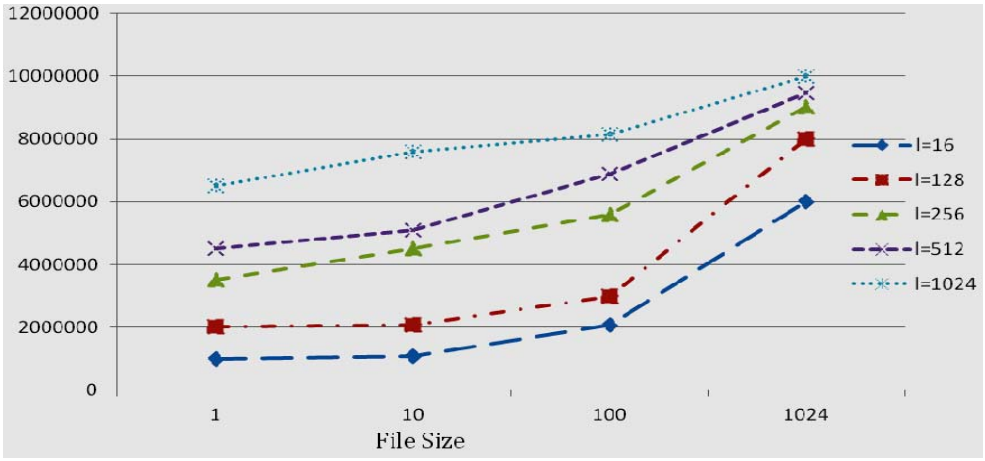

Figure 5. Execution of the Algorithm with Constant P: Performance analysis with variable Block Size

Execution time of the Fpow ,Dpow, Hpow is compared considering the different block sizes and the probability thatfradulent user might guess the portion of thr block is considered as 0.95 When the file size is less Fpow mechanism is better than Dpow,Hpow by small percentage only.When the file size increases proof of ownership provided by cuckoo filter is fast at the client side. The computation time involved at the client side for all the mechanisms isdescribed in figure 6.

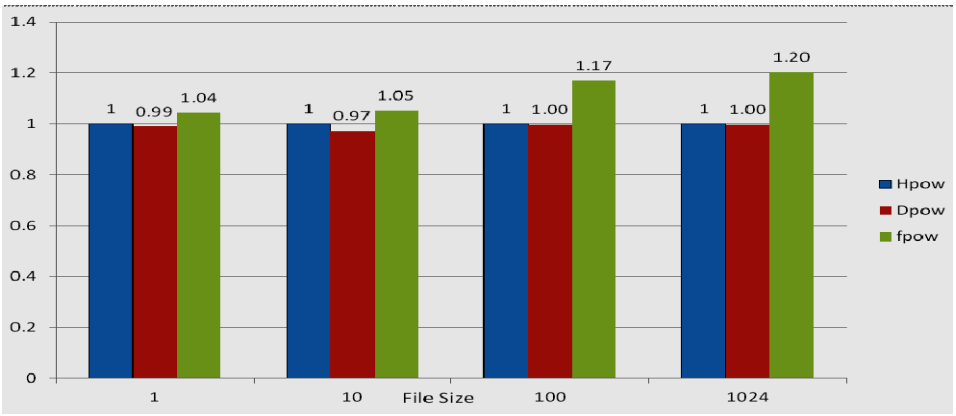

Figure 6. Speed Comparision at the Client Side

Interms of memory requirements also Fpow is offereing better peroformance compared to Works proposed by Halevi et al and Di pietro et al. when the file size is less the performance benefit in fpow is small but as the file size grows ,we are able to achieve better performance.

VI. CONCLUSION

The proposed mechanism dealt with the conflict of interest between storage optimization and security of the data in the cloud storage. We have focused on providing varying levels of security for the data items based on how popular the data is. Provided trade-off between storage space efficiency and privacy of data and users. We analyzed the probability of fraudulent users trying to convince the cloud storage and a thorough analysis has been provided. The proposed approach leveraging the advantage of perfect hash functions and cuckoo filter offers better performance than the existing mechanisms .The work proceeds in reducing the false positivity and provide efficient cloud storage system.

REFERENCES

[1] Anderson, P., & Zhang, Z. (2010) Fast and secure laptop backups with encrypted deduplication. In Proceedings of the 24th International Conference on Large Installation System Administration (LISA'10), (pp.1-12).

[2] Adams, C.,(2011) Dictionary Attack. In: van Tilborg H.C.A., Jajodia S. (eds) Encyclopedia of Cryptography and Security, Springer, Boston, MA.

[3] Bonomi, C., & Mitzenmacher, M., & Panigrahy, R.(2006)Beyond Bloom filters: From approximate membership checks to

approximate state machines. In Proc. ACM SIG-COMM, Pisa, Italy, August.

[4] Bellare., Mihir., Srirae Keelveedhi., & Thomas Ristenpart. (2013) Message-locked encryption add secure deduplication, In Advances in Cryptology EUROCRYPT (pp.296-312), Springer Berlin Heidelberg.

[5] Harnik., Dahny., Benny Pinkus.,& Alexandra Snalman-Peleg. (2010). Side channels in cloud Services, the case of deduplication in cloud storage, IEEE Security & Privacy 8(6), 40-47.

[6] Halevi, S., Harnik, D., Pinkas, B., & Shulman-Peleg, A. (2011). Proofs of ownership in remote storage systems, In Proceedings of the 18th ACM conference on Computer and communications security. ACM(pp. 491500).

[7] Hubert Ritzdorf., Ghassan Karame., Claudio Soriente., & Srdjan Capkun. (2016). On Information Leakage in Deduplicated Storage Systems, Proceedings of the 2016 ACM on Cloud Computing Security Workshop, October 26-28, Vienna, Austria.

[8] Li, J., chen, X., Li, M., Li, J., Lee, P., & Lou, W. (2013). Secure deduplication with efficient and reliable Convergent key management, In IEEE Transactions on Parallel and Distributed Systems.

[9] Paulo, J., & Pereira, J. (2014). A survey and classification of storage deduplication systems. ACM Computing Surveys, 47(1):11.

[10] Philip Shalane, Ravi Chitloor, Uday Kiran Jonnala. (2016). 99 deduplication problems, Proceedings of the 8th USENIX Conference on Hot Topics in Storage and File Systems, (pp.86-90), June 20-21, Denver, CO.

[11] Tirapathi Reddy, B., & Chandra Sekhara Rao, M.V.P. (2016). Performance Evaluation of Various data deduplication Schemes in Cloud Storage, International Journal of Pure and Applied Mathematics, 116(5), 175-180.

[12] Tirapathi Reddy, B. & Chandra Sekhara 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(12), 2121-2132.

[13] Tirapathi Reddy, B. & Chandra Sekhara Rao, M.V.P. (2018). Privacy-Preserving Proof of Ownership for Data in Cloud Storage Systems International Journal of Engineering and Technology, 2018, Vol.7 (2.8) 13-17.

[14] https://www.arl.wustl.edu/~ sailesh/download_files/Limited_Edition/hash/ Dynamic%20 Perfect%20 Hashing-%20 Upper%20 and%20 Lower%20 Bounds.pdf

[15] https://redis.io/

[16] https://www.openssl.org/docs/man1.0.2/crypto/crypto.html

[17] https://csrc.nist.gov/CSRC/media/Projects/Cryptographic-Standards-and-Guidelines/ documents/examples/SHA1.pdf

[18] Xiao, L., Greenstein, L. J., Mandayam, N. B., & Trappe, W. (2009). Channel-based detection of sybil attacks in wireless networks, IEEE Transactions on Information Foren-sics and Security, 4(3), 492503.

[19] Xu., Jia., Ee-Chien Chang., & Jianying Zhou. (2013). Weak leakage-resilient client-side deduplication of encrypted data in cloud storage, In Proceedings of the 8th ACM SIGSAC symposium on Information,(pp.195-206.) computer and communications security, ACM.

[20] Youngjoo Shin., Dongyoung Koo., Junbeom Hur. (2017). A Survey of Secure Data deduplication Schemes for Cloud Storage Systems, ACM Computing Surveys (CSUR), 49(4), 1-38.

[21] Leesakul, Waraporn, Paul Townend, and Jie Xu. "Dynamic data deduplication in cloud storage." Service-Oriented System Engineering (SOSE), 2014 IEEE 8th International Symposium on. IEEE, 2014.

[22] Stanek, Jan, Alessandro Sorniotti, Elli Androulaki, and Lukas Kencl. "A secure data deduplication scheme for cloud storage." In Financial Cryptography and Data Security, pp. 99-118. Springer Berlin Heidelberg, 2014.

[23] Pietro, Roberto Di and Alessandro Sorniotti. "Boosting efficiency and security in proof of ownership for deduplication." In proceedings of 7th ACM Symposium on Information , Computer and Communications Security , AsiaCCS (2012).