International Journal of Engineering and Advanced Technology (IJEAT)
ISSN: 2249-8958 (Online), Volume-9 Issue-3, February 2020

Secure Block Level De-Duplication on Big Data using Proof of Ownership

V. Ezhilarasi, K. Kulothungan, L. Sai Ramesh

Abstract: ***De-duplication technology is commonly used to decrease the space and bandwidth requirements of services by eliminating repeated data and store only a single copy of them. Unfortunately, it raises issues relating to security and ownership such as, unauthorized users may claim as owner of that file and security threat from curious server itself. To overcome these issues, Secure Block level de-duplication with Proof of Ownership Scheme is proposed in this work. Proof-of-Ownership Scheme allows any owner of the same data to prove to the server that he owns the data in a robust way. This scheme uses convergent encryption and it protects the data from attackers and honest but curious server. It also reduces storage space efficiently by checking the uniqueness of data at block level.***

Keywords: de-duplication, encryption, secure, block.

I. INTRODUCTION

Data de-duplication is a process of compression for removing the multiple copies of same data in the same database or login. Data de-duplication is operated at the file-level, block-level, and bit-level. If two files are having the same content with different file name then it would pointed by different pointers for the same content. In block-level de-duplication and bit-level de-duplication, it seems the content are available within the same file and sometimes it may saved in different files in different iterations. Block-level is more efficient process than file-level de-duplication.

Big data is characterized by three features such as an volume, variety and analysis need to be done on the data. Big data doesn't equate to any specific volume of data, the term if used to describe terabyte, petabyte, and Exabyte. Data may be raw or preprocessed using separate software tools before analytics are applied.

To manage encrypted data with de-duplication in an efficient way is a practical issue. Current industrial de-duplication solutions cannot handle encrypted data. De-duplication is suffered from brute-force attacks and it raises the issues related to security and ownership. To overcome this problem the Proof of Ownership has been proposed. Proof of ownership is a protocol used to communicate between a prover and verifier. By executing this protocol, the provider has to convince the verifier that he/she is an owner of a file. Protection mechanisms against this kind of adversary focus on the convergent encryption.

The main objective of this work is to eliminate the duplicate copy of a redundant file. Each de-duplicated file is checked for authorized data owner by implementing the proof of ownership protocol, thus it identifies the rightful owner of that file. The Proof of Ownership(POW) protocol verifies the ownership of the data by generating the hash tree. The hash tree is verified by the original hash tree that is generated from a file stored on the server. By implementing the proof of ownership protocol, the rightful owner of a file can be identified by generating block level Merkle hash tree.

II. REVIEW OF LITERATURES

Some research works carried out to reduce the attacks and prevent the data and avoid duplication by effective techniques which are discussed in this section. Jin Li et al. [1] has proposed the convergent encryption technique to encrypt the data before outsourcing. The notion of authorized data de-duplication was imposed by providing several level of privileges for user based on their authentication level. The key generation techniques are also used here to delete the duplicate files.

Mi Wen et al. [2] proposed a another mode of convergent key technique by providing session based key management technique. In this system, the data owner can verify their information is duplicated or not in somewhere without get access permission by other data owner in the encrypted form itself. Lorena Gonzlez-Manzano and AgustinOrfila [3] proposed the preservation of the confidentiality by verifying the proof of owner using convergent encryption technique. It doesn't require any additional key management algorithm to verify the ownership of the data owner.

Zheng Yan et al. [4] provided the access control mechanism on cloud data by verifying the encrypted data in cloud storage. This technique integrates the existing re-encryption technique with the de-duplication to implement the proposed scheme. Jin li et al. [5] gives the suggestion to avoid duplication in the stored data by storing data in different servers in the form of chunks. The chunks are tagged b secret key that will known only to the data owner by secret key sharing mechanism.

Revised Manuscript Received on February 06, 2020.
* Correspondence Author
V. Ezhilarasi, Department of Information Science and Technology, CEG Campus, Anna University, Chennai, India. Email: elz.iit@gmail.com
K. Kulothungan, Department of Information Science and Technology, CEG Campus, Anna University, Chennai, India. Email: kulo.tn@gmail.com
L. Sai Ramesh*, Department of Information Science and Technology, CEG Campus, Anna University, Chennai, India. Email: sairamesh.ist@gmail.com

Junbeom Hur et al. [6] suggested the server side de-duplication scheme for the encrypted data. This considered as the centralized mechanism for data sharing where the duplicate data will be removed by the server itself with the notification to the data owner. It reduces the overload on the individual data owner side and also maintains the cloud storage without duplicate data. It also prevent the data leakage because of centralized management. Nesrine Kaaniche et al.

[7] suggested the side scheme for avoiding duplication in the cloud storage by providing the privileges to the data owner to verify their data contains any duplication before it going to be stored in the public cloud. The usage of symmetric encryption is enciphering the file and identifies the duplication over the encipher information.

Roberto Di Pietro et al. [8] presented a protocol for maintaining the ownership of the data owner in de-duplication environment. It tries to reduce the computational complexity on client side by implementing de-duplication mechanism on cloud server. Ruiying Du et al. [9] introduced a technique for retrieving the deleted duplicated information from the cloud based on proof of ownership mechanism. This work considers the proof of ownership provided by actual owner and based on that the data will retrieved for the specific data owner based on the request. Shai Halevi et al. [10] proposed a another kind of PoW mechanism by achieving the identification of ownership of the client by analysing the piece of information provided by the user instead of analyzing whole data file. This also makes the attackers to fail in their attacking status by only providing the piece of data even in cipher text.

Sairamesh et. al [11] gives the system for sharing data from multiple owners in the common cloud on more secure way. The key management algorithm is proposed to ensure the secure transmission of data among multiple user's. Thangaramya et. al [12] used the Map reduce technique to share the same data with multiple user's based vertically partitioned data mechanism. And the work mentioned in [13] discusses about secure data storage with decentralized access control. This technique reduces the cost for common server for key management.

III. PROPOSED SYSTEM ARCHITECTURE

Data de-duplication is carried out for checking data redundancy and eliminate the duplicate files. This technique is used to improve storage utilization and can also be applied to network data transfers to reduce the number of bytes that must be sent. A file is encrypted using convergent encryption before uploaded. It is a cryptosystem that produces identical cipher text from identical plaintext files. For that a tag is generated for each file and with that tag, file de-duplication is carried out. Each deduplicated file is checked for authorized data owner by implementing the proof of ownership protocol, thus it identifies the rightful owner of that file.

The Proof of ownership(POW) protocol verifies the ownership of the data by generating the Merkle hash tree. Hash trees allow efficient and secure verification of the contents of large data structures. Hash trees are a generalization of hash lists and hash chains. The proof is verified by the original hash tree that is generated from a file stored on the server. Figure 1 describes the system architecture of the de-duplication on encrypted data using proof of ownership.

A. User Authentication

The purpose of this module is to provide an authentication service, allowing callers to determine whether a username/password combination is valid or not. Authentication is a process in which the credentials provided are compared to those on file in a database of authorized user's information. If the credentials match, the process is completed and the user is granted authorization for access. This module checks the given username and password is correct or not based on already registered details.

B. Convergent Encryption

Convergent encryption, also known as content hash keying, is a cryptosystem that produces identical cipher text from identical plaintext files. This has applications in cloud computing to remove duplicate files from storage without the provider having access to the encryption keys. The main idea of the convergent encryption is that each data owner encrypts his data by using a convergent key K with symmetric encryptions.

KeyGenCE(M) – K is the key generation algorithm that maps a data copy M to a convergent key K.

EncCE(K,M) – C is the symmetric encryption algorithm that takes both the convergent key K and the data copy M as inputs and then outputs a cipher text C=E(K,M).

DecCE (K,C) – M is the decryption algorithm that takes both the cipher text C and the convergent key K as inputs and then outputs the original data copyM.

TagGen(M) – T(M) is the tag generation algorithm that maps the original data copy M and outputs a tag T(M).

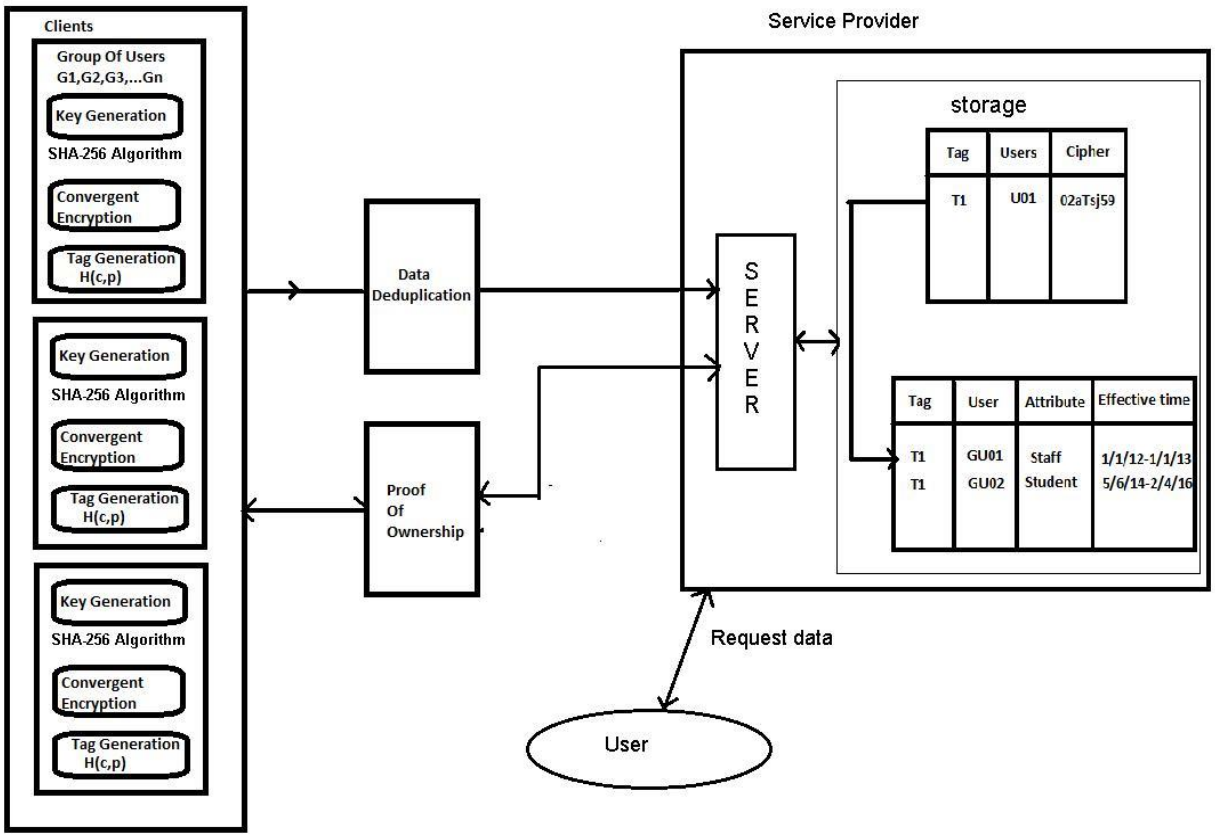

Fig. 1. Block level Data De-duplication with Proof of Ownership

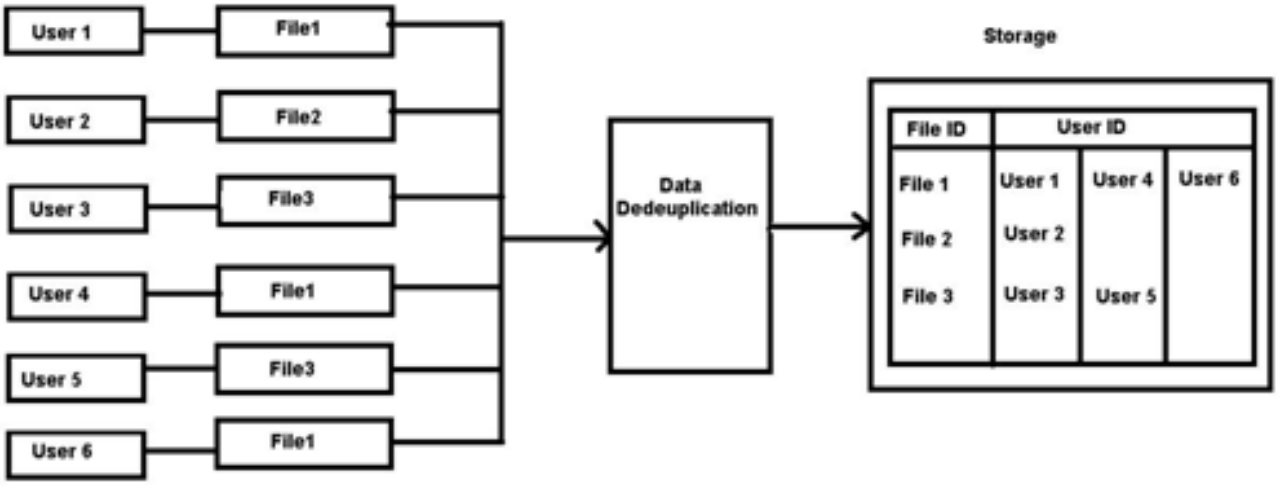

Fig. 2. De-duplication Process

C. Key Generation

Key generation is the process of generating the key using the hash algorithm. A key is used to encrypt and decrypt the data whatever data being encrypted and decrypted. To apply convergent encryption at the file level, each file should be encrypted using the secret key which is derived from the file itself. The user has to maintain a secret key for decryption. The SHA (Secure Hash Algorithm) is the cryptographic hash functions used to generate the unique key based on the file content.

Convergent Encryption(CE) is a deterministic symmetric encryption scheme in which the key **K** is derived from the message **M** itself by computing **K = H(M)** and then encrypting the message as C=E(K,M) =E(H(M),M) where, **H** is a cryptographic hash function and **E** is a block cipher.

D. Tag Generation

Tag generation is the process of generating unique file tags for user files using the hash algorithm. Convergent encryption algorithm also has a tag generation algorithm that derives a tag from a cipher text. The tag of a file is generated for checking the de-duplication of a file. Secure Hash Algorithm (SHA) is a hash function used in cryptography which is used to generate the unique tag based

on the ciphertext. After the tag generation, each tag is uploaded to the server and it is cross verified with previously uploaded tags. If the tag exists, it links that tag to the previously existing tag.

Tag generation algorithm derives a tag T from the ciphertext C by itself by computing T=H(C). The cipher text is derived from the message M as C=H(K,M)=H(H(M),M) where H is a cryptographic hash function and E is a block cipher.

E. De-duplication

Data de-duplication technique ensures that only one unique instance of data is retained in storage. In file level de-duplication the data redundancy is exploited on the file level and thus only a single copy of each file is stored on the server. For de-duplication check, the tag is generated from the encrypted file is needed. The de-duplication process is carried out by checking the file tag. It checks the newly uploaded tag with previously updated tags. If the file tag is not present in the previously updated tags the file gets uploaded otherwise, the file tag will be linked to the already stored file. It reduces the storage space and cost for storage. Figure 2 describes the flow of the de-duplication process.

To perform de-duplication each encrypted file is hashed using SHA-256. A 32-byte hash value will be generated which is assigned as file tag. Now, file tag is uploaded into the server to check redundancy of the file. If duplication occurs the file tag will be linked to the previously existing tag. Otherwise, it creates a new file.

F. Proof of Ownership

Proof-of-Ownership is an interactive protocol between a prover and a verifier. By executing the protocol, the prover convinces the verifier that he is the owner of a file stored by the verifier. In client side de-duplication anyone in possession of the file hash can gain ownership of the file by uploading the file hash. Motivated by this observation, proof of ownership (POW) has been proposed. A POW scheme is jointly executed by the server and client such that the client can prove to the server that it is indeed in possession of the file.

Pseudo code for de-duplication process

Input: File Tags
Output: Duplicate File Tag

1. f1 and f2
2. Both file f1 and f1 has a same content
3. Start with an empty list x [ ]
4. if tag[f1]== tag[f2] then
5. The file f2 is deduplicated and need share path of file f1.
6. for i = 0,i <= 10,i + + do
7. f 1path randomGenerator.nextInt[i]
8. end for
   a. fpath f 1:get path
9. path[f1] f
10. Path of file f1 is assigned to f2 and content has been shared.
11. else
12. for i = 0,i <= 10,i + + do
13. newpath randomGenerator.nextInt[]
14. end for
15. f 2: path newpath;
16. New path has been assigned for file f2.
17. end if

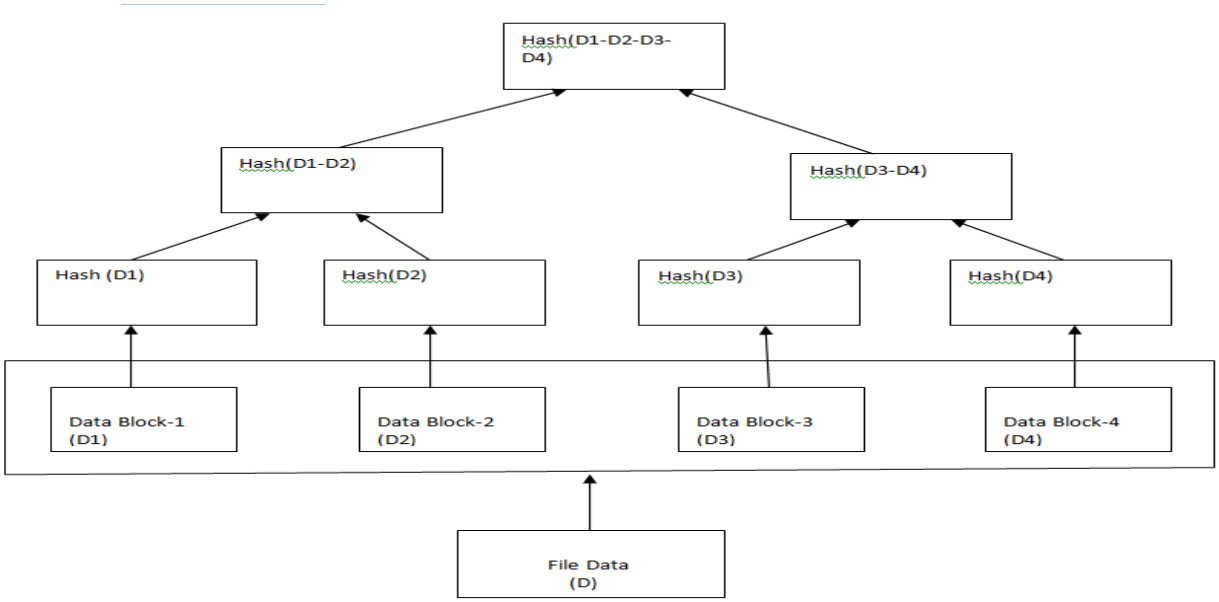

Fig. 3. De-duplication using Merkle tree

G. Block level Merkle Hash Tree

Merkle tree is a tree in which every non-leaf node is labeled with the hash of the labels or values (in case of leaves) of its child nodes. In Block level Merkle hash tree, big data file is split into equal sized blocks and each block is hashed and stored as nodes. Hash trees are a generalization of hash lists and hash chains obtained from the above process is assigned as a parent of the leaf nodes. This process keeps running until a single root is obtained. Server preprocesses the file and stores some short information per file (Tags). Figure 3 provides the structure of the Block level Merkle hash tree.

When user uploads the file, user has to generate Block level Merkle hash tree and has to send the block hash to server, server checks for the uniqueness of block hashes sent from client. If unique block hash is identified then, the client is added as one of the owner of that block and pointer is given to that block. So, this proposed scheme efficiently provides de-duplication of big data files by checking even for the similarity in blocks.

Proof stage: A challenge response is done only during file upload.
Prf: Prover takes a challenge **Q** and a file **M** as input, returns a response **R**.
Ver: Verifier takes a challenge **Q**, the file **M** and the response **R** as input, returns that he/she has a file or not.

IV. EXPERIMENTS AND RESULT ANALYSIS

The experiments are conducted in Cloudsim environment where the four different instances are created and considered as users. Every user's are stored their data in the cloud environment and system verifies the occurrence of duplication in the stored data by executing the proposed technique. Based on that experiments, the performance of the proposed technique is evaluated and comparative analysis are carried out. Table 1 shows the actual space utilized by individual memory in the available space.

Table I. Space Utilized by Individual user

User	Space utilization in GB	
	Before De-duplication	After De-duplication
u 1	8.2	8.2
u2	5.5	0
u3	2.5	0
u4	5	5

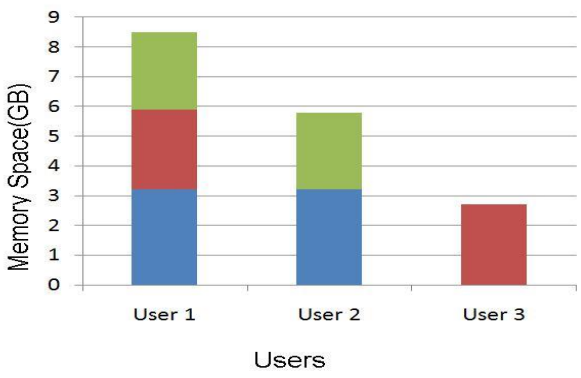

Fig. 4. User Memory Space - Before De-duplication

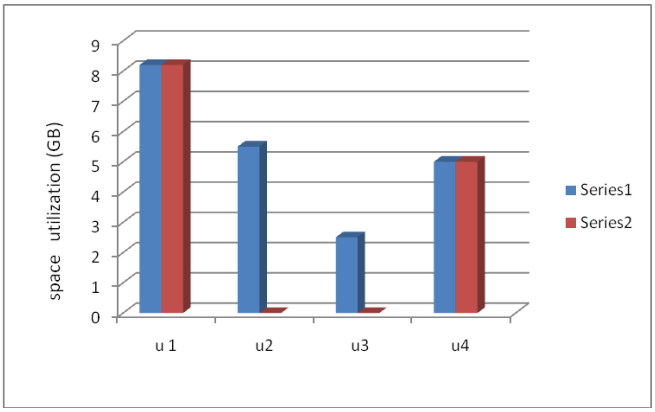

Fig. 5. Space Utilization of user before and After de-duplication

Figure 4 shows the chart for registered users and how much memory space it has occupied before de-duplication. This chart shows multiple users trying to upload the same file, in which the server that stores each file separately. It consumes a lot of memory space to store the same file again.

Figure 5 shows the chart for registered users and how much memory space it has occupied after de-duplication. This chart shows multiple users trying to upload the same file, in which the server that does not store the file separately. Instead of storing the same file, it will provide the reference of that file which is already updated by another user.

V. CONCLUSION

In recent days, people move to the public cloud to store their data and managing their transactions over the cloud itself. In this regard, duplication is the important scenario where sometimes data can be deleted by the cloud server to avoid duplication or sometimes any other data owner can update the duplicate information with their data. This proposed system is provided the scheme for de-duplication in an encrypted data to maintain the proof of ownership for data file. This work supports the data updation in encryption form and block level chunking helpful to avoid the overwriting of unwanted data in other data owner files.

The results of our computer simulations further showed the practicability of our scheme. In future, the work can be extended for dynamic data access control for block level de-duplication with proof of ownership, so that the storage efficiency and security can be further enhanced.

REFERENCES

1. Jin Li, Yan Kit Li, Xiaofeng Chen, Patrick P.C. Lee, and Wenjing Lou. A hybrid cloud approach for secure authorized de-duplication. IEEE Transactions on Parallel and Distributed Systems, vol. 26:pp. 1206 – 1216, 2015.
2. Mi Wen, Kaoru Ota, He Li, Jingsheng Lei, Chunhua Gu, and Zhou Su. De-duplication with reliable key management for dynamic updates in cpss. IEEE Transactions on Computational Social Systems, vol.2:pp. 137 – 147, 2015.
3. Lorena Gonzalez´-Manzano and Agustin Orfila. A efficient confidentiality-preserving proof of ownership for de-duplication. Journal of Network and Computer Applications, vol. 50:pp.49 – 59, 2015.
4. Zheng Yan, Wenxiu Ding, Xixun Yu, Haiqi Zhu, and Robert H Deng. De-duplication on encrypted big data in cloud. IEEE Transaction on big data, vol. 2:pp.138 – 150, 2016.

5. Jin Li, Xiaofeng Chen, Xinyi Huang, Shaohua Tang, and Yang Xiang. Secure distributed de-duplication dystems with improved reliability. IEEE Transaction on computers, vol. 64:pp. 3569 – 3579, 2016.
6. Junbeom Hur, Dongyoung Koo, Youngjoo Shin, and Kyungtae Kang. Secure data de-duplication with dynamic ownership management in cloud storage. IEEE Transactions on Knowledge and Data Engineering, vol. 28:pp. 3113 – 3125, 2016.
7. Nesrine Kaaniche and Maryline Laurent. A secure client side de-duplication scheme in cloud storage environments. IEEE International Conference on New Technologies, Mobility and Security (NTMS), pages . 1–7, 2014.
8. Roberto Di Pietro and Alessandro Sorniotti. Proof of ownership for de-duplication systems:a secure, scalable and efficient solution. IEEE Transactions on computer communication, vol. 82:pp. 71–82, 2016.
9. Ruiying Du, Lan Deng, Jing Chen, Kun He, and Minghui Zheng. Proofs of ownership and retrievability in cloud storage. IEEE International Conference on Trust, Security and Privacy in Computing and Communications, pages 328 – 335, 2014.
10. Shai Halevi, Danny Harnik, Benny Pinkas, and Alexandra Shulman-Peleg. Proofs of ownership in remote storage systems. Proceedings of the 18th ACM conference on Computer and communications security, pages 491 – 500, 2012.
11. SaiRamesh, L., S. Sabena, K. Thangaramya, and K. Kulothungan. "Trusted multi-owner data sharing among dynamic users in public cloud." Australian Journal of basic applied sciences, vol. 10, no. 2, pages. 315-319, 2016.
12. Kalidoss, Thangaramya, Ganapathy Sannasi, Sairamesh Lakshmanan, Kulothungan Kanagasabai, and Arputharaj Kannan. "Data anonymisation of vertically partitioned data using Map Reduce techniques on cloud." International Journal of Communication Networks and Distributed Systems 20, no. 4 (2018): 519-531.
13. Jayakumar, J., Sairamesh, L., Pandiyaraju, V., Muthurajkumar, S. and Rakesh, R., 2015. Secure data storage using decentralized access control in cloud. Advances in Natural and Applied Sciences, 9(6 SE), pp.192-197.

AUTHORS PROFILE

Ms. V. Ezhilarasi is the research scholar in Department of Information Science and Technology, Anna University. She completed her B. E from Bharathidasan University and M. E from Anna University in the year of 2004 and 2013 respectively. Her research area is cloud security including key management and block level duplication.

Dr. K. Kulothungan currently working as an Associate professor in Department of Information Science and Technology, Anna University. He completed his Ph. D from Anna University in the year of 2013. He guided more than ten research scholars. He has published more than 50 research articles in reputed International Journals and Conferences. His current area of research is Wireless Networks and security.

Dr. L .SaiRamesh is currently working as a faculty in Department of Information Science and Technology, Anna University. He completed his Ph. D from Anna University in the year of 2015 in the area of cloud security. He published more than 50 research articles in reputed International Journals and Conferences. His current area of research is Cloud security and Machine learning approaches in Image processing.