Webology (ISSN: 1735-188X)
Volume 18, Number 6, 2021

Dedupframe: Data Deduplication Framework For Effective Management Of Cloud Storage

Dr.G.Uma

Assistant Professor Department of Computer Science Srimad Andavan Arts and Science College-Trichy (Affiliated to Bharathidasan University)

Abstract

Cloud provides a virtual environment for storing a massive amount of data. Today's scenario, day by day, the size of data generation is increased. It needs a place to keep safe. Cloud is the only solution to store an enormous volume of data. However, when storing data in the cloud, it is a third party environment, and it can be used by different users all around the world. The place should ensure the security or privacy of data stored in the cloud storage. When using cryptographic techniques, the data are encrypted for maintaining security. Encryption produces a ciphertext, which means encrypted data, and it is generated based on the key used for encryption. So, it is understood that different keys could produce different ciphertexts for the same plaintext or original text. Convergent encryption is used for avoiding duplicate data, but the key used for the data for doing encryption at the first time is maintained and shared with all users who have the same and similar content of data to upload to the cloud storage. Sharing the key to all the users is critical in the open cloud environment. Users met many problems when using cloud storage services. Cloud storage is allocated for storing the same data multiple times. It creates difficulties in storage management. To address all these issues, it is necessary to propose an efficient framework to maintain the data in the cloud without duplication. This proposes a DEDUPFRAME for avoiding duplicated data stored in the cloud. the framework comprises different components to maintain the data in the cloud. the research work is implemented in the cloud environment and evaluated with test data. The result derived from the implementation is satisfactory for avoiding duplicate data in the cloud storage.

Keywords:- Deduplication; Convergent Encryption; security; Framework; Cloud Storage;

Introduction

Cloud is an advanced computing-oriented technology, which provides computational resources as a service. Most importantly, storage is the primary service provided by the cloud. Cloud offers many services like SaaS, PaaS, and IaaS, whatever the users use these services, finally, they have stored some data in the cloud. Moreover, the cloud is a public environment. Anyone can use cloud storage through any cloud service. In this situation, there is a chance that the same data can be uploaded and stored by different independent users. To secure the data stored in the cloud, uploading data are secured by using some traditional encryption techniques before they are stored in the cloud [1]. Different users use conventional encryption with their key, and it produces different encrypted data for the same original data. Each user's encrypted data is stored in the cloud in a different form. However, the content of the data is unique.

In this scenario, the same data could store in the cloud storage multiple times, and storage is allocated for the same data multiple times. It causes wastage of storage in the cloud. If it is for a small amount of data, then it is not a matter, but the cloud is an unlimited storage service provider. It is used by many enterprises and organizations to store their data in the cloud. In the future, it is analyzed by the International Data Corporation (IDC) and delivered a report stating that the usage of data in the world will increase to 40 trillion gigabytes by 2025 [2].

Users met many problems when using cloud storage services. Cloud storage is allocated for storing the same data multiple times. It creates difficulties in storage management. Cloud allow duplicate data to store in the cloud storage. It creates unwanted storage allocation in the cloud. The traditional cryptosystem is not suitable for maintaining non-duplicate data in the cloud storage environment. Deduplication mechanism is vulnerable to many attacks like brute force attack and dictionary attack, poison attack which includes duplicate faking attack and erasure attack. Clients have more burden on maintaining the key for convergent encryption on the client-side; it is more difficult for users. The convergent encryption key should be the same for all users who have the same data, but sharing of this key to all the users of the data owner is vulnerable to security attacks. To manage the data effectively in the cloud, it is essential to minimize the duplicate storage of data in the cloud. To avoid duplication in cloud storage, it is suggested to adopt a well-known methodology known as deduplication [3].

Deduplication supports in eradicating numerous replicas of similar data in cloud storage. Deduplication is the only possibility to avoid duplicate data uploaded to the cloud. This paper proposes a data deduplication framework to manage cloud storage without duplicate data effectively. The framework designed for file-level data deduplication, and it is an online deduplication framework, which means the data are verified for duplication before it is uploaded to the cloud [4]. The process of deduplication allows only to keeps the file, which is uploaded first by the user in the cloud, and for each following upload requests, it maintains a reference link from the uploading data owner (DO) to the original copy of the file. Now the duplication is avoided, and it saves storage space in the cloud, which could impact the economy of the business [5]. Deduplication is supported by the convergent encryption technique, which enables the cloud to store single copy data in the cloud [6]. The main objective of the paper is to design and propose a framework for effective management of cloud storage and to eliminate the duplicate copy of data in cloud storage. To provide the proper deduplication mechanism, it is necessary to propose a Token and Tag generation method for verifying duplication of key and data in the cloud storage.

Related Work

Periasamy et al. [7] presented an enhanced secure content de-duplication identification and prevention (ESCDIP) approach to improve the file-level and content-level de-duplication discovery of coded data with consistency in the cloud environment. Every cloud user's files comprise an independent master key for encryption using the ESCDIP technique and outsourcing them into the cloud. It shrinks the overheads that are connected with the collaborative duplication detection and query processes. The method identifies the unique data chunking to store in the cloud. The cloud user sets the size of the data chunk. Chunk is also denoted as segments. Each segment is assigned with an identifier. The method identifies the duplicated data through a comparison of the data segment identification, where only a copy of every repeated part will be stored. At any cost, duplicated segments cannot be saved, and pointers are designed for them. The technique improves storage efficiency and reduces backup costs. Based on the result, this method deduced data uploading and downloading time and minimize communication cost compared with other methods.

Gayathri Devi et al. [8] concentrated on the result to overcome the difficulties initiated due to substantially distributed pieces of data. Fragmentation can happen in the form of sparse containers or containers that are not in order. Refurbish speed, and garbage collection efficiency are compromised due to these containers. The disordered container triggers a debility in renovate speed owing to the reduction in restoring cache. The idea of weakening destruction is showcased by the proposal of the History-Aware Rewriting (HAR) algorithm. HAR practices some of the past material of the backups that have happened to make out and reduce thin containers. The hash code generation algorithm gives each of the chunks a unique hash code; for example, Message Digest 5 (MD5). The logical chunk address is used to combine all the chunks and attain the particular individual file. The encryption algorithm preferred to use is the Data Encryption Standard (DES) to produce a crucial top-secret file is given to the requested user when the data owner creates the user. Collectively using the algorithms mentioned above, the proposed system aims to minimize fragmentation problems for the in-line deduplication system with a backup storage. The volume of duplicate data will determine the quantity of the development of refurbishing performance.

Hemanth Chandra et al. [9] proposed the POD vs. IDedup deduplication technique. The POD is a performance-oriented Deduplication scheme that can improve the performance of primary storage systems in the Cloud by forcing data Deduplication on the I/O path to remove redundant write requests while also saving storage space. POD minimizes the data fragmentation problem by its process of considers a request-based selective Deduplication approach (Select-Dedupe) to Deduplicating the I/O redundancy on the critical I/O path. In the meanwhile, intelligent cache management (iCache) is employed in POD to improve read performance further and increase space-saving. The paper evaluation shows that POD significantly enhances the performance and save the capacity of primary storage systems in the Cloud. Latency-sensitive and primary storage workloads are focused on inline deduplication system, and the latest techniques like POD are compared. Challenges of Latency sensitive workloads and Inline Deduplication, large disk space needed for reading when implements fragmentation.

Suzhen Wua et al. [10] proposed a Deduplication-Assisted primary storage system in Cloud-of-Clouds (short for DAC). DAC removes the redundant data chunks in the cloud storage location and distributes the data among several autonomous cloud storage providers by manipulating the data reference characteristics. In DAC, the data chunks are deposited in various providers by examining the repetition and eradication code patterns. To better development, the benefits of both imitation and eradication code patterns and exploit the reference characteristics in data deduplication, the great referenced data chunks are stored with the replication scheme while the other data chunks are stored with the erasure code scheme. The experiments conducted on a lightweight prototype implementation show that DAC improves the performance and cost-efficiency significantly, compared with the existing methods.

Frederik Armknecht et al. [11] proposed a ClearBox, which enables a cloud provider to transparently attest to its client's deduplication patterns of their stored data. ClearBox additionally enforces fine-grained access control over deduplicated files, supports data confidentiality, and resists against malicious users. Evaluation results derived from a prototype implementation of ClearBox show that the system scales correctly based on the number of files in the system for the number of users. It is a full-fledge proposed system to verify the saving in cloud storage by the user at any time. ClearBox motivates a novel cloud pricing model, which promises a fairer allocation of storage costs amongst users—without compromising any data confidentiality or system performance. It is believed that such a model provides strong incentives for users to store accessible data in the cloud (since big data will be cheaper to store) and discourages the upload of personal and unique content.

Wen Xia et al. [12] presented an idea named SiLo with scalable deduplication, which efficiently achieves the location of data streams to duplicate removal, throughput, and well stable load at extremely low RAM overhead. The new system exposed and exploited more resemblance by grouping toughly connected small files into a segment and also segmenting large files. The system is to force the data stream location by consolidating the attached parts into blocks to capture similar and duplicate data lost. SiLo parallelizes and distributes the data chunks to multiple backend servers by implementing a locality-based stateless routing algorithm. SiLo reduces RAM utilization for index-lookup because of carefully improving resemblance through the exploitation of locality and vice versa. It also achieves the near-exact efficiency of duplicate elimination and maintains a high deduplication throughput, and obtains load balance among backup nodes.

Fatema Rashid et al. [13] proposed a new privacy-preserving framework for deduplication that addresses data privacy and security issue. The framework uses an efficient deduplication algorithm to divide a given file into smaller units. These units are then encrypted by the user using the combination of a secure hash function and a block encryption algorithm. An index tree of hash values of these units is also generated and encrypted using an asymmetric search encryption scheme by the user. This index tree is enabled the cloud service provider to search through the index and return the requested units. The framework allows CSPs to use proposed techniques, and they cannot access either the users' plaintexts or the users' decryption keys.

Methodology

The proposed framework is worked based on the sequence of steps. The framework consists of three primary cloud services, Key generation, and Token maintenance as a Service (KTaaS), Convergent Encryption as a Service (CEaaS), and Cloud Storage as a Service (CSaaS). This section defines the steps to be followed to avoid the data duplications in cloud storage. Initially, the data uploading request is created by the user for a specific data. First, a token is generated for the data, and it is verified for the duplication of Token in the KTaaS. Second, a key is generated for convergent encryption. This key generated based on the data of the user. It is an irreversible hashing approach to generate the convergent encryption key. Third, data are encrypted using a convergent encryption technique and with the convergent encryption key. Fourth, after completion of data encryption, a tag is generated from the encrypted data. The created tag is verified for duplication of data in the cloud storage. If it is not available, then the encrypted data is upload to the cloud storage. Otherwise, a link is generated for the specific data and forward to the user for further access concerning the authorization of the user. Figure 1 represents the methodological diagram of the proposed framework.

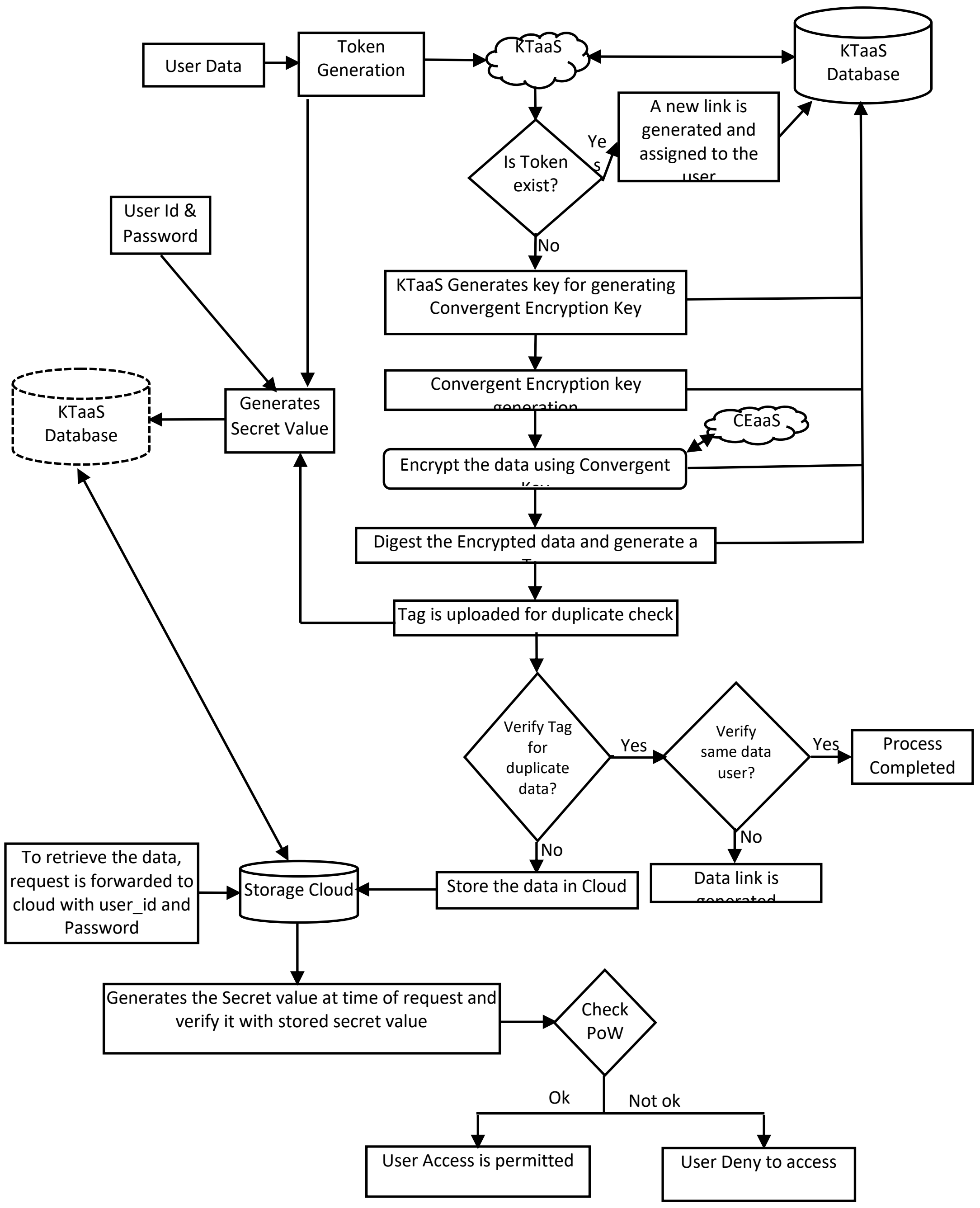
User Data
Token Generation
KTaaS
KTaaS Database
Is Token exist?
Yes
A new link is generated and assigned to the user
No
User Id & Password
KTaaS Generates key for generating Convergent Encryption Key
Convergent Encryption key generation
KTaaS Database
Generates Secret Value
CEaaS
Encrypt the data using Convergent Key
Digest the Encrypted data and generate a Tag
Tag is uploaded for duplicate check
Verify Tag for duplicate data?
Yes
Verify same data user?
Yes
Process Completed
No
No
To retrieve the data, request is forwarded to cloud with user_id and Password
Storage Cloud
Store the data in Cloud
Data link is generated
Generates the Secret value at time of request and verify it with stored secret value
Check PoW
Ok
Not ok
User Access is permitted
User Deny to access

Fig. 1 Methodological diagram of Proposed Framework

If data is already available in the cloud, then the same data is not allowed to store in the cloud by any other user. If a user tries to upload the data with the same content which is already stored in the cloud, then a link is generated for the data and assigned to the user instead to save another duplicate copy of the same content.

Proposed DedupFrame System

The proposed framework consists of three cloud services for key and token generation and maintenance, convergent encryption, cloud storage. The user's data should be securely stored in the cloud. Deduplication supports to avoid redundant copies of the same data stored in the cloud. This thesis proposes a scheme with the introduction of KTaaS and CEaaS. KTaaS is the trusted cloud service for generating a key for generating convergent encryption keys and tokens are maintained in this cloud service. CEaaS is an independent, trusted cloud service provider for convergent encryption. It uses a key that is generated from the data. The data is digested, and a hash value is generated from the data, is known as encrypted data. Users should request this service when they are ready to upload data to the cloud. Once the data is encrypted, then a tag is generated from the encrypted data. Now, for duplicate checking, the tag is verified with already stored tags in the KTaaS. If the tag is not available, then the data is uploaded to the cloud. Otherwise, based on the user authentication, a link is created and assigned to the user. Figure 2 shows the proposed system framework with different cloud services.

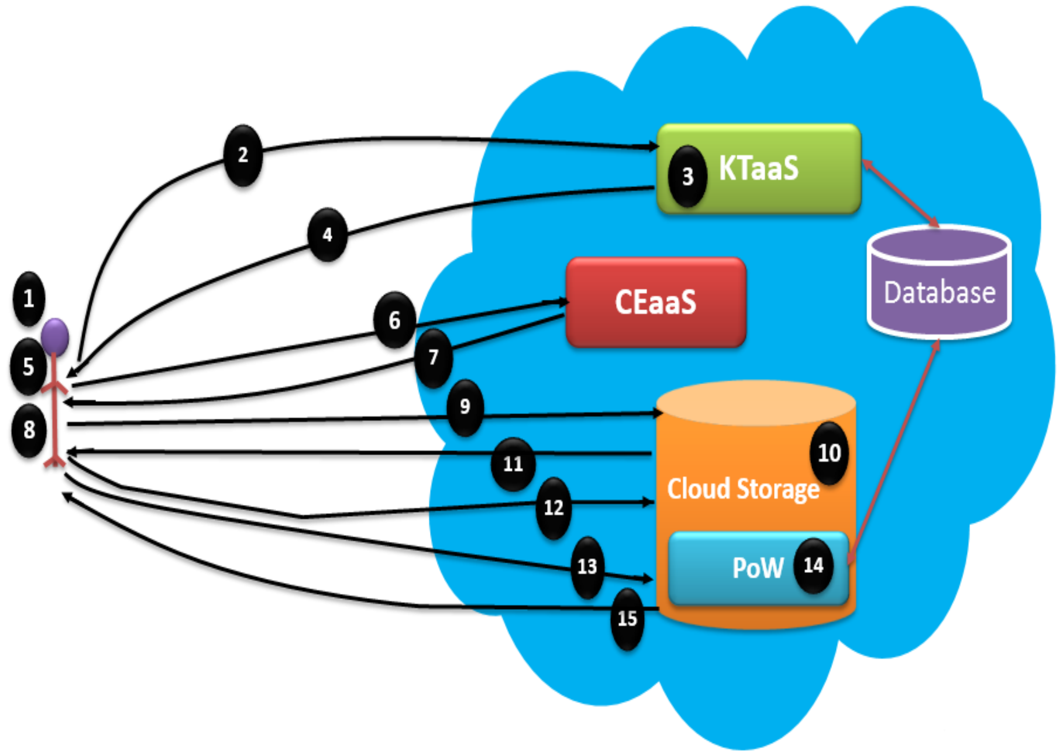

Fig. 2 Proposed System Framework with different Cloud services

The three services in the proposed framework are independent can be provided by three different cloud service providers. Each user wants to register with the system using their details, and the user name and password are assigned for each user. Authenticated users only can access the system. The working procedure of the proposed approach is explained in the following steps.

Steps involved in the proposed framework process:-

1. Initially, users want to upload data (U_D) to the cloud; they must generate a token (T_{KN}) from the user's data using T_{KN}= Tkn_Gen(U_D) primitive function.
2. The token T_{KN} is forwarded to the KTaaS through a secure channel to get a key for generating a convergent encryption key (CE_K).
3. KTaaS verify the metadata whether T_{KN} already exists in the database or not, if it already exists in the database, then, KTaaS forward the corresponding key previously generated for the same T_{KN} to the user. If the T_{KN} does not exist in the metadata, then, KTaaS creates a key (G_KCE_K) for T_{KN} send to the user. KTaaS maintain metadata for each user T_{KN} and its corresponding key details.
4. KTaaS forward the corresponding key G_KCE_K to the user for generating convergent encryption key CE_K.
5. Users generate the convergent encryption key using the proposed technique CE_K = CEkey_Gen(G_KCE_K) with the key G_KCE_K received from the KTaaS.
6. After the key generation, U_D is ready to encrypt, a user requesting convergent encryption from the CEaaS.
7. CEaaS provides environment for convergent encryption techniques E_D=CEnc_Alg().
8. Users encrypt the data using CE_K. After encryption, a Tag (T_G) is generated from the encrypted data (E_D) using T_G = Tag_Gen(E_D) for verifying the duplication of data in the cloud.
9. Users forward the T_G to the cloud to check whether the data is already stored in the cloud or not.
10. Cloud Storage providers verify the $T_{G,}$ which generated at the time of uploading with T_G' which is taken from the cloud if it is available in the cloud. If T_G==T_G's, then data is already in the cloud storage. Now a link is generated for this user for the same data.
11. If T_G != T_G' then the cloud system asked the user to forward the data E_D.
12. Users forward the E_D to CS, now deduplication is verified, and only one copy of data is stored in the CS.
13. Uploaded data can be downloaded by verifying the proof of ownership in cloud storage. Users have to submit their authentication details.
14. Cloud storage has a verifier to verify that the ownership of the user corresponds to the data.
15. If the ownership of the user is proved, then data can be accessed by the users.

The proposed deduplication framework consists of separate cloud services for different services. In the proposed procedure, initially, a token (T_{KN}) is generated using the proposed token generation method. After the generation of T_{KN}, a key (G_KCE_K) is generated from KTaaS for the generation of convergent encryption key (CE_K), the proposed convergent encryption key generation technique is described in [CE]. After the encryption, a tag (T_G) is generated from the encrypted data. Now the T_G is verified for duplication. At the success of the T_G verification, the data either stored in the cloud or a link is generated for a new user when he/she tries to upload the same content of data. The correspondence of data and other generated values for uploading a file are shown below,

$$U_D \rightarrow T_{KN} \rightarrow G_K CE_K \rightarrow CE_K \rightarrow E_D \rightarrow T_G \qquad \ldots (1)$$

The formula shows the relations and correspondence of each data with a key and other generated values produce for each file going to store in the cloud storage. The file is uploaded to the cloud without duplication. The procedures for Token generation T_{KN}, $G_K CE_{K,}$ and Tag generation T_G are described in the following sections.

After uploading the data, it can be downloaded based on the proof verification of the user. By confirming the ownership of the data, the data can be accessed by the user. The file is accessed based on the PoW. The PVAuth verifier authenticates the data owner using the challenge-response method.

For every file of data stored in the cloud, a secret value is assigned to each data file. The secret value S_{VAL} is generated by using Token, Tag of the file, user id, and password of the user. Token and tag are involved in identifying the data, and user id and password are included for determining the authorized user. This secret value is stored along with other details of the data. If a user tries to access the data, the user submits their credential and mention that on which file they need to access.

$$S_{VAL} \leftarrow E[T_{KN} || T_G || Uid || Pwd] \qquad \ldots(2)$$

S_{VAL}' is stored value in the database at the time of uploading the data, where the proof is verified by S_{VAL}==S_{VAL}'.

The following sections describe Token generation, Key generation for generating $CE_{K,}$ and Tag generation.

Token Generation

The proposed framework is described with the generation of Token, Tag, and Key. This section explains how a Token is generated for the data which is uploaded to the cloud. The token is generated for verifying the duplication of the key in the KTaaS. After a generation of a token, it is mapped with the data in the KTaaS. Steps involved in the proposed token production are as follows.

1. User data are converted into binaries
2. Find the size of the binary block. Check the total binary size is equal to the multiplied value of 32. If not then add desired zero with user data
3. Divide the block into two and find the XOR of both blocks, repeat the step until the total bits come to 32 bits.
4. Convert the bits into character code

5. The character code is taken as the token for the users' data

Experiment with Sample Data

Users' data (U_D) = A cat has nine lives

Knowledge is power

Step 1→ Find of the size of U_D in N

N← sizeof(U_D)

Step 2→ U_D is converted into ASCII decimal code

A←65 c←99 a←97 t←116 h←104a←97 s←115 n←110i←105 n←110e←101l←108
i←105 v←118e←101s←115 K←75 n←110o←111w←119 l←108 e←101d←100
g←103 e←101i←105 s←115 p←112 o←111 w←119 e←101r←114

Step 3→Each ASCII decimal code is transformed into its equivalent binary code

65←01000001	99←01100011	97← 01100001	116←01110100	104←01101000
97←01100001	115←01110011	110←01101110	105←01101001	110←01101110
101←01100101	108←01101100	105←01101001	118←01110110	101←01100101
115←01110011	75←01001011	110←01101110	111←01101111	119←01110111
108←01101100	101←01100101	100←01100100	103←01100111	101←01100101
105←01101001	115←01110011	112←01110000	111←01101111	119←01110111
101←01100101	114←01110010			

Buff←01000001 01100011 01100001 01110100 01101000 01100001 01110011 01101110 01101001 01101110 01100101 01101100 01101001 01110110 01100101 01110011 01001011 01101110 01101111 01110111 01101100 01100101 01100100 01100111 01100101 01101001 01110011 01110000 01101111 01110111 01100101 01110010

Step 4→Divide the block into two and find the XOR of both blocks, repeat step until the total bits will come to 32 bits.

BLK_1←01000001 01100011 01100001 01110100 01101000 01100001 01110011 01101110 01101001 01101110 01100101 01101100 01101001 01110110 01100101 01110011

BLK_2←01001011 01101110 01101111 01110111 01101100 01100101 01100100 01100111 01100101 01101001 01110011 01110000 01101111 01110111 01100101 01110010

Find XOR,

01000001 ⊕	01100011 ⊕	01100001 ⊕	01110100 ⊕
01001011	01101110	01101111	01110111
~~00001010~~	00001101	00001110	00000011
01101000 ⊕	01100001 ⊕	01110011 ⊕	01101110 ⊕
01101100	01100101	01100100	01100111
~~00000100~~	00000100	00010111	00001001
01101001 ⊕	01101110 ⊕	01100101 ⊕	01101100 ⊕

01100101 01101001 01110011 01110000

00001100 00000111 00010110 00011100

01101001⊕ 01110110 ⊕ 01100101 ⊕ 01110011 ⊕

01101111 01110111 01100101 01110010

~~00000110~~ 00000001 00000000 00000001

Round2 →00001010 00001101 00001110 00000011 00000100
00000100 00010111 00001001 00001100 00000111
00010110 00011100 00000110 00000001 00000000
00000001

Repeat the process by divide the block into two halves and find XOR.

BLK1←00001010 00001101 00001110 00000011 00000100
00000100 00010111 00001001

BLK$_2$←00001100 00000111 00010110 00011100 00000110
00000001 00000000 00000001

00001010 ⊕ 00001101 ⊕ 00001110 ⊕ 00000011 ⊕

00001100 00000111 00010110 00011100

~~00000110~~ ~~00001010~~ ~~00011000~~ ~~00011111~~

00000100 ⊕ 00000100 ⊕ 00010111 ⊕ 00001001 ⊕

00000110 00000001 00000000 00000001

~~00000010~~ ~~00000101~~ ~~00010111~~ ~~00001000~~

Round3→ 00000110 00001010 00011000 00011111 00000010
00000101 00010111 00001000

BLK$_1$←00000110 00001010 00011000 00011111

BLK$_2$←00000010 00000101 00010111 00001000

00000110 ⊕ 00001010 ⊕ 00011000 ⊕ 00011111 ⊕

00000010 00000101 00010111 00001000

00000100 00001111 00001111 00010111

Block
←00000100 00001111 00001111 00010111

Step5→Convert the bits into decimal code

4←00000100 15←0000111115←0000111123←00010111

Step6→Convert the decimal into ASCII Character Code

Token T$_{KN}$← ♦☼☼↨

Key Generation for CE_K

The key generation is the process of finding a secret value for generating secret data or transforming the data. This section describes a key generation approach. This key generation is used to create a key G_KCE_K. This key is used to produce the convergent encryption key, which used to encrypt the data using convergent encryption. It is a 128 bits key. It is generated in the KTaaS if the token does not match with KTaaS database. Each G_KCE_K is corresponding with the token T_{KN}, and each T_{KN} corresponds to a specific user's data.

Steps involved in the generation of G_KCE_K:-

1. Keys are generated using the Random number generation.
2. This key is 128 bits key, 16 random values are generated
3. The 16 random decimal values are randomly chosen between 32 to 126
4. The 16 decimal values are converted into ASCII character code
5. The 16 character code is represented as GCE_K, key for generating CE_K

This procedure is used to generate a key (G_KCE_K); this key is used to create the convergent encryption key. G_KCE_K is a randomly generated key. It generates 16 random numbers between 32 to 126 printable ASCII decimal value. This 16 character is considered as the 128 bits key for generating convergent encryption key CE_K.

Tag Generation from E_D

The data are encrypted using CE_K, and a tag is generated from the encrypted data E_D. The generation follows the below step to generate the tag T_G from E_D.

Steps involved in the generation if Tag:

1. Encrypted data is considered as input for tag generation
2. All encrypted codes are converted into corresponding decimal and converted into binary values.
3. Find the size of the total bits.
4. Consider a single bit for 0 or 1 when they have occurred continuously for two times.
5. Divide the total block bits into two blocks
6. Consider 8 bits values of each block and find XOR of both blocks of each 8 bits
7. Repeat step 5 and 6 until whole bits comes under or equal to 64 bits.
8. The result of 8bits blocks are converted into ASCII character code
9. Final Character code derived from step 8 is a tag for the encrypted data.

Experiment with sample data

Tag is generated after the data are encrypted using the proposed convergent encryption. The tag generation is based on encrypted data. The encrypted data is taken as input for tag generation. The below step describes the tag generation for a sample encrypted data.

Consider the ciphertext,

E_D**🡐.↑ñb▲♫■ órL¿╛ xp╜ ^ ╙∞╥>╤ë╕ Æ╥♦Éµêp\**

Step 1→ Find of the size of E_D in N

N← sizeof(E_D)

Step 2→ E_D is converted in to ASCII decimal code

46 24 164 98 30 14 254 32 162 114 76 168 190 120 112 188 94 200 236 210 62 209 137 184 146 210 4 144 230 136 112 92

Step 3→ Decimal E_D is converted in to binaries

46←00101110 24←00011000 164←10100100 98←01100010 30←00011110 14←00001110
254←11111110 32←00100000 162←10100010 114←01110010 76←01001100
168←10101000 190←10111110 120←01111000 112←00001110 188←10111100
94←01111010 200←11001000 236←11101100 210←11010010 62←00111110
209←11010001 137←10001001 184←10111000 146←10010010 210←11010010
4←00000100 144←10010000 230←11100110 136←10001000 112←01110000 92←01011100

Buff←00101110000110001010010001100010000111100000111011111110001000001010001001110010010011001010100010111110011110000000111010111100011110101100100011101100110100100011111011010001100010011011100010010010110100100000100100100001110011010001000011100000101110

Step 4→ consider 1 bit of 0 or 1 when 0 or 1 is occurred continuously for two times.

Buff←01011001001010100100100110001101111001000101001011010101010101001011100110011001011001011010100110101010100111010100100101011001010101010100001010100110101001001100 0101

Step 5→ Divide the total block bits into two blocks

Blk1→01011001001010100100100110001101111001000101001011010101010101001011100110011001 0110

Blk2→01011010100110101010100111010100100101011001010101010100001010100110101001001100 0101

Step 6→Consider 8bit values of each block and find XOR of both block of each 8bits

XOR Operation→Round 1

$$\begin{array}{r} 01011001\oplus \\ \underline{01011010} \\ \underline{00000011} \end{array} \quad \begin{array}{r} 00101010\oplus \\ \underline{10011010} \\ \underline{10110000} \end{array} \quad \begin{array}{r} 01001001\oplus \\ \underline{10101001} \\ \underline{11100000} \end{array} \quad \begin{array}{r} 10001101\oplus \\ \underline{11010100} \\ \underline{01011001} \end{array}$$

$$\begin{array}{r} 11100100\oplus \\ \underline{10010101} \\ \underline{01110001} \end{array} \quad \begin{array}{r} 01010010\oplus \\ \underline{10010101} \\ \underline{11000111} \end{array} \quad \begin{array}{r} 11010101\oplus \\ \underline{01010100} \\ \underline{10000001} \end{array} \quad \begin{array}{r} 01010100\oplus \\ \underline{00101010} \\ \underline{01111110} \end{array}$$

$$\begin{array}{r} 10111001\oplus \\ \underline{01101010} \\ \underline{11010011} \end{array} \quad \begin{array}{r} 10011001\oplus \\ \underline{01001100} \\ \underline{11010101} \end{array} \quad \begin{array}{r} 0110\oplus \\ \underline{0101} \\ \underline{0011} \end{array}$$

If total bits length greater than 64 bits then repeat the step 5 and 6

0000001110110000111000000101100101110001110001111000000101111110110100111
10101010011

XOR Operation→Round 2

Divide the total block bits into two blocks

```
 00000011      10110000      11100000
 00011110      00000101      11111011
 00011101      10110101      00011011

 01011001      01110001      00000011
 01001111      01010100      00000011
 00010110      00100101      00000000
```

Blk← 00011101 10110101 00011011 00010110 00100101 00000000

Step 7→ The result of 8bits blocks are converted into ASCII character code

00011101←2910110101←181 00011011←2700010110←22
00100101←3700000000←0

Step 8→Final Character code derived from step 6 is a tag for the encrypted data.

29←↔181←╡ 27←← 22←▬37←%
0←~~[NUL]~~

Tag T_G ← ↔ ╡ ← ▬ % ~~[NUL]~~

Implementation Setup and Results

The proposed research work is implemented in the cloud environment. The implementation setup has a cloud server from Microsoft Azure. The configuration of the server is as follows, Windows server 2008 micro instance with 30GB storage and 1GB RAM. The cloud-based application is developed and hosted on the cloud server. The proposed application is developed in C#.NET using visual studio 2012. The hosted cloud application is served as a cloud service which consumed by all procedures proposed in the research work. Figure 3 shows the diagrammatical representation of the implementation environment created for research work.

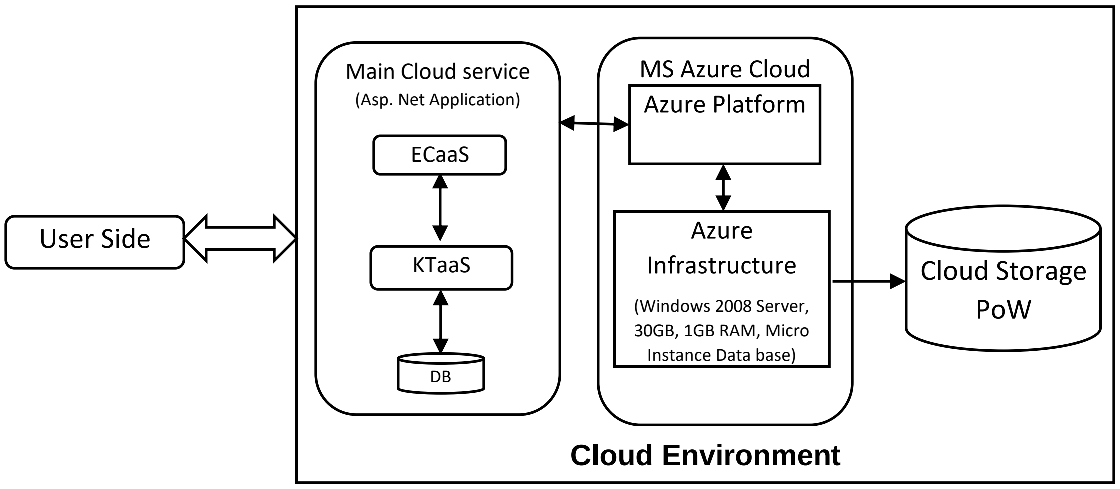

Fig. 3 Implementation Environment

Convergent encryption, Key generation and token management, convergent encryption key generation and tag generation, and proof of ownership all these procedures are running in the developed cloud-based application. For storing data, the Azure cloud server is used. From the implementation, it shows that if the same content of the file is uploaded for the second time, then it gives an error message that the file is already uploaded and please try another file. At the same time, the data is not uploaded to the cloud storage. It proves that the proposed framework is not allowed to upload duplicate data. Hence, it enables effective management of data in the cloud storage.

The efficiency of the proposed approach is measured by considering the computational time caused by the proposed and existing methods. It can be calculated by the time taken by the deduplication techniques for uploading data to the cloud. Calculation includes the processing of all steps describes in the framework. A comparison of computational time considers the different sizes of data.

Table 1 and figure 4 shows the result from the comparison of proposed and existing approaches concerning computational time. It illustrates that the proposed method has taken a minimum computational time compared to other existing plans.

Table 1 Computational time caused by the Proposed and Existing Deduplication Techniques

Size	**MLE**	**Dekey**	**Dupless**	**Dedup**
5MB	143	173	148	130
10MB	268	301	275	250
15MB	391	427	401	377
20MB	512	546	519	498
25MB	601	639	613	589

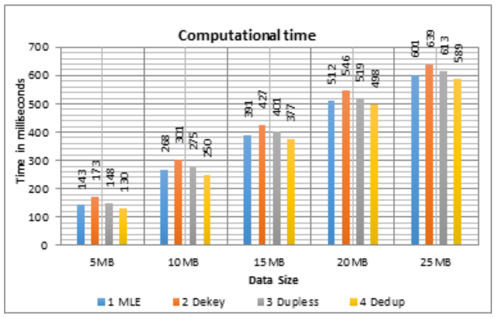

Fig. 4 Computational time caused by the Proposed and Existing Deduplication Techniques

Table 2 and figure 5 shows the cloud storage allocation with deduplication and without deduplication.

Table 2 Data storage allocation concerning with and without deduplication when two users upload the same data file

Size	**With Dedup (MB)**	**Without Dedup (MB)**
5MB	5	10
10MB	10	20
15MB	15	30
20MB	20	40
25MB	25	50

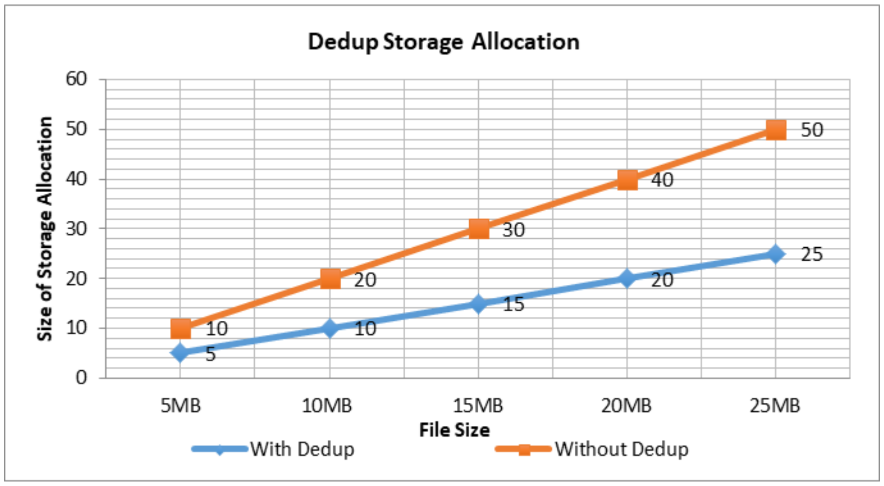

Fig. 5 Data storage allocation concerning with and without deduplication when two users upload the same data file

If two users try to upload the same data file, how the cloud storage allocates the space in the storage? By concerning with or without deduplication. The comparison shows that without deduplication, allocates separate storage for each user's file when the user uploads the data. In the case of deduplication, it allocates only a single storage allocation for all user's data with the same content in a file. It is observed that the proposed dedup framework can be more effective in avoiding duplicate data storage allocation and managing cloud storage.

Conclusion

The aim and objective of the research work are to eliminate duplicate data in the cloud storage before it is uploaded to the cloud. Elimination of duplicate data effectively helps to manage cloud storage. The framework follows a sequence of steps for token generation, key for CE_K generation, convergent encryption key generation, convergent encryption, tag generation, and proof of ownership verification. The overall research work mainly concentrates on proposing the generation of a convergent encryption key and convergent encryption and proof of ownership. All the proposals are incorporated in a framework and maintain effective cloud storage.

Along with the above proposals, a Token and Tag generation method is also proposed in this paper. Token and tag are generated from the user's data file. The token is used to verify the duplication of keys in the KTaaS. The tag is used to verify the duplicate files in the cloud storage. All these procedures are for uploading the data to the cloud storage without duplication. The framework also describes the procedure to download the file from the cloud. The user's file is downloaded or accessed based on the proof verification of the user's authentication. The framework allows only the verified users to access the files.

The proposed research work is implemented in the cloud environment. A cloud-based application is developed and hosted in the Windows Azure platform service. The developed application runs for token generation, key generation, and convergent encryption and proof verification. From the implementation results, it is proved that research work is implemented in the cloud environment and also showed how it eliminates the duplication in cloud storage.

Reference

[1] MS.C.Kamatchi, R.Pooja, S.Serishma, R.Vanitha, Data Deduplication Security with Dynamic Ownership Management, International Journal of Computer Science Trends and Technology (IJCST) – Volume 5 Issue 2, Mar – Apr 2017, pp.252-256.

[2] D. T. Meyer and W. J. Bolosky, "A Study of Practical Deduplication", Trans Storage, vol. 7, no. 4, pp. 14:1–14:20, 2012.

[3] Nishant N. Pachpor and Prakash S. Prasad Securing the Data Deduplication to Improve the Performance of Systems in the Cloud Infrastructure, Performance Management of Integrated Systems and its Applications in Software Engineering, Asset Analytics, springer, 2020, pp.43-58

[4] Yukun Zhou, Dan Feng, Wen Xia, Min Fu, and Yu Xiao, DARM: A Deduplication-Aware Redundancy Management Approach for Reliable-Enhanced Storage Systems, Springer Nature Switzerland, ICA3PP, LNCS 11335, 2018, pp. 445–461.

[5] Sanjeet Kumar Nayak1 · Somanath Tripathy1, SEDS: secure and efficient server-aided data deduplication scheme for cloud storage, International Journal of Information Security, Springer, 2019, pp. 1-12

[6] K.Kanimozhi, N.Revathi, Secure Deduplication on Hybrid Cloud Storage with Key Management, International Research Journal of Engineering and Technology, Volume 03, Issue 06, 2016, pp. 2267- 2271.

[7] J. K. Periasamy and B. Latha, An enhanced secure content de-duplication identification and prevention (ESCDIP) algorithm in cloud environment, January 2019Springer-Verlag London Ltd., 2019, PP. 1-10.

[8] K. Gayathri Devi, S. Raksha and Kavitha Sooda, Enhancing Restore Speed of In-line Deduplication Cloud-Based Backup, Systems by Minimizing Fragmentation, Smart Intelligent Computing and Applications, Springer Nature Singapore Pte Ltd, 2020, pp. 9-21.

[9] Hemanth Chandra N, Sahana D. Gowda, Secure and Efficient Client and Server Side Data Deduplication to Reduce Storage in Remote Cloud Computing Systems , International e-Journal For Technology And Research, Volume 1, Issue 5, May 2017, pp 1-8.

[10] Suzhen Wua, Kuan-Ching Li c, Bo Maob and Minghong Liao, DAC: Improving storage availability with Deduplication-Assisted Cloud-of-Clouds, Elsevier Future Generation Computer Systems, Volume 74, September 2017, pp. 190-198,

[11] Frederik Armknecht, Jens-Matthias Bohli, Ghassan O. Karame, and Franck Youssef, Transparent Data Deduplication in the Cloud, ACM Conference on Computer and Communications Security, ISBN: 978-1-4503-3832-5, 2015, pp. 886-900

[12] Wen Xia, Hong Jiang, Dan Feng and Yu Hua, Similarity and Locality Based Indexing for High Performance Data Deduplication, IEEE Transactions on Computers, Vol. 64, No. 4, April 2015, pp. 1162-1176.

[13] Fatema Rashid, Ali Miri, Isaac Woungang, A Secure Data Deduplication Framework for Cloud Environments, IEEE International Conference on Privacy, Security and Trust, ISBN: 978-1-4673-2326-0/12, 2012, pp. 81-87.