An Improved Blockchain-Based Secure Data Deduplication using Attribute-Based Role Key Generation with Efficient Cryptographic Methods

Ruba S (ruba3112@gmail.com)
GCE: Government College of Engineering Salem
A.M. Kalpana
GCE: Government College of Engineering Salem

Research Article

Keywords: Secure data deduplication, cloud storage, blockchain, Role key generation, SHA-256, MLE

Posted Date: June 18th, 2021

DOI: https://doi.org/10.21203/rs.3.rs-596633/v1

An Improved Blockchain-Based Secure Data Deduplication using Attribute-Based Role Key Generation with Efficient Cryptographic Methods

Mrs. S.Ruba[1,*], Dr. A.M.Kalpana[2]

[1]Assistant Professor, Department Computer Science and Engineering, Government College of Engineering, Salem - 636 011.

[*]Corresponding author mail id: ruba3112@gmail.com

[2]Professor/Department of Computer Science and Engineering, Government College of Engineering, Salem - 11.

kalpana.gce@gmail.com

Abstract

Deduplication can be used as a data redundancy removal method that has been constructed to save system storage resources through redundant data reduction in cloud storage. Now a day, deduplication techniques are increasingly exploited to cloud data centers with the growth of cloud computing techniques. Therefore, many deduplication methods were presented by many researchers to eliminate redundant data in cloud storage. For secure deduplication, previous works typically have introduced third-party auditors for the data integrity verification, but it may be suffered from data leak by the third-party auditors. And also the customary methods could not face more difficulties in big data deduplication to correctly consider the two conflicting aims of high duplicate elimination ratio and deduplication throughput. In this paper, an improved blockchain-based secure data deduplication is presented with efficient cryptographic methods to save cloud storage securely. In the proposed method, an attribute-based role key generation (ARKG) method is constructed in a hierarchical tree manner to generate a role key when the data owners upload their data to cloud service provider (CSP) and to allow authorized users to download the data. In our system, the smart contract (agreement between the data owner and CSP) is done using SHA-256 (Secure Hash Algorithm-256) to generate a tamper-proofing ledger for data integrity, in which data is protected from illegal

modifications, and duplication detection is executed through hash-tag that can be formed by SHA-256. Message Locked encryption (MLE) is employed to encrypt data for data uploading by the data owners to the CSP. The experimental results show that our proposed secure deduplication scheme can give higher throughput and a low duplicate elimination ratio.

Keywords: Secure data deduplication, cloud storage, blockchain, Role key generation, SHA-256, MLE

.

1. Introduction

At present, cloud computing has been employed to store, manage and process the data as alternating storage for data backup with limited local storage space. For data storage, cloud storage is utilized by several big companies. Cloud storage might be suffered from several difficulties such as security issues, storage space issues, etc, due to a huge quantity of backup storage constraints. In cloud storage services, one challenge is the vertical amount of duplicate data. Particularly, there could be almost 68% duplicate data is obtainable on normal file systems, and up to 90-95% duplicate data is available on backup applications. As a result, issues of storage space are managed through the data deduplication process, in which the storage space effectiveness can be enhanced. In the deduplication techniques, the cloud is allowed to store only one copy of duplicate data, and links are given to that copy when required. The deduplication is mostly done through hash values (fingerprints) for the files and chunks representation wherever these values can be compared with others to establish if the chunk or file they symbolize will be a duplicate or not. The fingerprints are stored in an index structure that can be too large to fit in memory. Therefore, it will be stored in an on-disk structure that will direct to a popular difficulty called disk index lookup bottleneck.

In general, the data deduplication process is done on the block level or file level. In the block-level data deduplication, the files are segmented into blocks and only a copy of every block is stored. Block-level deduplication can thus eliminate duplicate data blocks in non-

identical files and it is used to improve storing efficiency more compared to file-level data deduplication. Thus, we focused on the block-level data deduplication in this paper. However, data deduplication can direct the tension between data confidentiality and storage efficiency. Mostly, cloud users have tension about the confidentiality of their sensitive data hosted by cloud servers that could not be trusted. A simple idea to data confidentiality consideration can be to let every cloud data owner outsource encrypted data to the cloud server that does not recognize the decryption key. In the traditional encryption process, every user needs to employ a different key to encrypt the data. Consequently, the similar data at different users can appear as completely different ciphertexts. In the data deduplication methods, the cloud server needs to recognize the same data. Thus, natural conflict may have happened between data confidentiality and storage efficiency. As a result, improved data deduplication is presented in this paper with a computing environment for efficient storage in a more secured manner.

Now a day, the blockchain-based security method is mostly focused on a distributed computing paradigm for its low cost, high efficiency, high credibility, and high data security, and. It is a key technology that has been derived from the agreement system that can be an instance of a disseminated computing scheme with higher fault tolerance. In the blockchain technique, point-to-point transactions or collaboration are realized in disseminated systems with high traceability. For example, decentralized storage systems such as File coin have taken blockchain's benefits, in which security file storage could be achieved using an incentive mechanism and distributed structure. But none of these schemes have attempted to achieve the storage effectiveness that can be achievable through deduplication. In this paper, a cloud storage deduplication system is presented with high fault tolerance to overcome the issues of integrity, confidentiality, and reliability. For data integrity and system confidentiality, blockchain technology can be exploited. The communication frequency can also be reduced between the data owner and cloud servers, in which the risk of communications for monitoring and information leakage is reduced. In our proposed system, a smart contract is done with SHA-256, and data owner can encrypt their data by MLE scheme to make sure file confidentiality in the case of showing the files information on the blockchain. Moreover, the role key is generated according to the user's attributes hierarchically for data uploading to a cloud storage server, in which authorized users are only allowed to access the data.

The paper can be organized as follows: the related work is reviewed in section 2 based on the secure deduplication in cloud storage. In section 3, the presented secure duplication algorithm is discussed in detail. After that, the presented method is evaluated and compared to other methods applied to deduplication in section 4. The conclusion can be discussed briefly in section 5.

2. Related works

The Secure Deduplication and Virtual Auditing of Data have been presented in Cloud called (SDVADC) [2], in which integrity auditing and deduplication of information have been realized in the cloud. Secure deduplication of information was supported by this method and effective virtual auditing of the documents was done in the download process. However, a Duplicate chunk detection problem may have occurred. A novel method has been presented in [3] using Convergent and Modified Elliptic Curve Cryptography (MECC) algorithms over the cloud and fog environment for the secure deduplication system construction. The file encryption was done initially with the help of the Convergent Encryption (CE) technique and afterward, re-encryption was done using MECC. However, the data may be obtained by the opponent through the brute-force attack when the data belongs to a knowable set for CE. A Collaborative Edge (CE) network combined is presented in [4] with the CE-D2D framework for solving maximizing video caching with efficient cellular and bandwidths. The proposed system has been using a strategy by cached only the watching chunk videos instead of offloading and caching video by one edge node. The ZEUS (zero-knowledge deduplication response) system was presented in [5]. The authors have developed ZEUS and ZEUS+ as two privacy-aware deduplication protocols: in which weaker privacy guarantees were given by ZEUS as being more capable in the communication cost, as ZEUS guarantees stronger privacy properties, at an enhanced cost of communication. However, low deduplication throughput may have happened during the ZEUS process. A security evaluation model was presented in [6] that consisting of API (Application Programming Interface) from different clouds evaluating including video discovery, security recovery engine, scanning engine, quantifiable evaluation, etc. however, energy consumption may have increased during the process of measurable evaluation in various cloud clients. The RSA (Rivest–Shamir–Adleman) based Cross-Domain Secure Deduplication (RCDSD) of synchronization was presented in [7] between dispersed storage managers without enlightening a

lot of information about the real data stored by the customers. The synchronization part has been done through an interactive protocol of collaboration between the servers. But, computing overhead may have happened in the process of CSPs' distribution of duplication information to the users through the domain managers.

The investigation about backup storage setting enterprise of secure chunk-based deduplication method is done in [8]. The authors proposed a randomized key generation based on their inner works of backup service. Conversely, low chunking throughput can happen. Three stages are done in [9] for deduplication such as chunking, fingerprint, and indexing fingerprints. In chunking data files are divided into chunk boundary is decided by the divisor values. For each one there may be a unique value of identification that is computed through hash signature (i.e. MD-5 (Message Digit-5), SHA-1, SHA-256), which are called a fingerprint. Finally, fingerprints are stored as an index term to detect replication chunks by comparing the same fingerprints. However, in this method, high-cost computing, and medium-security is happened. Two secure data deduplication methods were proposed in [10] according to the Rabin fingerprinting in wireless sensing data of cloud computing. In the first system, deterministic tags and the other one adopts random tags were applied. But, low secure deduplication may have happened through Rabin fingerprinting. A secure data deduplication is focused on cloud storage [11]. In this work, the hash codes have been generated with the help of the Secure Hash Algorithm (SHA-512) [11]. According to this hash code-verifying system, authentication is provided. For secure Data Deduplication, cloud storage and service providers are permitted to use data with confidentiality. However, high key utilization is done by SHA-512 for deduplication and thus, low throughput can have happened.

3. Proposed Methodology

In this paper, an improved blockchain-based secure data uploading and downloading processes are suggested to more secure deduplication in a cloud storage system with high throughput and reduced deduplicate elimination ratio to save cloud storage given in Figure 1. The following methods are presented to the data uploading and downloading process to prevent data duplication.

Role key generation: The secret role key SK_r and public role parameter PUB_{rp} is generated according to the identity and policy role when the data owner uploads the data to Clod server for the data accessing by the authorized users only.

Duplicated data checking: the duplicated data can be checked through the block-chain process, in which agreement between CSP and data owner is done by SHA-256 method utilized for hash tag and hash table generation to check tag availability before the data uploading by the data owner. The cloud server can check whether the same tag exists in the hash table upon receiving data. Here, the cloud server responds to the data owner with "no duplicate" or "duplicate".

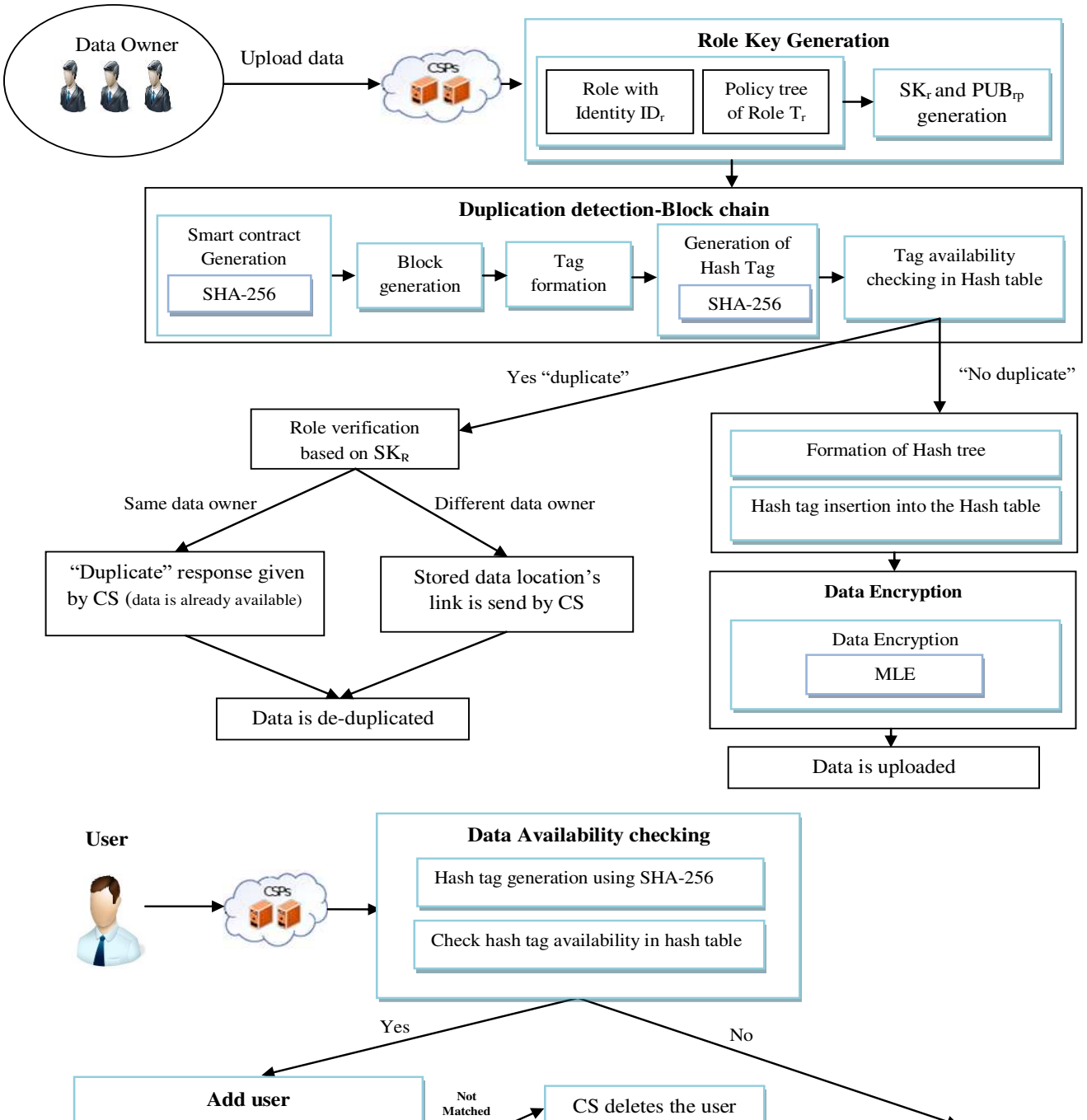

Figure 1 (a) Proposed Data Uploading to cloud server (CS), (b) Data downloading using the proposed scheme

upload. If the data owner gets "no duplicate" replied from cloud server, then data will be encrypted by applying the MLE and after that, encrypted data is uploaded to cloud server with generated secret role keys. If the same data is available in the different data owner storage space, cloud server sends the stored location link.

Secure Data downloading: For secure data downloading, the role is assigned by the cloud server to the user (data consumer) if the user's attributes can satisfy the role policy, in which the role policy is checked through generated identity role ID_R and role secret key SK_R. In this process, the data consumer can be added according to the policy role. Thus, added authorized users can only download the requested data from the cloud server.

The proposed phases are described in detail below:

a. Role Key generation

In this paper, attributed-based role key generation is done before the uploading of data to a cloud server. If the data owner wants to outsource some data to cloud storage, the data should be encrypted before being outsourced. Consequently, the data owner can be responsible to establish the access policy for each data, establish the roles, encrypting the data under the access policy, and master public and private keys can be generated. In this proposed methodology, a hierarchical tree is applied for the access policy structure representation. Here T is an access tree contains interior nodes that can be utilized as threshold gates such as AND/OR gate as the leaves are connected with attributes. A data consumer will be allocated to a role and as a result, ciphertexts will be decrypted corresponding to that role if and only if there can be an assignment of the attributes from the private key of data consumer to the tree's leaf nodes such that the tree will be satisfied. For an example of attributes, in the university domain, the staff may be in the

position of “Assistant professor,” “lecturer”. etc., the courses are in the form of “commerce,” “medicine”, “computer sciences”. The data owner is responsible for the following three processes:

Setup: In this process, the security parameter is taken as input and a master key and a group public key is given as output.

Role creation: In this process, role with identity ID_r and policy tree of role T_r are taken as inputs, and the secret key role SK_r is generated and role’s public parameters set PUB_{rp} is returned and after that, an empty user list will be provided, in which list users can be listed whose attributes matched with the role policy tree.

Data Encryption: In the proposed methodology, the encryption process executed by the data owner after the duplication-checking process using MLE and ciphertext C can be outsourced to the cloud.

Algorithm:

Input: Role with identity ID_r , role with policy tree T_r

Output: Secret key role SK_r , public role parameter PUB_{rp}

Step 1: To state a role with policy tree T_R over an attribute set ‘S’

Step 2: Assume th_a can be the node’s threshold value for every node $'a'$ in the hierarchical tree T_R

Step 3: If the non-leaf node contains a logical gate of “OR”

Then $th_a = 1$

If the non-leaf node contains a logical gate of “AND”

Then $th_a = Number\ of\ its\ children$

Step 4: Select polynomial as Y_a initiating from the tree’s root node A with its degree and is given by:

$$D_a = th_a - 1 \qquad (1)$$

Step 5:If the node $'A'$ can be the root

Then

Select a random value of tree $v \in Z_X^*$

Set $Y_a(0) = v$

Else

Set $Y_a(0) = Y_{X(a)}(index(a))$ (2)

where $X(a)$ denotes the node $a's$ parent node and unique index number will be assigned in $index(a)$ to every node in T_r

Step 6: To assume P can be leaf node set in T_r

Step 7: To generate a secret key role using the following equation (3):

$$SK_r = (K_r, \left(\forall_p \in P: C_P = g^{Y_p(0)}, C'_p = H\left((p)^{Y_p(0)v}\right)\right) \quad (3)$$

where $K_r = g^{\frac{1}{s+H_1(ID_r)}}$, C_P: cipher text for leaf node P, g: key generation, H: hierarchical tree manner

Step 8: To generate public role parameters PUB_{rp} using the following equation (4):

$$PUB_{rp} = \left(h^{(s+H(ID_r))\prod_{k=1}^{n}\left(s+H\left(ID_{r_k}\right)\right)}, \left(h^{(s+H(ID_r))\prod_{k=1}^{n}\left(s+H\left(ID_{r_k}\right)\right)}\right)^m\right) \quad (4)$$

Where, $\left(ID_{r_1}, ID_{r_2}, \ldots, ID_{r_m}\right)$ can be the identities role r's every predecessor roles.

Thus, the role has been generated in the hierarchical tree manner when the data owners upload their data to cloud service providers, and therefore, authorized data consumers are only allowed for data accessing or downloading.

b. Duplicated data checking

Before the data uploading, the duplicated data is checked through the blockchain process, in which the smart contract is done using SHA-256 to ensure integrity and confidentiality of the data stored on CS (Cloud Server) and to avoid illegal data uploading and modifications. Compared to SHA-512, SHA-256 can provide shorter outputs that save bandwidth, and thus, we have chosen SHA-256. The SHA-256 hashing function will be utilized in the bitcoin process of

blockchain to generate smart contracts between the data owner and CSP for duplicated data checking process. The SHA-256 hash function has been illustrated in Figure 2.

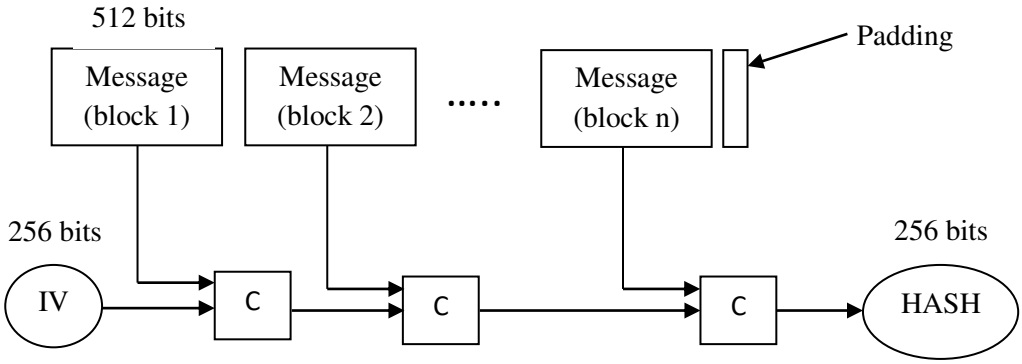

Figure 2 SHA-256 Hash function

In this process, the hash code key is generated by the cloud server using SHA-256 when the data owner tries to upload new data. The input file can be added with padding and a fixed 64-bit length field. The input enlarged data (message) is chopped into 512 bits-sized files (messages). These messages are in the multiple blocks with an extra 64 bits padding only for the final message. From the input data, the first message and 256 bits are provided to C that can be a compression process, in which 256-bit output is, produced that can be an input for a second message that goes through C to produce 256 bits. When continuing this process until the end of every message, the input file's HASH is obtained in 256 bits by the cloud server. Thus, original input data has been divided into several blocks. Afterward, for each block, the hashtag values are allocated. For a particular block's tag value, hash code is generated by applying the same SHA-256 algorithm. Here, hashtag values can be shown in blockchain with transaction information about the file's storage payment. The data owner can check duplication check through a cloud

server locally as this unchangeable information. In the duplicated data checking process, cloud servers check the hashtag is available in the hash table. If the hashtag is already available in the hash table, the cloud server replies "duplicate" to the data owner, in which role verification is done according to the secret role key SKr and thus, datais not required to be uploaded by the same data owner. If the same input files obtainable in different owner data storage space, then the cloud server sends the stored data's location link.

For example, while the data owner tries to upload the same data again, the hash value is calculated with the SK_r using SHA-256. Here, the input file is divided into various blocks. In our proposed system, the block sizes 10 MB, 20 MB, 30 MB, and 40 MB considered. For every divided block, the tag key has been generated. After that, for every tag key, the hash value can be calculated by SHA-256. The hashtag is checked for an appropriate file by the cloud server during the data uploading process. If the input data's hashtag is already in the hash table, the queries asked by the CSP "what is the hash tree's path?". If the data owner gives the correct path, the data owner's SK_r is verified by the CSP. If the SK_r is the same, the data is not stored again by the cloud server. The hash tree path is in the form of $P(HT) = \{RRL, RLR, RRR, etc\}$, where, HT: Hash Tree, RRL: Root, Right, Left, RLR: Root, Left, Right, RRR: Root, Right, Right. Therefore, the leaf node is not needed to add to the hash tree. The same data owner tries to upload the same data is estimated by the following equation (5):

$$S_{do} \xrightarrow{upload\ data} S_d \left(CSP \xrightarrow{replies} duplicate \right) \tag{5}$$

Where S_{do} denotes the same data owner, S_d represents the same data.

If the SK_r is different, the stored particular data location's reference link is sent by the cloud server to the different data owner and it is expressed by the following equations:

$$D_{do} \xrightarrow{upload\ data} S_d \left(CSP \xrightarrow{ask} SK_r \& P(HT) \right) \tag{6}$$

$$\left(SK_r \& P(HT) \right)_{matched} \rightarrow S_d \left(CSP \xrightarrow{send} R_l(d) \right) \tag{7}$$

$$\left(SK_r \& P(HT) \right)_{not\ matched} \rightarrow S_d \left(CSP \xrightarrow{replies} Invalid\ user \right) \tag{8}$$

Where, D_{do} denotes the different data owner and $R_l(d)$ represents the reference link of particular stored data

If the hashtag is not available in the hash table, the cloud server replies "no duplicate" to the data owner, and encrypted input data is uploaded to the cloud server. Here, the encryption is done through the MLE method. The data owner first puts the decryption key in the encryption area before encrypting input data. Efficient data confidentiality is provided by the MLE function. The six primitive functions are utilized in this paper for the data encryption such as $MLE = (ParGen, KeyGen, Enc, Dec, Equ, Valid)$ working with space of plaintext, ciphertext, keyspace such as $M = \{M_\lambda\}_{\lambda \in \mathrm{N}}$, $C = \{C_\lambda\}_{\lambda \in \mathrm{N}}$, and $K = \{K_\lambda\}_{\lambda \in \mathrm{N}}$. For the data encryption, the following processes are done before the data uploading to the cloud server.

- $ParGen$ represents the parameter generation algorithm takes 1^λ as input and returns public parameters pp as output:

$$PP \leftarrow keyGen(1^\lambda) \quad (9)$$

- $KeyGen$ represents the random key generation algorithm that takes public parameters pp and message M as inputs and returns a key to the message

$$SK_M \leftarrow KeyGen_{pp} \quad (10)$$

- Enc is the encryption algorithm that takes public parameters PP, a message M, and generated secret key SK_M as inputs. It returns a cipher text:

$$C \leftarrow Enc_{PP}(SK_M, M) \quad (11)$$

- Dec represents the decryption algorithm that takes PP, C, and SK_M as inputs and outputs a message M:

$$M \leftarrow Dec(SK_M, C) \quad (12)$$

- Equ is the equality algorithm which takes as PP, two cipher texts C_1 and C_2 as inputs, and it given outputs as 1 when the both C_1 and C_2 have been generated from the same message and is given by:

$$Equ_{PP}(C_1, C_2) = 1 \quad (13)$$

- $Valid$ represents the validity-test algorithm that takes PP and C as inputs and gives output as 1 if the C can be valid cipher text

$$Valid_{PP}(C) = 1 \quad (14)$$

Algorithm

Input: Original data

Output: deduplication, file uploading to the cloud server

Step 1: To initialize smart contract with key K, tag T, message blocks (MB_1, MB_2, MB_3,…, MB_n), and hash tree HT

Step 2: To generate tag key K by applying the SHA-256 for appropriate data 'D'

Step 3: To divide 'D' into several message blocks (MB_1, MB_2, MB_3,…, MB_n)

Step 4: To generate hashtag 'T' for the data 'D'

Step 5: To format the hash table using the same SHA-256

Step6: To check whether the hashtag is obtainable in the hash table through the cloud server

Step 7: To verify the role of data owner based on the SK_r, if cloud server replies "duplicate". If the SK_r is the same, data is not needed to upload again. If the SK_r is different, then the reference link of appropriate stored data is sent to the data owner. Thus, the data duplication is eliminated.

Step 8: To insert has tag into the hash table and to encrypt data using MLE if cloud server replies "no duplicate" and then encrypted data is uploaded with generated roles in the cloud server.

Step 9: Stop

Thus, the encrypted data can be uploaded to the cloud storage server with generated key roles such as SK_r, ID_r that have been generated by the ARKG.

C. Secure data downloading

For the more secure data downloading by the data consumers, the tag value and generated roles are utilized in the proposed methodology. Initially, the data consumer sends particular data's tag value to the cloud server. After that, the hashtag value can be generated by

applying the SHA-256 algorithm. Consequently, the cloud server verifies whether the tag value available in the hash table. If the tag value is not available in the hash table, then the cloud server considers that data consumer as an invalid user. If the tag value is available in the hash table, then the following processes are done to download the data securely:

Add data consumer: This process can be done through the generated key roles that have been given in the above section (a). Cloud server takes SKr (that contains access policy) and identity of data consumer (user) ID_C as inputs. The cloud server can check whether the consumer attributes satisfy the role's access policy. If the attributes of the data consumer satisfy the role's access policy, then the data consumer is added and a secret decryption key is given to that added data consumer to download the data securely.

Decryption: This process can be done by the added consumer to decrypt the encrypted data outsourced in the CSP using MLE (equation (12)). In this task, the data consumer's secret decryption key SK_M is taken as input and the plain text M is returned as output if the user or data consumer received permission from the cloud server to access the data and otherwise, it gets fails.

Delete data consumer: In this process, SKr and ID_C are taken as input, the user is removed from the data consumer role's list RUL, and public role parameter PUBrp is updated.

Algorithm:

Input: Tag value, SKr, ID_C

Output: Secure data downloading

Step 1: To initialize original data, hashtag value, SKr, ID_C

Step 2: To generate hashtag value for all tags by applying the SHA-256

Step 3: To check whether the hashtag value is obtainable in the hash table through the cloud server

Step 4: If the cloud server informs hash value is obtainable in the hash table, the data consumer is permitted to access the data for data downloading, and otherwise, the cloud server informs that consumer is invalid.

Step 5: To add data consumer or user if satisfies the role's access policy according to the SKr, ID_C and to provide the secret decryption key to the added user.

Step 6: To decrypt the ciphertext C using MLE (equation (12))

Step 7: Data is downloaded by the added user securely

Step 8: Stop the algorithm

Thus, the data can be downloaded by authorized users only to avoid data duplication.

4. Results and Discussions

The performance of the proposed secure data deduplication and existing data deduplication methods such as SDVADC [2], ZEUS [6], and RCDSD [7] are simulated and compared using TensorFlow Tool in terms of data deduplication rate, deduplication throughput, time complexity, and communication overhead. The technical specifications are summarized in Table 1 for the experiments.

Table 1 Technical Specifications

Parameters	Specifications
Processor	Intel core I3
Hard Disk	500 GB
Monitor	15 VGA Color.
Mouse	Logitech
RAM	2GB
Operating System	Windows 7, 64 bit
Program Software	TensorFlow Tool
Program Language	Python 3.5

a. Data Deduplication rate

The comparison of the data deduplication rate has been illustrated in Figure 3. From the blow comparison chart, the proposed method improved blockchain-based deduplication has taken high data deduplication rate compared to existing methods SDVADC [2], ZEUS [6], and RCDSD [7]. The proposed method has taken a 28% data duplication rate for the 5MB file while the existing methods SDVADC [2], ZEUS [6], and RCDSD [7] given 23%, 22%, and 26%. The proposed method has given a 26.7% data duplication rate for the 25MB file while the existing methods SDVADC [2], ZEUS [6], and RCDSD [7] given 20.9%, 20%, and 24.5%. Likewise, for other file sizes 10MB, 15 MB, and 20 MB, the proposed secure deduplication scheme has provided the best results compared to SDVADC [2], ZEUS [6], and RCDSD [7].

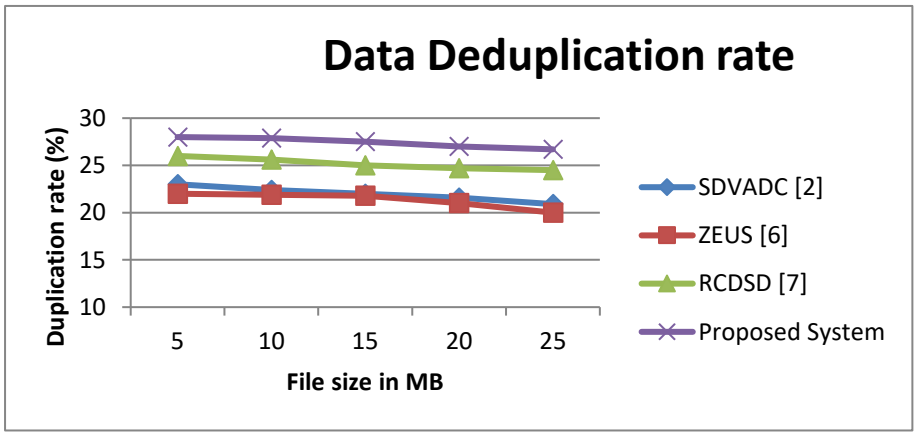

Figure 3 Data deduplication rate

b. Deduplication Throughput

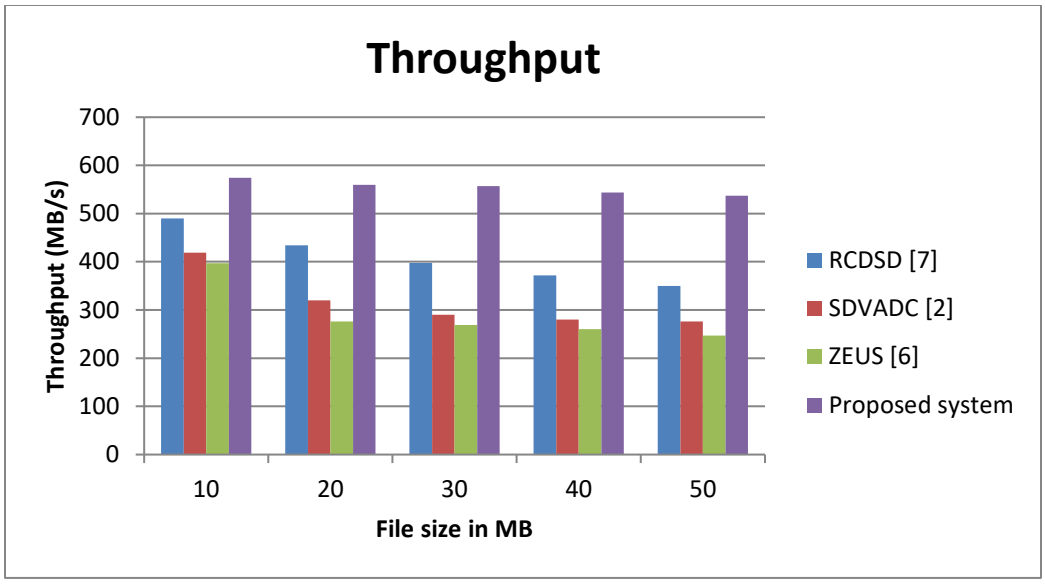

Figure 4 De-duplication throughput

Deduplication throughput is the rate at which blocks, hashtags, and keys are generated successfully. Deduplication throughput has been demonstrated in Figure 4. In Figure 4, X-axis contains the files in MB size and Y-axis contains the throughput rate in MB/S (megabit per second). From Figure 4, it is clearly stated that the proposed system has taken high throughput which means it has generated the blocks, keys, and hashtags at faster rates for the deduplication checking process compared to existing methods such as SDVADC [2], ZEUS [6], and RCDSD [7].

c. Time Complexity

The comparison chart of time complexity has been demonstrated in Figure 5. In this kind of estimation, the execution time of key role generation, block generation, the hashtag generation is evaluated on a given dataset. The execution time is computed in milliseconds. The below comparison chart 5 has demonstrated that the proposed secure deduplication method has taken only a small amount of time to deduplication detection and elimination. As a result, this presented deduplication method can take low time complexity to detect and eliminate duplicated data compared to existing deduplication methods SDVADC [2], ZEUS [6], and RCDSD [7].

Figure 5 Comparison of Time complexity

d. Communication Overhead

The communication overhead of the proposed and existing methods has been illustrated in Figure 6. In Figure 6, X-axis contains the number of data files used for the performance of the proposed and existing deduplication model in terms of communication overhead's percentage level. As observed from Figure 6, the proposed deduplication system has taken less communication overhead for secure data deduplication process in the cloud storage than existing methods SDVADC [2], ZEUS [6], and RCDSD [7].

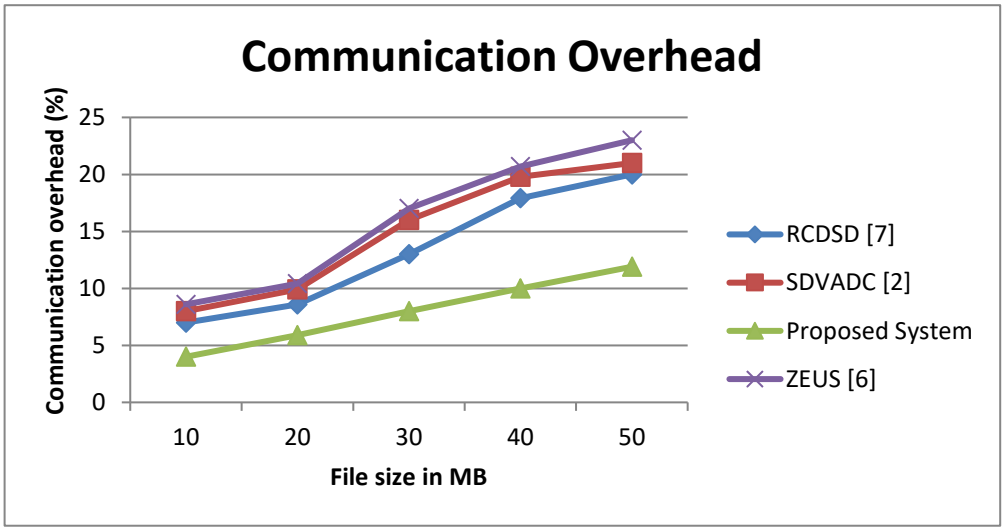

Figure 6 Communication overhead Comparison

5. Conclusion

In this paper, an improved block-chain based secure data deduplication method has been presented to save cloud storage space efficiently. In this work, ARKG was applied to generate key roles before the data uploading for data accessing by authorized users. The duplicated data has been checked in the block-chain process, in which the smart contract was done with SHA-256 to verify the integrity and confidentiality of the users. For the duplicated data checking, the hash key, the hashtag was generated by applying the SHA-256. The cloud server verifies the duplicated data through hashtag availability in the hash table. The data is not uploaded to the cloud storage server if received "duplicate". The data has been encrypted efficiently with the help of the MLE method when gets information "no duplicate" from the cloud server before the uploading. Thus, ciphertext has been uploaded by the data owner. For secure downloading, the generated roles are verified based on the data consumer's attributes. If roles are matched, then authorized consumers can download the data securely with a decryption key. Finally, the performance of the proposed deduplication method was estimated and compared with existing deduplication methods SDVADC [2], ZEUS [6], and RCDSD [7]. This proposed method has given a high deduplication rate, and high deduplication throughput.

Declarations

1. **Funding**
   Not Applicable
2. **Conflicts of interest/Competing interests**
   There is no conflict of interest from all the authors in the manuscript.
3. ***Availability of data and material**
   Not Applicable
4. ***Code availability** (software application or custom code)
   Not Applicable
5. ***Authors' contributions**
   Ruba S – Overall concepts, literature survey, Working and ideology, Results development
   AM Kalpana – Supervising, Proof editing

References

[1]. A-guide-to-data-de-duplication 2015, http://www.computerweekly.com/feature/A-guide-to-data-de-duplication. Copyright 2015

[2]. Geet, C M, Shreyas Raju, R.G, Raghavendra, S, Rajkumar Buyya, Venugopal, K.R, SIyengard, S & Patnaike, LM, "SDVADC: Secure Deduplication and Virtual Auditing of Data in Cloud", Procedia Computer Science, vo.171, no.2020, pp. 2225-2234, 2020.

[3]. Shynu P. G, Nadesh R. K, Varun G. Menon, Venu P, Mahdi Abbasi and Mohammad R. Khosravi, "A secure data deduplication system for integrated cloud-edge networks", Journal of Cloud Computing Advances, Systems and Applications, vol.9, no.61, pp.1-12, 2020.

[4]. Emna Baccour, Aiman Erbad, Amr Mohamed, Mohsen Guizani & Mounir Hamdi, "CE-D2D: Collaborative and Popularity-aware Proactive Chunks Caching in Edge Networks", Ninth International Conference on Computational Intelligence and Security, pp.607-609, 2020.

[5]. Chia-Mu Yu, Sarada Prasad Gochhayat, Mauro Conti & Chun-Shien Lu, "Privacy Aware Data Deduplication for Side Channel in Cloud Storage", IEEE Transactions on Cloud Computing, vol.8, no.2, pp. 597 – 609, 2020.

[6]. Aobing Sun, Guohong Gao1, Tongkai Ji & Xuping Tu, "One quantifiable security evaluation model for cloud computing platform", 2018 Sixth International Conference on Advanced Cloud and Big Data, 978-1-7281-3129-0/20/$31.00 ©2020 IEEE, pp.197-201, 2020.

[7]. Shivansh Mishra, Surjit Singh & Syed Taqi Ali, "RCDSD: RSA Based Cross Domain Secure Deduplication on Cloud Storage", pp.1-7, 2018.

[8]. Wenhai Sun, Ning Zhang, Wenjing Lou & Thomas Hou, Y, "Tapping the Potential: Secure Chunk-based Deduplication of Encrypted Data for Cloud Backup", IEEE Conference on Communications and Network Security (CNS), pp.1-9, 2018.

[9]. Shubhanshi Singhal, AkankshaKaushik & Pooja Sharma, "A Novel Approach of Data Deduplication for Distributed Storage", International Journal of Engineering & Technology, vol.7, no.2.4, pp.46-52, 2018.

[10]. Yinghui Zhang , Haonan Su, Menglei Yang , Dong Zheng , Fang Ren and Qinglan Zhao, "Secure Deduplication Based on Rabin Fingerprinting over Wireless Sensing Data in Cloud Computing", Security and Communication Networks, vol.2018, pp.1-12, 2018.

[11]. Sai Nikhila, B, Kiranmaiee, K, Divya Vadlamudi ,Dr.K.Thirupathi Rao & Deevi Radha Rani, "A Framework To Implement Secure De-Duplication Using SHA-512 In Cloud Environment", vol.115, no.8, pp. 37-43, 2017.

[12]. Rishikesh Misal & Boominathan Perumal 2019, 'Data Deduplication for Efficient Cloud Storage and Retrieval', The International Arab Journal of Information Technology, vol.16, no.5, pp.922-927.

[13]. Chia-Mu Yu, Sarada Prasad Gochhayat, Mauro Conti & Chun-Shien Lu, "Privacy Aware Data Deduplication for Side Channel in Cloud Storage", IEEE Transactions on Cloud Computing, vol.8, no.2, pp. 597 – 609, 2020.

[14]. Guangping Xu, Bo Tang & Hongli Lu, "LIPA: A Learning-based Indexing and Prefetching Approach for Data Deduplication", 35th Symposium on Mass Storage Systems and Technologies (MSST), pp.299-310, 2019.

[15]. Yinjin Fu, Nong Xiao, Hong Jiang, Guyu Hu & Weiwei Chen, "Application-Aware Big Data Deduplication in Cloud Environment", IEEE Transactions on Cloud Computing, vol.7, no.4, pp.921 – 934, 2019.

[16]. Zheng Yan, Wenxiu Ding, Xixun Yu, Haiqi Zhu & Robert H. Deng, "Deduplication on Encrypted Big Data in Cloud", IEEE Transactions on Big Data, vol.2, no.2, pp. 138 – 150, 2016.

[17]. Kaur, Hasveen & Mann, P. S, "An Improved Hybrid Re-Encryption Scheme for Mobile Cloud Computing Environment", International Journal of Computer applications, vol.162, no.4, pp.12-16, 2017.

[18]. Namrata P. Kawtikwar & Joshi, M. R, "Data Deduplication in Cloud Environment using File-Level and Block-Level Techniques", IJIR, vol.3, no.5, pp.1294-1298, 2017.

[19]. Li, J, Qin, C, Lee, P & Zhang, X, "Information Leakage in Encrypted Deduplication via Frequency Analysis", in Proc. of IEEE/IFIP DSN, pp. 2110–2118, 2017.

[20]. Periasamy, J. K & Latha, B, "An enhanced secure content deduplication identification and prevention (ESCDIP) algorithm in cloud environment", Neural Computing and Applications, vol.32, pp. 485–494, 2018.

[21]. Priteshkumar Prajapati & ParthShah, "A Review on Secure Data Deduplication: Cloud Storage Security Issue", Journal of King Saud University - Computer and Information Sciences, pp.1-9, 2020.

[22]. MyungKeun Yoon 2019, 'A constant-time chunking algorithm for packet-level deduplication', ICT Express, vol.5, no.12, pp. 131-135.

[23]. Paulo, J & Pereira, J, "A survey and classification of storage deduplication systems", ACM CSUR, vol. 47, no. 1, pp. 1-30, 2014.

[24]. Okonski & Aleksander, "Implementing Security Rules, Safeguards, and IPS tools for Private Cloud Infrastructures: GROOT: Infrastructure Security as a Service (ISaaS)", Thesis, pp.1-57, 2018.

[25]. Di Pietro, R & Sorniotti, A, "Proof of ownership for deduplication systems: a secure, scalable, and efficient solution", Computing Communication, vol.82, pp.71–82, 2016.