Modelling a Request and Response-Based Cryptographic Model For Executing Data Deduplication in the Cloud

Doddi Suresh Kumar[1], Nulaka Srinivasu[2,*]

[1,2*]Computer Science and Engineering, Koneru Lakshmaiah Education Foundation, Guntur, Andhra Pradesh, India

Emails: sureshdkumarmca@kluniversity.in; srinivasu28@kluniversity.in

Abstract

Cloud storage is one of the most crucial components of cloud computing because it makes it simpler for users to share and manage their data on the cloud with authorized users. Secure deduplication has attracted much attention in cloud storage because it may remove redundancy from encrypted data to save storage space and communication overhead. Many current safe deduplication systems usually focus on accomplishing the following characteristics regarding security and privacy: Access control, tag consistency, data privacy and defence against various attacks. But as far as we know, none can simultaneously fulfil all four conditions. In this research, we offer a safe deduplication method that is effective and provides user-defined access control to address this flaw. Because it only allows the cloud service provider to grant data access on behalf of data owners, our proposed solution (Request-response-based Elliptic Curve Cryptography) may effectively delete duplicates without compromising the security and privacy of cloud users. A thorough security investigation reveals that our approved safe deduplication solution successfully thwarts brute-force attacks while dependably maintaining tag consistency and data confidentiality. Comprehensive simulations show that our solution surpasses the evaluation in computing, communication, storage overheads, and deduplication efficiency.

Keywords: data deduplication; cloud; encryption; decryption; redundancy

1. Introduction

Data owners are increasingly outsourcing their data and delivering it to authorized users through the cloud due to the many benefits of cloud computing [1]. For instance, according to the Cisco Global Cloud Index, there will almost certainly be 1.3 zettabytes of data stored in the cloud. As a result, managing the constantly growing amount of data poses a severe problem for cloud storage providers [2]. According to the research, around 75% of digital data are similar, while backup and archive storage systems have redundancy levels that are substantially higher than 90%. Given that it may drastically with data deduplication, which keeps just one copy of redundant data, you may save storage expenses approach has been widely explored in cloud storage [3]. Data deduplication may lower storage costs in backup applications by more than 90% and ordinary file systems by more than 50%, resulting in significant cost reductions for users and cloud service providers. To preserve the security and privacy of their data, users are likely to encrypt their data using personal keys before outsourcing [4]. Due to identical data being encrypted into several ciphertexts, data deduplication would be inhibited. Convergent encryption and its applications are discussed, or variations were created to make it possible to deduplicate encrypted data. However, convergent encryption is susceptible to attacks using brute force for predictable communications [5]. Some server-aided encryption techniques have been suggested to address this problem.

Unfortunately, the duplicate fake attack makes it impossible for genuine users to get accurate data. To put it more precisely, the antagonist generates the related tag and cipher-text from various data, m and n, respectively. Users who later submit the label only when the cloud service provider has registered the incompatible ciphertext

Doi : https://doi.org/10.54216/JCIM.140206
Received: January 10, 2024 Revised: March 13, 2024 Accepted: July 01, 2024

and tag may correspond to m receive the incorrect data for m. After acquiring the ciphertext, users check the tag consistency; however, even though specific methods attempt to defend against this attack, it is difficult to determine if the wrong data was produced by repeated fake assaults during uploaded data was corrupted or destroyed while being stored [6]. Since the related tag and the ciphertext are created separately, it is challenging for the cloud service provider to ensure the uniformity of the titles, which is the main driving force behind this attack [7]. Therefore, a technique that directly computes the tag by hashing the ciphertext is described to execute the tag consistency check by comparing the ciphertext hash to the received tag [8]. Message authentication and tag consistency have been considered. Using this concept as a foundation, users authenticate messages after data download, and the provider of cloud services checks tag consistency [9].

Access control is often used in practical cloud solutions as one of the critical elements of cloud computing [10] – [12]. Consequently, the development of permitted secure deduplication is an unavoidable trend. But as far as we know, only a few techniques address allowed safe deduplication by utilizing a hybrid cloud architecture or an additional approved server. Still, they are susceptible to duplicate fake assaults, etc. From this, it is observed that no current technique can simultaneously ensure data secrecy, access restriction, resistance against brute-force assaults and tag consistency [13]. To solve this issue, we provide a user-defined access restriction cross-user deduplication approach in this study. This scheme primarily consists of the following four contributions:

- ✓ Since user identity is the cornerstone of many cloud-based access settings, we picked user identification as an example feature rather than complexity attributes. Our method lets users choose the symmetric key randomly rather than immediately hashing the data to thwart the brute-force attack. The symmetric ciphertext may be directly generated into a tag, facilitating the consistency check in the interim.

- ✓ The Request-Response method and the Elliptic Curve cryptosystem (ECC) are expertly combined in our scheme to support access control, resulting in a robust proxy re-encryption method that guarantees that the only party authorized to approve on behalf of the access is granted without compromising privacy, the cloud service provider and data owner can access the data. The final re-encryption requires the private essential process can only be accessed by the CSP; our architecture can monitor the CSP's activity in the interim. To keep its good reputation, when executing re-encryption operations, the CSP must follow each user's authorization set even if it is not entirely trusted.

- ✓ According to thorough security tests, our technology is the first to simultaneously offer data secrecy, security against attacks, tag consistency, and access management. In contrast to rival methods, ours does not require adding a server or continuous user connectivity during the data upload phase.

- ✓ To effectively check the duplication, we employ the filter to lessen the temporal complexity of the duplication search. Extensive testing has shown the advantages of our technique in terms of deduplication effectiveness, computational cost, communication overhead, and storage cost.

In this work, the user-level access right was specified as the degree of granularity. In this case, the data owner creates detailed access guidelines and transmits them. It is up to the cloud service provider to evaluate the eligibility of the other users. They might obtain the owner's secret key with the aid of the cloud service provider if they follow the rules laid out by the data owner. To make our presentation more transparent, we use the property of user identification as an example; for instance, the data owner u_1 permits other users with IDs of u_2, u_3 and u_4 to share its storage. Expanding our system to allow more granular access rights, such as file-level access, is simple. To be more specific, only if the file attributes and policies are by one another can the providers of cloud services do data deduplication.

The work is drafted as follows: section 2 gives a broader analysis of data deduplication approaches, and the methodology is demonstrated in section 3. The experimental outcomes are presented in section 4, with a work summary in section 5.

2. System design

This section includes a threat model, a model of approved deduplication connected to it and our design objectives [14]. Three entities comprise our authorized deduplication system: a cloud service provider, a key generation centre, and many users $U = [U_1, U_2, \ldots, U_w]$—comparable studies on cloud computing. The

Doi : https://doi.org/10.54216/JCIM.140206
Received: January 10, 2024 Revised: March 13, 2024 Accepted: July 01, 2024

system's architecture is depicted, and the specifics of each entity are detailed in the following paragraphs, as seen in Fig 1.

Key generation centre (KGC): The initial configuration of the entire system is the KGC's responsibility. It generates system settings, public/private key pairs, and the cloud service provider's secret key for each user. If no further users sign up for the system after that, the KGC might remain down [15]. Remember that for the newbie, the public/private key pair must only be generated by the KGC.

Cloud service provider (CSP): Users can get storage services through the CSP. It would want to eliminate unnecessary data from the permitted deduplication system and retain just one copy to make data storage more affordable without breaking access control [16].

Users: Authorized users or data owners can utilize our approved deduplication system. The person who makes authorized users aware of encrypted data and outsources it to the CSP is often called the data owner. The term "authorized user" usually refers to a person to whom the owners of the data have permitted to access that data [17].

3. Design protocol

The system configuration, note creation and Insertion, initialization of checks, calculation of responses, and integrity checks are the five essential steps of the cryptographic protocol [18].

System configuration: AA outputs secret function f upon input of the security parameter, which is subsequently used for production.

Insertion: After entering the hidden function f and the data holder's secret keys, the data holder generates randomized message set S and position set P using blocks that were uploaded to its file and inserts the note set into the appropriate areas of the encrypted blocks the start of the check [19]: The data holder constructs a coefficient set e after gathering the block check indices, then calculates the challenge set $v = b * e \, mod \, m$, where gcd $(m, b) = 1$.

Response evaluation: CSP generates the response $Response = v * D$ after receiving the challenge set as input.

Integrity verification: The data bearer computes $Response = Response * b^{-1}$ to identify the note set after receiving the answer $Response$, and they confirm that the selected messages are consistent with the hidden function f. The data holder verifies the saved file's integrity if the verification is successful [20].

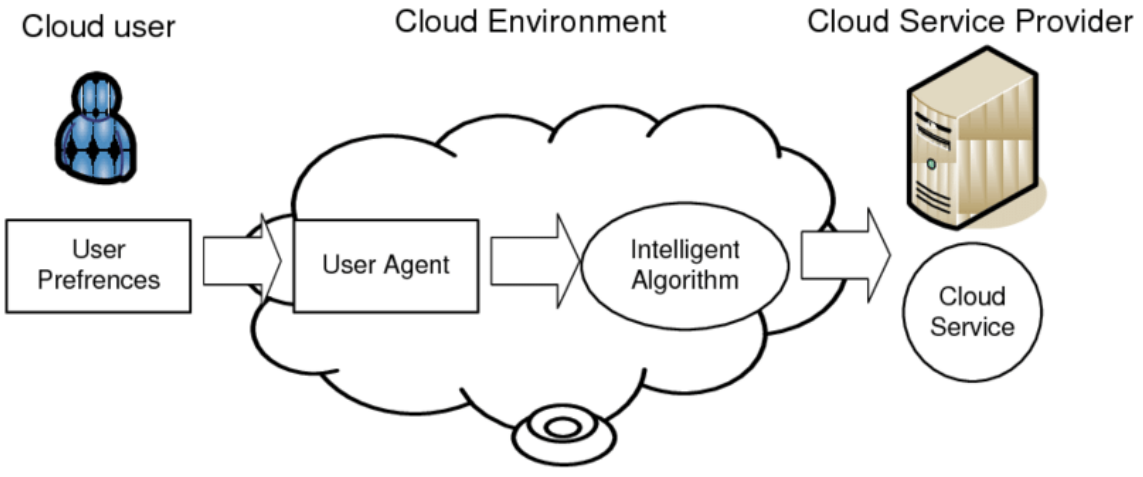

Figure 1: Cloud environment setup

Doi : https://doi.org/10.54216/JCIM.140206
Received: January 10, 2024 Revised: March 13, 2024 Accepted: July 01, 2024

4. System setup

Assume $e: G_1 * G_1 \rightarrow G_T$ The random number generators $g \in G_1$ and Z, e.g., $Z = e(g, g) \in G_T$ are the settings for the system. There are two prime orders q groups: G_1 and G_T, and G_T is a bilinear map. Each data bearer in the system creates $sk_w = a_w$ and $pk_w = g^{a_w}$ for PRE during system setup where $a_w = Z_{p^*}$. The re-encryption key for u_w is created by AA using the public key pk_w. Here, P will serve as a starting point for system entities $Eq(a, b)$ across an elliptic curve $GF(p)$, $s_w \in R\{0, \dots, r^\sigma - 1)$ data holder u_w elliptic curve secret key, $V_w = -s_w P^*$ a matching both a security parameter as well as a public key. A unique data owner identity is associated with the keys (pk_w, sk_w) and (V_w, s_w) of u_w, which may be a pseudonym essential for user identity verification [21]. To establish their own note sets $S_{w,i}$, each data holder will subsequently employ the hidden f AA created as a consensus and broadcast to all data owners if f might be any arbitrary function chosen by the level of security demanded by the data proprietors. The production of pk_{AA} and sk_{AA} for PRE is another function of AA and pk_{AA} to broadcast data owners. The CSP uses public and private keys (e, d) to initialize the ECC algorithm under module N. The crucial team encrypts the tags supplied by the CSP and the data owners for duplicate checks [22].

4.1. Verify duplication

Assume that F_1 and F_2 are two data files that two data holders, u_1 and u_2, willingness to surrender to the CSP. u_1 should upload the file first; it will then carry out the following data preparation and duplication check [23]:

Step 1: Perform the following calculations using input F_1 and the symmetric key SK_1.

1) Divide F_1 into several splits, each with m blocks. Using error-correcting code (ECC), expanding m blocks to $m + d - 1$ blocks and having an effective $d/2$ correction for errors can safeguard the file against minor corruptions.

2) For each block $B_{1,i}, u_1$ creates a block tag $y_{1,i} = H(H(B_{1,i}) * P^*)$.

3) Deliver the collection of tags $\{y_{1,i}\}$ to the CSP.

Step 2: The CSP and AA work together and u_1 to carry out a duplication check in line with the processes below, supposing the CSP retains track of a tag set $\{x_j\}$ created from previous data owners [24].

1) The CSP creates $a_j = H(x_j) mod\ N$ for each $H(x_j)$ and transmits $\{a_j, H(x_j), sign\ (H(x_j))\Delta$ to AA. Here, D represents the overall number of tags the CSP manages and $sign\ (H(x_j))$ stands for the original data owner's signature on $H(x_j)$.

2) Once the data has been received from the CSP, AA checks whether $\{a_j, H(x_j), sign\ (H(x_j))$ includes D items to ensure the CSP utilizes all its kept tags when doing calculations. If so, it next ensures the CSP uses the tags left behind by the previous data owners provided by verifying each signature on $H(x_j)$. Third, AA will carry out further batch verification to evaluate the consistency of the CSP computations. Non-overlap subsets (N_v) for this batch verification will be generated randomly of $\{a_j, H(x_j)\}$ and check whether $\prod H(x_j) = (\prod a_j)^e$ holds in each subgroup [25].

The AA constructs an input cuckoo filter CF to $\{a_j\}$, $CF = CF.Insertion\ (\{a_j\})$ as a response to u_1, assuming all the verification has succeeded. The CSP and AA will update the cuckoo filter whenever a new data owner requests to transfer new files to the CSP during system setup; this step is only performed once. For the data above owners, there is no need to recalculate the settings [26].

3) For every $y_{1,i}, u_1$ first chooses random numbers $r_{1,1,\dots,}r_{1,N_c}$ and computes $r_{1,i}^{inverse} = r_{1,i}^{-1}$ and $r'_{1,i} = r_{1,i}^e\ mod\ N$ for all $i \in [1, \dots, N_c]$ and then computes $A[1, i] = H(y_{1,i}).r'_{1,i}$, where $i = 1, \dots, N_c$ and delivers them to CSP. Then, the CSP responds to u_1 by computing C'; $C[1, i] = A[1, i]^d$. Then, after creating $N'_{1,v}$ non-overlapping subsets of $\prod C[1, i] = (\prod A[1,])^{i.e}$. random, u_1 checks whether $CF.Check\ (C[1, i].r_{1,}^{inverse}\ mod\ N)$ holds in each subset to validate the duplicate blocks and demonstrate the accuracy of the CSP computations.

Doi : https://doi.org/10.54216/JCIM.140206
Received: January 10, 2024 Revised: March 13, 2024 Accepted: July 01, 2024

4.2. Data upload

Let u_2 upload the file after u_1 to receive the same blocks as u_1, then u_1 and u_2 will append notes and upload data as follows:

Step 1: U_1 must upload its matching blocks $\{B_{1,i}\}$. The duplicate check is unsuccessful because u_1 is the first user to submit a brand-new file that has never been stored by the CSP, previously making u_1 the data owner. Assuming block B_1; i is the i^{th} block to be uploaded, u_1 first encrypt block $B_{i,1}$ with DK_1 to obtain $CT_{1,i}$; it is encrypted using pk_{AA} to create CK_1 and then saved as an $X * Y$ matrix. An assortment of notes that fulfils the hidden purpose f is denoted as $S_{1,i} = \{\eta_{1,i,0,} \dots, \eta_{1,i,k} | f(\eta_{1,i,0}, \dots, \eta_{1,i,k}) = 0\}$. After rearranging the column indexes, the PIR method chooses to enter the notes in the first r columns $[1, \dots, X]$ and using $B_{1,i}$ as a seed. As a result, $c = [k/r]$ notes must be placed in each column. The first c indices are chosen as the last locations to place the notes following another row index shuffle by u_1 where $[1, \dots, Y]$ in each chosen column. The position indexes are denoted as $P_{1,i} = \{p_{1,i,1,\dots,}p_{1,i,k}\}$. Next, u_1 inserts all of the notes $\{\eta_{1,i,k}\}$ into $CKI_{1,i}$. To acquire $CKI_{1,i}$, the position indexes $P_{1,i}$, sends $CKI_{1,i}$and CK_1 to CSP. u_1 simultaneously uploads tags for the new blocks $\{y_{1,i}\}$ to further check for duplication. The CSP adds the new block tag $y_{1,i}$ to the collection of tags it keeps initially after receiving it $x_j = x_j \cup y_{1,i}$, compute $a_i = H(y_{1,i})^d \, mod \, N$, and transmits $\{a_i, H(y_{1,i}), sign\ (H(y_{1,i}))\}$ to AA. Then, after checking the signatures on $H(y_{1,i})$, AA constructs $N_{1,v}^{\square}$" non-overlap subsets of $\{\{a_{1,i}\}, \{H(y_{1,i})\}\}$ and tests each subset to see whether $\prod H(y_{1,i}) = (\prod a_{1,})^{i.e}$. is true. If the verification is successful, AA updates the cuckoo filter CF, assuming the CSP calculation is accurate, which will be used in the subsequent duplicate check cycle using $CF = CF.Insert\ (\{a_{1,i}\})$. If the duplicate check is successful and the restored blocks originate from the same data holder, the data holder will advise the CSP to do nothing but maintain its blocks [27]. If blocks come from a certain data holder, the CSP will be notified and instructed to do deduplication.

Algorithm 1:

1. For $(i = 1, i \leq 3; i + +)$ do //upload dataset
2. For $(j = 1, j \leq 10; j + +)$ do //user upload dataset in 10 times
3. For $(k = 1, k \leq 3; k + +)$ do
4. At U_i, t_j selects input files randomly
5. end for
6. end for
7. For $(j = 10, j \leq 12; j + +)$ do
8. //for every input dataset, all the uploaded files at t_1 and t_2, respectively
9. For $(i = 1, i \leq 3; i + +)$ do
10. At t_j selects input files randomly S_k
11. end for
12. end for
13. end for

Step 2: When the CSP receives notification of the copying by a separate user, u_2. It initially confirms by assigning the AA responsibility for determining the ownership of the blocks, which then queries the data owner u_2 about its legitimacy as the legitimate owner of the data blocks $B_{2,i'} = B_{1,i}$. We provide a mechanism for confirming ownership based on the identifying method used by cryptography [29]. The guidelines state AA picks $c \in R\ \{0, \dots, 2^{\sigma} - 1\}$ at random and then challenges u_2 by c. In addition to V_2, u_2 computes $H(B'_{2,i}) + (s_2 * c)$ as a reaction to AA. AA will add the $H(hP^* + cV_2)$ and will be compared to the tag $y_{1,i}$. If the verification is successful, that is, if $y_{1,i} = H(hP^* + cV_2)$, then AA verifies that u_2 has the duplicated blocks $B'_{2,i} = B_{1,i}$ and creates the r_k, the re-encryption key. $rk_{AA \to u_j} = RG(pk_{AA}; sk_{AA}, PK_2)$ transfers it to CSP as its inputs. CSP then calculates $rk_{AA \to u_j} = RG(pk_{AA}; sk_{AA}; PK_2)$ to transfer CK_1 to CK_2 , $rk_{AA \to u_j} = E(pk_{AA}; sk_{AA}; DK_1)$ for u_2. The data blocks $B_{1,i}(B_{2,i'})$ stored at the CSP are currently accessible to both u1 and u2, who may utilize the appropriate $CTI_{1,i}(CTI^{i}_{2,i})$ to perform the integrity check below. Remember that each $B_{1,i}$ only has one correlation with $CTI_{1,i}$, i.e. $CTI_{1,i} = CT1_{2,i'}$.

Doi : https://doi.org/10.54216/JCIM.140206
Received: January 10, 2024 Revised: March 13, 2024 Accepted: July 01, 2024

4.3. Integrity verification

Assume that data owners u_1 and u_2 desire to upload block sets, $\{B_{1,i}\}$and $\{B'_{2,i}\}$ respectively. A large number m and $b \in Z_m^*$, are initialized by user u_1 where $\gcd(b,m) = 1$ as a secret regardless of deduplication before doubting the consistency of one block, $B_{1,i}$, which i saved at CSP since it is duplicate-free. Then, based on the position indices $P_{1,i}$, it produces a set $(e_{1,i,0,} \dots e_{1,i,z})$for each column $(x_{1,i,0}, \dots, x_{1,i,z})$ with random selected $d_{1,i,l}N^r$, where if $x_{1,i,l} \neq P_{1,i}$, then $e_{1,i,l} = d_{1,i,l}N^r$; otherwise, if $x_{1,i,l} \neq P_{1,i}$, then $e_{1,i,l} = N^l + d_{1,i,n}N^r$.In the interim, everyone has a choice of $\{\{v_{1,i,}tag_{1,i}\};$.

To transmit $Res_{1,i} = Response_{1,i} * b^{-1}$ to the CSP, it computes $\{v_{1,i,l} | v_{1,i,l} = be_{1,i,l} \, mod \, m, l \in [1, \dots, z]\}$. When the CSP receives the request $Response_{1,i}$, it computes the answer $Response_{1,i} = Response_{1,i} * b^{-1}$, and sends it back to u_1. After obtaining the requested columns by computing $Response_{1,i} * b^{-1} \, mod \, m$, u_1 selects the notes based on position indexes $P_{1,i}$ and determines if they satisfy the hidden function. Similar to this, when user u_2 challenges the CSP, it creates its unique (m', b') as secrets as well as its unique $(d_{2,i'}, \dots, d_{2,i'z})$ to generate other $(e_{2,i',\dots,}e_{2,i',z})$and its additional $Request_{2,i'} = (v_{2,i',}tag_{2,i'})$ where U_2 may furthermore get the note set based on position indices and assess if the notes comply with the concealed function $P_{2,i}$ in conjunction with the CSP.

Assume, therefore, that u_1 and u_2 each have their distinct block,$\{B_i^*\}$, while u_2 has $\{B_i^2\}$ and that they share a duplicate block, $\{B_i^2\}$. For $B_{1,i} \in \{B_i^*\}$ verifies $f(\eta_{1,i,0}, \dots, \eta_{1,i,k})$ to ensure $B_{1,i}$ integrity and u_2 verifies $f(\eta_{2,i'}, 0, \dots \eta_{8,i,k}) = 0$ for $B_{*,i} \subset \{B_i^*\}$. To confirm the consistency of $B_{*,i}$. Given that they both perform the same secret function, f, they may all confirm that $f(\eta_{*,i,0}, \dots, \eta_{*,i,k})$. and $P_{1,i} = P_{2,i'}$ even though u_2 is uninformed of the precise inserted notes of u_1. As a result, we go one step further, and the verification tags of blocks generated and reproduced by different data holders should be deduplicated in addition to the identical block uploaded to the CSP. We use ECC to aid in file recovery; an integrity check of all blocks is unnecessary. Our protocol ensures the integrity of F_1 and F_2 if u_1 and u_2 complete all of their associated block sets, above g times random verification [30].

4.4. Data download

Each time, u_1 wants to download F_1. The CSP receives a request along with the file name. When the request is received, the CSP determines if u_1 can download the file. The matching block is returned by CSP if it was passed, setting $\{CTI_{1,i}\}$ to u_1. To retrieve the ciphertexts $\{CT_{1,i}\} = \{CT_i^1\} \cup \{CT_i^*\}$, u_1, the notes are initially extracted based on each block's position indices $P_{1,i}$. Then, u_1 directly decrypts each $CT_{1,i}$ using DK_1 to produce $\cup \{B_{1,i}\} = \{B_i^*\} \cup \{B_i^1\}$. ECC allows for the recovery of F_1 from $\cup \{B_{1,i}\}$ with errors no more than $d/2$. As with u_2, it obtains a re-encrypted DK_1 key from the CSP after performing the identical processes to get the ciphertexts $\{CT_{2,i}\} = \{CT_i^*\} \cup \{CT_i^2\}$. Then, by directly employing DK_2, u_2 may acquire the critical DK_1 using its key pair of (pk_2, sk_2) and decode each CT_i^* to retrieve the duplicate original blocks $\{B_i^*\}$ and its unique original blocks $\cup \{B_{2,i}\} = \{B_i^*\} \cup \{B_i^2\}$.The original file $\cup \{B_{2,i}\} = \{B_i^*\} \cup \{B_i^2\}$ may be obtained, and F_2 can be recovered using ECC.

The CSP will likely enhance its revenue due to the vast computation/storage from deduplication. Verifying that deduplication has already occurred at the CSP is necessary to obtain a reduced storage price offer. It is the problem our research attempts to address. Another area of our research focuses on convincing all relevant parties to adopt and use deduplication schemes. To do this, Server-controlled, client-controlled, and hybrid deduplication are three examples. The client manages deduplication—we apply game theory to construct the appropriate reward or punishment systems. We might use the reward system suitable for server-controlled deduplication techniques to promote the proposed approach because our scheme design is based on the server-controlled deduplication scheme. Additionally, giving each CSP a trust score can help customers choose a dependable CSP.

Algorithm 2:

The user uploads dataset files, and CSP performs a successive process to validate duplication

1. Verify file duplication with input dataset;
2. if duplication prevails, then

Doi : https://doi.org/10.54216/JCIM.140206
Received: January 10, 2024 Revised: March 13, 2024 Accepted: July 01, 2024

3. return decision
4. else
5. for every user, $U_j \in I_i$ do
6. Verify file relationship;
7. if $O_j \subseteq O_i$, then
8. verify successive dataset duplication;
9. if duplication exists, then
10. if $O_i \subseteq O_j$ then
11. set $'j'$ value
12. end if
13. if $O_j \subseteq O_i$ then
14. set $'j'$ value as one and delete duplication
15. end if
16. else
17. set $j = 1$
18. end if
19. else
20. set $j = 1$
21. end if
22. end for
23. Execute outcome
24. return value
25. end if

5. Numerical results and discussion

In this section, we run tests in MATLAB 2020a. We use cloud libraries on an 8GB RAM, 2.3 GHz Intel Core i5 processor to assess our system's computational, communication, and storage overheads. We also compare and contrast our strategy with various existing approaches based on the user's access credentials regarding chunk level and the deduplication of files. We compare our method to demonstrate its efficacy without a second independent server. Since data deduplication only impacts, we did not reach the associated overheads from downloading data and configuring the system with the data upload stage. The simulation considers nine users $(U_1, U_2, \ldots, U_{10})$. Corresponding in Fig 2 and Fig 3, permission and access sets contain all potential set relationship scenarios. The simulation uses three datasets, $S_1, S_2, and\ S_3$, each with a distinct file size and 100 files each. $S_1, S_2, and\ S_3$ have files around Sizes ranging from 1 KB to 1 MB. A dataset is uploaded by every user S_k within ten-time intervals, where $k = 1, 2, 3$. The simulation of the data upload procedure is shown. To be more precise, ten individuals randomly select ten files from each dataset S_k $(k = 1, 2, 3)$ using $S_k/1 - (j - 1)$ before uploading them one at a time t_j. Each user completes uploading 100 S_k files at time t_{10}. Then, at times t_{11} and t_{12}, each user chooses ten files randomly, then from S_k to upload to confirm the benefit of intra-deduplication. We select type A for our scheme and type A for the other three methods regarding the ECC pairing parameters. We use the ECC encryption technique, which means that $k_1 = 256$ bits. In addition, we set the label's $|U|$ value to be $|link| = 64$ bits. To enable the Bloom filter to take up to 220 data, we additionally set $f = 8\ and\ 227$. This work determines the deduplication ρ, which is used to determine how successful deduplication is:

$$\rho = \left(1 - \frac{No.of\ all\ files\ in\ storage}{total\ no.of\ upload\ request}\right) * 100\% \quad (1)$$

5.1. Simulation results

The efficiency of deduplication for the four approaches is compared in Fig. 6. Consequently, the advantages of chunk-level deduplication over file-level deduplication are demonstrated. Our method has the third-best deduplication effectiveness, after Cui et al.'s scheme and other approaches having the worst. Although our method's deduplication efficacy is inferior to different directions, the difference is gradually closing, and our approach could even better fulfil access control. The CSP searches for duplication throughout all stored data

Doi : https://doi.org/10.54216/JCIM.140206
Received: January 10, 2024 Revised: March 13, 2024 Accepted: July 01, 2024

since the existing design disregards access control. To enhance the effectiveness of deduplication, the CSP only retains a single duplicate of each supplied data item. Because O_i and O_j are two authorization sets and share three other schemes, three linkages prevent them from being mutually contained. These connections are $O_j \subset O_j, O_i \subseteq O_j$ and O_i and O_j. The anticipated method solely considers the $O_i \subset O_j$ circumstance while deduplicating the duplicate to offer access control. For instance, the CSP only looks for duplicates in the datasets of U_3 and U_9 when U_1 wants to upload m. We take into account the first two scenarios in our system. In the same example, the CSP verifies the duplicate in the dataset of U_2 as a result of $O_2 \subset O_1$ in addition to the datasets of U_3 and U_9. As a result, our deduplication method is more effective than other method. All potential outcomes are included to equal the efficacy of the existing method while somewhat sacrificing access control. The private cloud, for instance, would spot the double if U_4 asked to upload m that U_5 had already uploaded. O_4 and O_5 are not mutually constrained. The private cloud must carry out the re-encryption procedure, and the symmetric key must be made available to these users for all $O_4 \cup O_5$ users to decode C_m generated by U_5 correctly. Access control is broken since an unauthorized user $U_6 \in / O_5$ can access the U_5 key.

In Fig 7, we show the computing costs side-by-side. It is clear from the figure that file-level deduplication incurs lower computational costs than chunk-level deduplication. Additionally, Operations related to deduplication (such as tag creation, duplicate search, and verification) are performed when the message's size ($|m|$) is minimal and significantly negatively influences computational efficiency. Fig 4 illustrates that the computational expenses associated with deduplication methods are the lowest in our system. As |m| rises, our scheme's computational efficiency is marginally worse than other approaches. The primary cause is that, although our technique generates the tag using encrypted data, the symmetric encryption approach continues to influence the computational expenses or $|m|$. As shown in Fig 5, our method's computational efficiency is significantly greater than that of notably for the chunk-leveled duplication. The primary determinants are the creation of tags, duplication verification procedures, and duplicate search time complexity.

Table 1: Performance comparison with Encryption time

File Size (MB)	Encryption Time (s)					
	RR-ECC (Proposed)	MCDD	MECC	DH	ECC	RSA
5	4.85	5.26	6.25	10.25	14.5	12.55
10	8.45	9.08	10.09	19.75	23	22.85
15	12.12	14.5	15.90	28.5	29	30.55
20	17.8	19	21	36.5	37	35.90
25	23.6	25.8	27.85	46.5	45	48.2

Table 2: Performance comparison with Decryption time

File Size (MB)	Decryption Time (s)					
	RR-ECC (Proposed)	MCDD (Proposed)	MECC	DH	ECC	RSA
5	4.54	5.11	5.25	12.2	13.5	11.4
10	7.95	9	10.1	18	22.6	20.5
15	11.8	13.8	16.5	26.3	29.3	28.3
20	16.78	18.5	20.1	35.5	38.5	36.5
25	24.5	26.2	25.3	45.6	45.5	48.2

Table 3: Performance evaluation based on Key Generation time (ms)

S. No	Approaches	Key Generation time (ms)	Security (%)	Deduplication (%)	Communication overhead (MB)	Storage cost (MB)	Computational cost (MB)
1	RR-ECC (Proposed)	400.5	98.5	90	1	0.1	100
2	MCDD	425.5	97	87	3	1	450
3	MECC	425.20	96	82	5	0.6	285

Doi : https://doi.org/10.54216/JCIM.140206
Received: January 10, 2024 Revised: March 13, 2024 Accepted: July 01, 2024

4	ECC	612.30	90	65	6	0.7	320
5	RSA	765.5	88	55	9	0.5	156
6	DH	856.3	86	68	10	0.2	189

The existing approach imposes significant computational costs on users by requiring them to generate the tag by interactively executing repeatedly utilizing the other technique for the same input. The complexity of the duplicate search time increases linearly with the number of stored files (i.e., $O(2n)$ pairing operations for n stored data). It checks for duplicates by carrying out time-consuming pairing operations. Our approach, however, does not need interaction protocols, and the duplicate search that goes along with it is based on a proven Bloom filter methodology, with the exact search's temporal complexity being $O(f)(f" n)$.

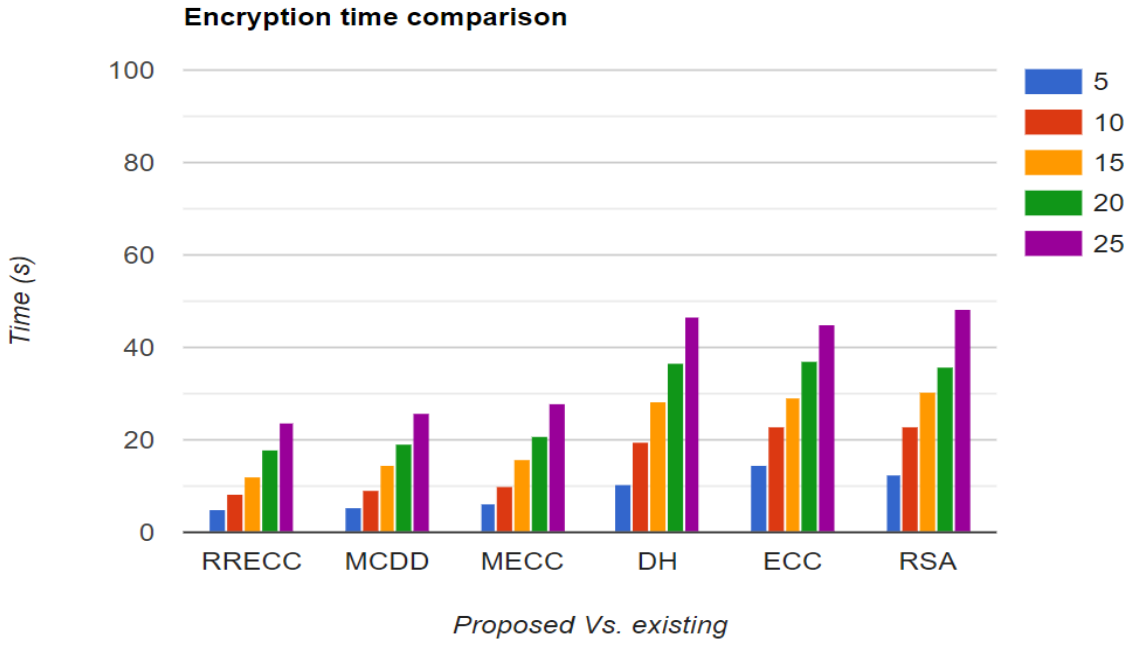

Figure 2: Encryption time comparison

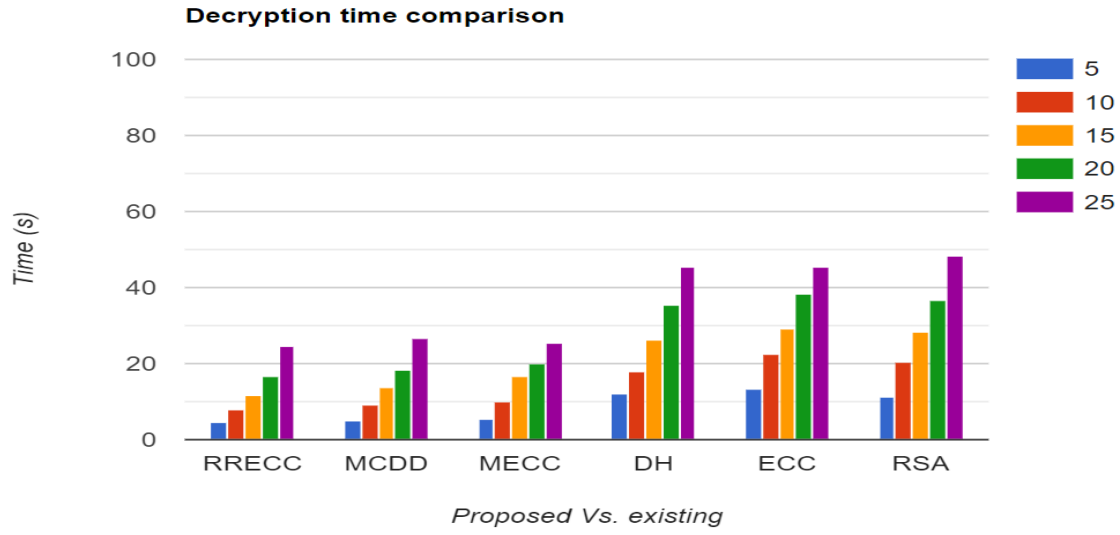

Figure 3: Decryption time

Doi : https://doi.org/10.54216/JCIM.140206
Received: January 10, 2024 Revised: March 13, 2024 Accepted: July 01, 2024

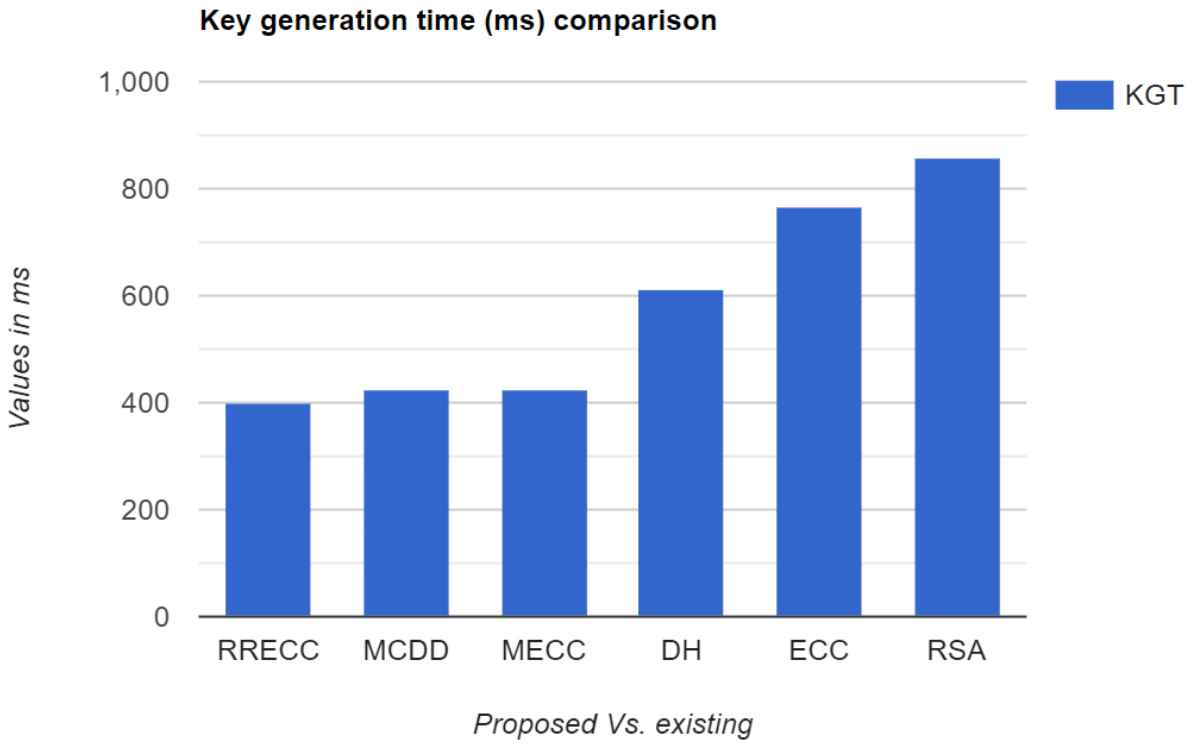

Figure 4: Key generation time (ms) comparison

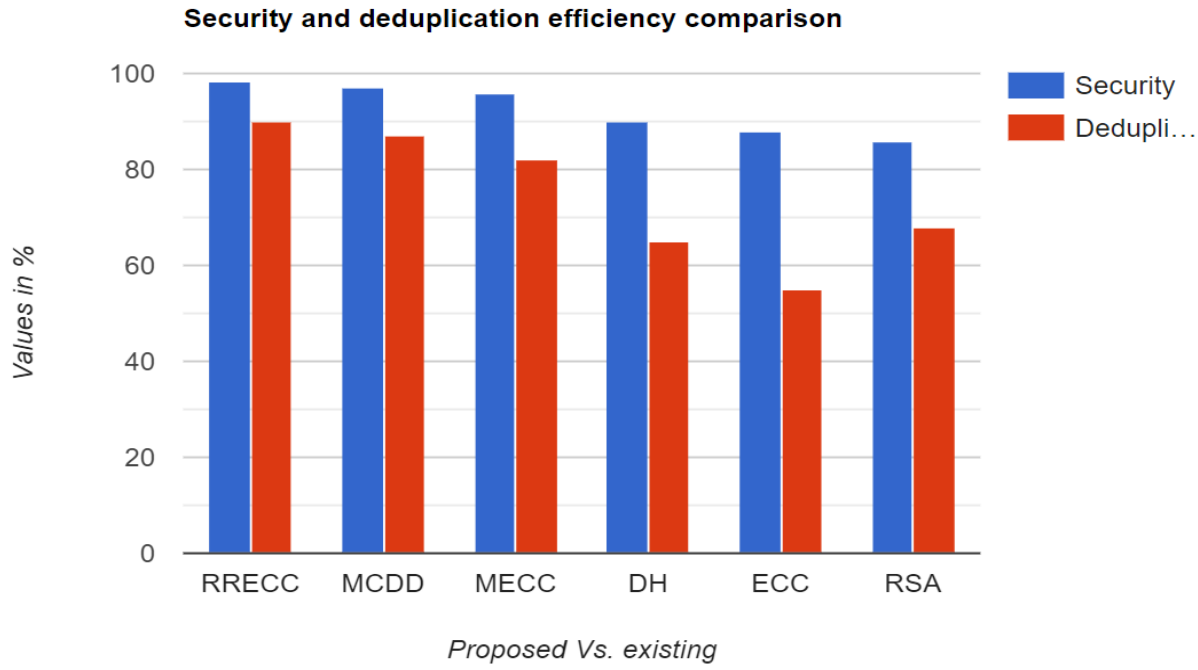

Figure 5: Security and data deduplication efficiency comparison

Doi : https://doi.org/10.54216/JCIM.140206
Received: January 10, 2024 Revised: March 13, 2024 Accepted: July 01, 2024

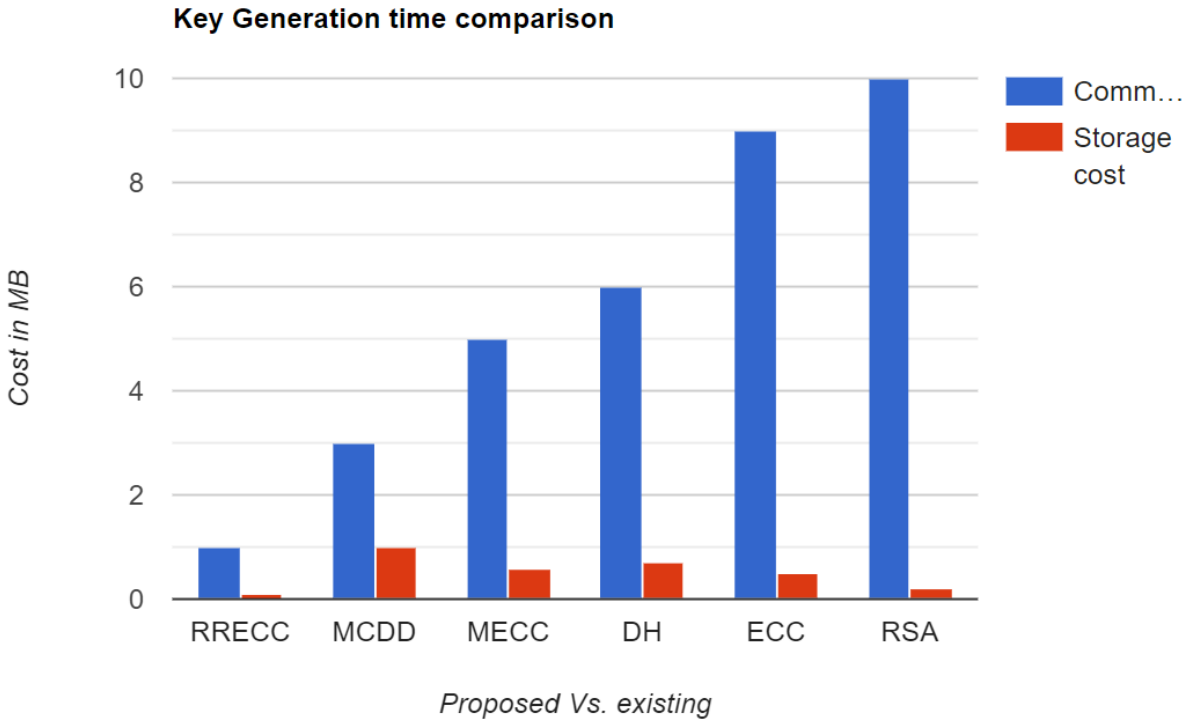

Figure 6: Communication overhead and storage cost evaluation

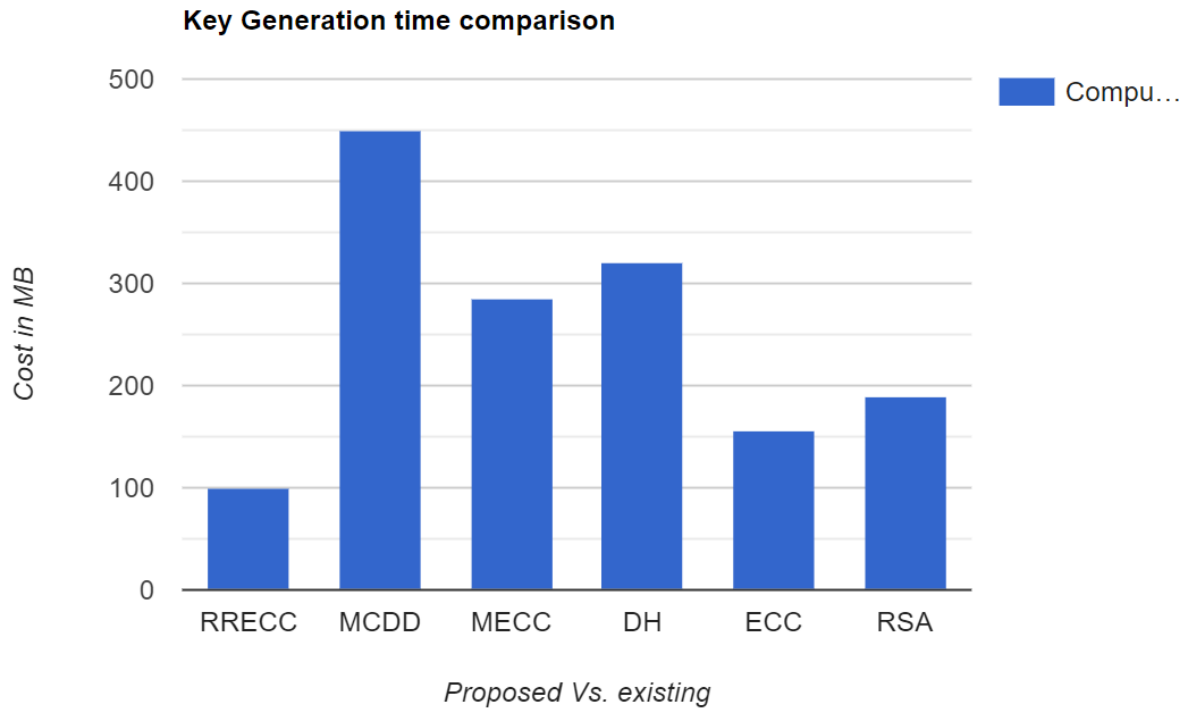

Figure 7: Computational cost

Doi : https://doi.org/10.54216/JCIM.140206
Received: January 10, 2024 Revised: March 13, 2024 Accepted: July 01, 2024

Fig 6 displays a comparison of communication overhead. The image unequivocally demonstrates that as message length |m| increases, so does the transmission overhead. The deduplication-related communication overhead predominates when |m| is modest. Our scheme's communication overhead is the smallest. Unlike the (chunk-level deduplication) technique [25], the scheme's communication cost is the highest. Similar to the study above, The strategy. would result in increased communication expenses due to interactions between the same-input-PAKE protocol and chunk-level deduplication, which would lead to more times this protocol is executed. Fig 6 demonstrates the communication overheads of our system, which are more significant than existing but lower than [25] and are essentially similar to [26] as $|m|$ grows. Furthermore, it illustrates that chunk-level deduplication is more successful at facilitating communication than file-level deduplication, suggesting that the efficacy of deduplication also impacts communication efficiency. Due to [25] solution's server-side deduplication, users continue transmitting outsourced data to the CSP even if duplicates exist. Despite the existence of duplicate data, it is evident that this deduplication method cannot reduce transmission costs. On the other hand, the three different approaches take advantage of when duplicates are present; client-side deduplication can reduce transmission costs. Furthermore, because Liu et al.'s method has the most excellent deduplication efficacy and the fewest ciphertext uploads, the communication efficiency associated with it is also the best. Comparing the costs of storage is shown in Fig 6. The efficacy of deduplication and message length $|m|$ significantly impact storage efficiency. Specifically, as $|m|$ rises, the encrypted data increasingly consumes more storage space. Additionally, as the efficacy of the deduplication increases, the amount of saved encrypted data reduces. Thus, it is clear from the graphic that chunk-level deduplication provides more storage efficiency than file-level deduplication. Comparing our method to the [25] way, Fig 7 demonstrates that our strategy minimizes the deduplication-related storage expenses when $|m|$ is sufficiently small. As shown in Fig 6, the existing scheme has the lowest storage efficiency when $|m|$ is big enough, whereas some current methods have the highest storage efficiency. The designs of [25] may be used to achieve the maximum levels of storage efficiency, whereas [25] design completely disregards access control. Our plan may achieve storage effectiveness while maintaining access control. Therefore, when compared to more recent works, when the given security constraints are guaranteed, our technique is, in fact, more efficient in terms of storage costs, communication costs, calculation costs, and deduplication efficiency. It shows our system's importance and value for enabling approved secure deduplication.

6. Conclusion

This research presents a user-defined access control efficient, safe deduplication system. In particular, our method does not need to deploy hybrid cloud architecture or install a second approved server to achieve the allowed deduplication. The CSP is the only entity in our system capable of managing access rights on behalf of data owners without endangering data privacy. Also included in our plan is the Bloom filter, which effectively completes the duplication check. Our technique can concurrently provide data privacy, access management, tag consistency, and defence against brute-force assaults, according to strict security assessments. Our method is effective in all four metrics, including the efficiency of file- and chunk-level deduplication, storage, communication, and computational costs.

References

[1] Miao, J. Wang, H. Li, and X. Chen, “Secure multi-server-aided data deduplication in cloud computing,” Pervasive and Mobile Computing, vol. 24, pp. 129–137, 2015.

[2] Shin, D. Koo, J. Yun, and J. Hur, “Decentralized server-aided encryption for secure deduplication in cloud storage,” IEEE Trans. Services Computing, vol. PP, no. 99, pp. 1–1, 2017.

[3] Liu, W. Sun, W. Lou, Q. Pei, and Y. Zhang, “One-tag checker: Messagelocked integrity auditing on encrypted cloud deduplication storage,” in 2017 IEEE Conference on Computer Communications, INFOCOM 2017, Atlanta, GA, USA, May 1-4, 2017, 2017, pp. 1–9.

[4] Jiang, T. Jiang, and L. Wang, “Secure and efficient cloud data deduplication with ownership management,” IEEE Trans. Services Computing, vol. PP, no. 99, pp. 1–1, 2017.

[5] Ambeth Kumar, V.D. (2017). Automation of Image Categorization with Most Relevant Negatives. Pattern Recognition and Image Analysis, 27(3), 371–379.

[6] Kumar, I., Kumar, A., Kumar, V.D.A. et al. (2022) Dense Tissue Pattern Characterization Using Deep Neural Network. Cogn Comput 14, 1728–1751.

Doi : https://doi.org/10.54216/JCIM.140206
Received: January 10, 2024 Revised: March 13, 2024 Accepted: July 01, 2024

[7] Yan, L. Zhang, W. Ding, and Q. Zheng, "Heterogeneous data storage management with deduplication in cloud computing," IEEE Trans. Big Data, pp. 1–1, 2017.
[8] Liu, K. R. Choo, R. H. Deng, R. Lu, and J. Weng, "Efficient and privacy-preserving outsourced calculation of rational numbers," IEEE Trans. Dependable Sec. Comput., vol. 15, no. 1, pp. 27–39, 2018.
[9] Yan, W. Ding, X. Yu, H. Zhu, and R. H. Deng, "Deduplication on encrypted big data in the cloud," IEEE Trans. Big Data, vol. 2, no. 2, pp. 138– 150, 2016.
[10] Ambeth Kumar, V.D. Vaishali,S. Shweta, B. (2015). Basic Study of the Human Foot. Biomedical and Pharmacology, 8(1), 435-444.
[11] Koo and J. Hur, "Privacy-preserving deduplication of encrypted data with dynamic ownership management in fog computing," Future Generation Comp. Syst., vol. 78, pp. 739–752, 2018.
[12] Tang, Y. Cui, C. Guan, J. Wu, J. Weng, and K. Ren, "Enabling ciphertext deduplication for secure cloud storage and access control," in Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security, AsiaCCS 2016, Xi'an, China, May 30 - June 3, 2016, 2016, pp. 59–70.
[13] Yang, H. Zhu, H. Lu, J. Weng, Y. Zhang, and K. R. Choo, "Cloud-based data sharing with fine-grained proxy re-encryption," Pervasive and Mobile Computing, vol. 28, pp. 122–134, 2016.
[14] Yan, L. Zhang, W. Ding, and Q. Zheng, "Heterogeneous data storage management with deduplication in cloud computing," IEEE Trans. Big Data, vol. 5, no. 3, pp. 393–407, Sep. 2019.
[15] Shen, Y. Su, and R. Hao, "Lightweight cloud storage auditing with deduplication supporting strong privacy protection," IEEE Access, vol. 8, pp. 44 359–44 372, 2020.
[16] Kumar, V.D.A., Sharmila, S., Kumar, A. et al. (2023). A novel solution for finding postpartum haemorrhage using fuzzy neural techniques. Neural Comput & Applic. 35(33), 23683–23696
[17] Liang, Z. Yan, X. Chen, L. T. Yang, W. Lou, and Y. T. Hou, "Game theoretical analysis on encrypted cloud data deduplication," IEEE Trans. Ind. Informat., vol. 15, no. 10, pp. 5778–5789, Oct. 2019.
[18] Liang, Z. Yan, R. H. Deng, and Q. Zheng, "Investigating the adoption of hybrid encrypted cloud data deduplication with game theory," IEEE Trans. Parallel Distrib. Syst., vol. 32, no. 3, pp. 587–600, Mar. 2021.
[19] Sathya Preiya, V., and V. D. Ambeth Kumar. (2023). Deep Learning-Based Classification and Feature Extraction for Predicting Pathogenesis of Foot Ulcers in Patients with Diabetes. Diagnostics 13(12), 1983.
[20] Chen, Y. Mu, G. Yang, and F. Guo, 'BL-MLE: Block-level message-locked encryption for secure large file deduplication," IEEE Trans. Inf. Forensics Security, vol. 10, no. 12, pp. 2643–2652, Dec. 2015.
[21] Kiss, J. Liu, T. Schneider, N. Asokan, and B. Pinkas, "Private set intersection for unequal set sizes with mobile applications," in Proc. Privacy Enhancing Technol., 2017, pp. 177–197.
[22] Pooranian, M. Shojafar, S. Garg, R. Taheri, and R. Tafazolli, "LEVER: Secure deduplicated cloud storage with encrypted two-party interactions in cyber–physical systems," IEEE Trans—Ind. Informat., vol. 17, no. 8, pp. 5759–5768, Aug. 2021.
[23] Hemamalini, Selvamani, and Visvam Devadoss Ambeth Kumar. (2022). Outlier Based Skimpy Regularization Fuzzy Clustering Algorithm for Diabetic Retinopathy Image Segmentation. Symmetry, 14(12), 2512
[24] Sherubha, "Graph-Based Event Measurement for Analyzing Distributed Anomalies in Sensor Networks", Sådhanå(Springer), 45:212, https://doi.org/10.1007/s12046-020-01451-w
[25] Sherubha, "An Efficient Network Threat Detection and Classification Method using ANP-MVPS Algorithm in Wireless Sensor Networks", International Journal of Innovative Technology and Exploring Engineering (IJITEE), Volume-8 Issue-11, September 2019
[26] Sherubha, "An Efficient Intrusion Detection and Authentication Mechanism for Detecting Clone Attack in Wireless Sensor Networks", Journal of Advanced Research in Dynamical and Control Systems (JARDCS), Volume 11, issue 5, Pg No. 55-68
[27] Piyush K. Pareek, Pixel Level Image Fusion in Moving objection Detection and Tracking with Machine Learning ",Fusion: Practice and Applications, Volume 2 , Issue 1 , PP: 42-60, 2020
[28] Shivam Grover, Kshitij Sidana, Vanita Jain, "Egocentric Performance Capture: A Review", Fusion: Practice and Applications, Volume 2, Issue 2 , PP: 64-73, 2020.

Doi : https://doi.org/10.54216/JCIM.140206
Received: January 10, 2024 Revised: March 13, 2024 Accepted: July 01, 2024

[29] Abdel Nasser H. Zaied, Mahmoud Ismail and Salwa El- Sayed, A Survey on Meta-heuristic Algorithms for Global Optimization Problems, Journal of Intelligent Systems and Internet of Things,Volume 1 , Issue 1 , PP: 48-60, 2020

[30] Mahmoud H.Alnamoly, Ahmed M. Alzohairy, Ibrahim M. El-Henawy, "A survey on gel images analysis software tools, Journal of Intelligent Systems and Internet of Things,Volume 1 , Issue 1 , PP: 40-47, 2021.

Doi : https://doi.org/10.54216/JCIM.140206
Received: January 10, 2024 Revised: March 13, 2024 Accepted: July 01, 2024