JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue
JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)
An International Scholarly Open Access, Peer-reviewed, Refereed Journal

Cooperative Cloud Storage Provides Safe and Effective Data Deduplication

[1]S.Sreekanth,[2] K. Priyanka,[3] Y.Gayathri, [4] T.siva Kumar reddy, [5] M.Nikhil kumar

[1,2,3,4,5] Gokula Krishna College of Engineering.

Abstract: Users' required bandwidth can be decreased and data redundancies in cloud storage can be effectively eliminated using data deduplication. However, because they have trouble disclosing information, the majority of earlier systems that relied on the assistance of a trusted key server (KS) are weak and constrained. inadequate defence against intrusions, high processing overhead, etc. Specifically, a single-point-of-failure occurs when the trusted KS fails, resulting in the system as a whole failing. insufficient protection against breaches, excessive processing overhead, etc. In particular, a single-point-of-failure happens when the system fails because the trusted KS fails. Theoretical analyses show that our SED has significant anti-attack capabilities, including resistance to collusion and brute-force attacks, and guarantees semantic security in the random oracle model. Furthermore, SED can efficiently remove data redundancies with no overhead in terms of computation, connection, or storage. SED's effectiveness and features enhance client-side usage. Furthermore, SED can efficiently remove data redundancies with no overhead in terms of computation, connection, or storage. SED's effectiveness and features enhance client-side usage.

Index Terms—Dynamic data, Data sharing, Single-point-of-failure,Cloud storage, JointCloud.

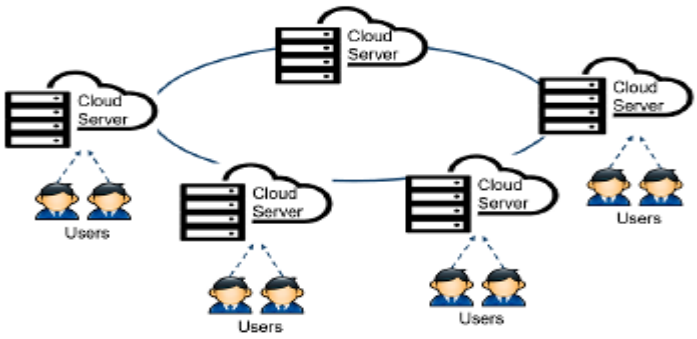
INTRODUCTION: cloud Storage is a platform that offers "pay-as-you-go" access to services and large-scale data storage. Nevertheless, a significant amount of superfluous data in cloud storage has severely consumed and wasted storage capacity. One useful technique to find and get rid of superfluous data is data deduplication [1]. Following that, the data is uploaded and saved in a single copy. As a result, data deduplication technology can lower client-side bandwidth requirements while increasing server-side space utilisation efficiency. It is currently frequently utilised to enhance user experience and conserve storage space in a variety of cloud computing services. With a framework made up of a key server (KS), cloud storage providers (CSPs), and users, the traditional data deduplication scheme and its variations guarantee security based on the reliable KS. Even worse, there's a chance that these time-tested plans will experience "platform lock-in" and single points of failure. The cloud storage system malfunctions and data outsourcing procedures cannot be used if the trusted KS fails. Recently, a brand-new cloud computing paradigm known as the Joint-Cloud computing system [8] was created to effectively address the problems listed above. JointCloud's network architecture is made up of users and numerous CSPs that offer different services. Users can connect with any of these clouds since they work together without the need for a reliable Keystone. to obtain computer services from them. It goes without saying that JointCloud's multilateral cooperation across different clouds enables it to offer effective cross-cloud services and meet the demands of globalised cooperative cloud services. It can also be included into a decentralised system . JointCloud computing has garnered significant interest from both academic and industrial sectors.

Large-scale data breaches that compromise billions of personal records frequently occur. In order to protect data confidentiality in cloud storage systems, the outsourced data are typically requested to be encrypted. Nevertheless, finding and eliminating the duplicate copies within the ciphertext domain is challenging. because various users' encrypted versions of the same plaintext using conventional encryption techniques have distinct ciphertexts.

Diverse variations of convergent encryption (CE) have been proposed to accomplish encrypted deduplication. These systems encrypt data using keys that are extracted from the data. In other words, the system is susceptible and the secret key is deterministic. There are specific security flaws, such as: 1) the tag exposes the plaintext's hash value, making it susceptible to the chosen-plaintext attack; 2) the ciphertext violates semantic security; 3) the predictable plaintext is not resistant to the brute-force attack; 4) users must bear a heavy computational burden in order to safeguard their data from malevolent attackers; etc. The prior approaches to resolving the aforementioned problems can be broadly divided into two groups, with the reasons behind their shortcomings given below. One kind of solution (e.g., [14], [15], [16], [17], [18]) limits the ways in which data is available and accessible in the traditional system paradigm, hence enhancing the security of data that is outsourced. That is, they manage encryption keys and guarantee the security of outsourced data access by using key management and access control approaches. However, because they share many keys and store auxiliary data, these methods often have high computational complexity and communication costs [19]. The alternative kind of solution creates new system architectures to enhance functionality and security. To do deduplication, users must communicate with several trustworthy key servers (KSs). It's clear that the frequent communication between the client-side and server-sides raises the likelihood that users may be attacked. Furthermore, the single-point-of-failure is still unsolvable by the majority of earlier techniques. Additionally, users' data in the JointCloud system may be dispersed among many clouds or stored in one of several clouds. The JointCloud technology does not support the direct application of traditional deduplication algorithms. We create a Secure and Efficient data Deduplication strategy (SED) based on the JointCloud system concept, driven by the aforementioned problems. Our SED's design objectives include efficiency, usefulness, and security. In particular, 1) our SED employs secure data outsourcing and deduplication without the need for a trusted key server; this should guarantee data confidentiality and integrity and guard against data intrusions. Functionality-wise, our SED ought to enable the sharing, access management, and renewal of data kept across several CSPs. In other words, the data owners have the ability to share and update their outsourced data from any location, but the non-owners are unable to effectively access or modify any outsourced data. 3) Our SED should execute tasks like uploading, deduplicating, updating, and sharing with little processing overhead in order to maximize efficiency. In this study, we present an effective and safe data deduplication strategy (SED) for the JointCloud storage system that does not rely on the trusted KS. Our SED's fully randomized tag generation method [11], which aids in duplicate identification and shields the outsourced data from collusion attempts, served as an inspiration for a few of its sub-algorithms.

1.1.Contribution

Unlike earlier deduplication techniques, our SED guarantees semantic security for both the tag and the ciphertext. There is no way for an adversary to obtain any valuable information from the ciphertext and tag. Furthermore, our SED is the first plan that allows for safe data transfer and updates. We create an encryption technique in our SED that facilitates sharing, updating, and data deduplication. To the best of our knowledge, SED is the first programme that takes into account the scenario in which the data owner gives authorised people access to their outsourced data. In particular, the cooperation of the involved CSPs generates a master encryption key. It guarantees key generation's adaptability and security. Data updating and sharing processes are also made easier by the data access control that is based on SED authentication. The efficiency of data deduplication is then increased by SED by combining the intra- and inter-deduplication approaches to remove duplicates from the JointCloud system. Theoretical analyses then show that the SED performs better in terms of functionality, robustness against collusion and assaults, data integrity, and confidentiality. To assess the difficulty through experimentation, SED is simulated and implemented using the Crypto++ , GNU, and PBC libraries. within Ubuntu. The evaluation's findings demonstrate SED's low computational overhead and efficiency.



To sum up, we mostly contribute the following: Semantic security is guaranteed by the proposed SED's tag generation and encryption algorithms. Additionally, SED is resistant to common assaults including collusion, manipulation, and brute-force attacks.

Secure deduplication is implemented by the SED independently of the trusted key server. Additionally, it allows cross-cloud data sharing and updating. Moreover, SED increases the scalability of the traditional deduplication approach and resolves the single-point-of-failure problem.

By using cryptographic libraries in our SED scheme trials, we confirm that our SED is effective and has a little processing burden. Our unique simulations of intra- and inter-deduplication show that these techniques can enhance the effectiveness of duplicates detection.

1.2 Organization

This is how the remainder of the paper is structured. In Section 2, the threat models and assessment criteria that are utilised to analyse the scheme are defined together with the system model. Basic primitives and notations are introducedinSection3.The concrete design of the suggested SED in the JointCloud storage system is shown in Section 4. The security of our SED is examined in Section 5. The SED efficiency assessments, encompassing compute, communication, and storage complexity, are presented in Section 6. Section 7 details the execution and assessment of our SED system. Section 8. examines relevant data deduplication works. Section 9 provides a succinct summary of the conclusion.

2 PROBLEM STATEMENT

In order to explain and thoroughly analyse our SED scheme, we specify some evaluation criteria in this section along with introducing the system model and threat model.

2.1 System Model

The JointCloud storage network architecture, as depicted in Fig. 1, is made up of two distinct network entities that are distinguished as follows. providers of cloud services (CSPs). It is an organisation with substantial resources and processing capacity sufficient to carry out protocols and offer distributed cloud computing services. User. (U). Any client who needs data stored in the cloud and wants to request data computing services like updating and sharing can be that client. Users may be businesses or lone customers. Additionally, users can be categorised into two groups. The original data is in the possession of the owner. In particular, the person who uploads the data into CSPs for the first time is referred to as the initial uploader, and users who upload the same data into CSPs later are referred to as subsequent uploaders. The other is the nonowner, who requests the plaintext of the data that has been outsourced from CSPs but does not have the original data. A user of JointCloud storage outsources data to clouds that operate in a dispersed and cooperative manner. Users interact with CSPs to access, retrieve, and manage the data that has been outsourced in terms of functionality. Operations typically involve sharing and updating, like adding and removing. Users no longer own their data locally, so it is vital to make sure that the data that has been outsourced is preserved and stored in clouds in an accurate and safe manner.

2.2 Dangerous Model

Users are generally trustworthy parties with the ability to regularly challenge clouds to verify the accuracy of the data that is outsourced. CSPs is a semi-trusted, sincere, but inquisitive entity that follows our protocol honestly but would like to take advantage of any opportunities to learn anything of value regarding the outsourced data. Nonetheless, CSPs are not allowed to alter or remove stored data without authorization because of the industry's accountability and reputation. Furthermore, we consider that users and clouds have dependable and authenticated point-to-point communication connections. Channel threats are outside the purview of this paper and will be the subject of future research. Two categories of enemies are taken into consideration based on the aforementioned assumptions.

1) Inside adversary. It can be a highly powerful CSP with administrative rights and semi-trusted status.
2) It would make use of all of its resources to obtain any pertinent data regarding the outsourced data.

2.3. Evaluation Criteria

In this part, we outline certain assessment standards for assessing the suggested SED, including a few efficiency and security considerations.

The confidentiality of the data that is outsourced should be guaranteed by the deduplication process. It is impossible for someone without the secret key to properly decipher the ciphertext. As long as the tag and ciphertext don't divulge any important information about the material, they should meet semantic security requirements.

Notation	Description
$\mathbb{G}$, $\mathbb{G}_T$	multiplicative cyclic groups with the prime order p and $\mathbb{G} =< g >, \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$.
$\mathbb{K}$	key space of symmetric encryption algorithm.
$f(\cdot)$	key derivation function f: $\mathbb{G}_T \to \mathbb{K}$.
$H(\cdot)$	collision-resistant hash function H: $\{0,1\}^* \to Z_p^*$.
$SE_x(m)$	encrypting m with a key $x \in \mathbb{K}$ by symmetric encryption algorithms.
τ	the tag of a file m and $\tau = e(a, b)$.
$\mathrm{Mat}(\tau_i, \tau_j) = 1$	τ_i is matched with τ_j, i.e., $e(a^i, b^j) = e(a^j, b^i)$.
$\mathrm{Mat}(\tau_i, \tau_j) = 0$	τ_i is not matched with τ_j, i.e., $e(a^i, b^j) \neq e(a^j, b^i)$.
$x \leftarrow X$	randomly and uniformly sampling an element x from a finite set X.
$\lvert x \rvert$	the length of x.
ID_j	the identity of the user $\mathcal{U}_j, j = 1, 2, \ldots$
K_i	the convergent key of $m_i, i = 1, 2, \ldots$
k	the auxiliary key.
L_i	the ownership list of $m_i, i = 1, 2, \ldots$
T^j_{intra}	a set of tuples $T_i, i = 1, 2, \ldots$, for intra-deduplication.
T_{inter}	a set of tuples $T^j_{intra}, j = 1, 2, \ldots$, for inter-deduplication.
T_{all}	the set of all tuples in the cloud storage, i.e., $\mathrm{T}_{all} = \mathrm{T}_{intra} + \mathrm{T}_{inter}$.
$\perp$	invalid or even error.

Table 1: Notations used in this paper

(2) The accuracy of the data. The inability of the outsourced data to be maliciously altered, erased, or destroyed in clouds is necessary for maintaining data integrity. Owners can challenge clouds to confirm the integrity of the data. Besides, even if they are a later uploader, any truthful user can effectively retrieve accurate data.

(3) Consistency of tags. The tag must meet the requirements of tag consistency (TC) in order to rule out the aforementioned types of integrity violations. These requirements include: 1) validity, which states that the tag is the same if and only if the data are the same; and 2) security, which requires the tag to be unforgeable and indistinguishable in random oracles.

(4) Takes aim at opposition. It is necessary for the deduplication method to withstand common attacks like the collusion attack (CA), tampering attack (TA), poison attack (PA), chosen plaintext attack (CPA), and brute-force attack (BFA). Moreover, the single-point-of-failure (SPoF) problem ought to be resolved by the deduplication strategy. In conclusion, even if a PowerPoint opponent launches the aforementioned attacks, they are unable to obtain any valuable information about the plaintext.

(5) Effectiveness. For the deduplication scheme to be client-side usable, it should have minimal computational, communication, and storage complexity.
(6) Usability. It needs to be possible for cloud data to be effectively shared, renewable, and access-controlled.

3 PRELIMINARIES

This section lists and reviews a few of the primitives and notations that we utilised in our SED. The notes are mentioned in the Table 1.

3.1 Bilinear Pairing

Let G =< g > and GT =\ g1 > be two multiplicative cyclic groups with a suitably large prime order p, given a security parameter _.
An injective function e: G G π! GT is a bilinear map, and it has the following characteristics:
1) Bilinearity: e(ua; vb) = e(u; v)ab; holds for all u, v 2 G and a, b 2 Zp.
2) Non-degeneracy: 6= 1 for e(g; g);
3) Computability: e(u, v) for u, v 2 G can be computed efficiently using an algorithm. Because e(ga; gb) = e(g; g)ab = e(gb; ga) and e(g; g) is a generator of GT, it can be observed that e({; }) is symmetric.
The Computational Diffie-Hellman (CDH) problem is defined as, Given (g, ga, gb) and an unknown random a, b 2 Z^p, it is difficult for an adversary operating in probabilistic polynomial-time (PPT) to compute gab 2 G unless the discrete logarithm (DL) issue in G is solvable.
Definition 3.3: Assumption of Decisional Diffie-Hellman (DDH).
Given g, ga, gb, gab, gc 2 G and unknown random elements a, b, c 2 Zp, it is difficult for any PPT opponent to distinguish tuples (ga, gb, gab) from tuples (ga, gb, gc).
Definition 3.4: Assumption of Decisional Bilinear Diffie-Hellman (DBDH).
Given (g, ga, gb, gc) 2 G, e(g; g)abc 2 GT, and a random element Q 2 GT, together with unknown random elements a, b, and c 2 Z~p, it is difficult for any PPT adversary to discern the value of e(g; g)abc from the random element Q.
Definition 3.5 RSA Assumption.
It is difficult for any PPT adversary to recover M without private key d given the RSA public parameter (N; e), the ciphertext C = Me mod N, and the modulus N = pq, where p and q are two different large prime numbers, d is the modular multiplicative inverse of constant e, and the modulus is sufficiently large and randomly generated. Let (N; p; q; e; d) RSAKGen({) be the RSA scheme's key generation algorithm.

3.2 Deduplication Techniques

There are two primary methods for deduplication that efficiently remove redundant data from the traditional cloud storage architecture. Intra-deduplication is one that only takes into account the fact that the data owner has contracted out the same KS to handle their data. That improves the backup system's efficiency. The other is called inter-deduplication, and it takes into account data that has been outsourced through several KSs by various data owners.
It works well for a storage system where there are a lot of duplicates in the data that is outsourced to several owners. With the JointCloud system, we have redefined them. The scenario where the data owner has outsourced data to a cloud server is taken into account by intra-deduplication. Inter-deduplication takes into account the data owner's outsourcing of data across several clouds.

4.THE PROPOSED SCHEME

The proposed SED scheme is outlined in this section. After providing an outline of SED, we go over how it is constructed in concrete.

4.1. Overview

It is suggested that the SED be used to effectively and safely remove redundancies and enable certain dynamic data operations, such as data exchange and updating, in the JointCloud storage system. SED = f System setup, Data outsourcing, Data access, Data update, and Data sharing g are the five components that make up SED. Users and clouds carry out mutual authentication and initialization at the first step of the system setup [31] to confirm each other's legitimacy.
Users then encrypt their data and send it to the JointCloud system. Computing and storage services are offered by CSPs in response to user requests. During the data outsourcing phase, the data are kept and encrypted.

Redundancies in the ciphertext are removed using deduplication techniques. After that, users can complete the data access step to retrieve their data. Furthermore, accurate data recovery is only possible for those who are data owners or who possess critical information.

Additionally, users can use the Data Update and Data Sharing phases to update and share their data that is stored in clouds. In particular, the update processes involve deletion and change. The insertion process is not repeated since it is treated as a specific instance of a new data block creation and is saved with a designated location label. Keep in mind that SED primarily addresses the JointCloud system's deduplication. Therefore, the cloud collaboration method—an intriguing subject for our future research—is not covered in detail in this study.

4.2. The Detailed Construction

The five steps of our SED scheme are: system setup, data sharing, data update, data access, and data outsourcing.

5.SECURITY ANALYSES

We examine SED's security in this section while adhering to the security specifications listed in Section 2.3.

5.1. Data Confidentiality

The outsourced data is encrypted as C = (C_1;C_2) in the proposed SED technique. Evidently, the AES algorithm that produced C1 is anti-brute force and semantic security. C2 is the encrypted version of C1's secret key. As a result, we talk about the security of the C2 algorithm. Our SED satisfies the semantic security if it is semantically secure. The random oracle model proves the semantic security of the encryption technique that generates C2, as per Theorem 5.1. In other words, the ciphertext C cannot provide any meaningful information about plaintext to any PPT attacker. Therefore, data secrecy is guaranteed by the suggested SED method.

Theorem 5.1

Under the decisional bilinear DBDH assumption, the algorithm generating C2 in the suggested scheme SED is semantically safe, even in the event that a PPT opponent connects C2 with the tag.

Proof.

Semantically secure is a method that generates C2 if it satisfies the ciphertext indistinguishability against chosen-plaintext attack. With respect to the challenge oracle, any PPT adversary selected two distinct plaintexts (R1;R2) 2 GT, where jR1j = jR2j. Next, a uniformly random bit b 2 f0,1g is chosen by the challenge oracle, which then gives the adversary a ciphertext C2 b and tag {b. The adversary makes an estimate of b. If the opponent prevails, they can figure out the plaintext. The following is a presentation of the cryptographic game details.

The adversary receives a ciphertext C2 b = e(g; g)vb{{Rb encrypted with the secret key vb = h \ sb from the challenge oracle, which chooses a uniformly random bit b 2 f0,1g.4) The opponent is free to ask for as many encryptions as they like. Lastly, an estimate for the value b is provided by the opponent. 5) The value e(g; g) vb is difficult for any PPT opponent to differentiate from a random element in GT, based on the DBDH assumption. Furthermore, it is difficult for any PPT adversary to discern between the random element and the ciphertext e(g; g)vb~ ~Rb. Since {({) is a tiny function with the security parameter {, the adversary can predict the number b with only a small advantage, that is, 1 2 + {({).

5.2. Data Integrity

As per the definition given in Section 2.3, data integrity necessitates that the data that is outsourced in clouds cannot be deliberately altered, erased, or destroyed. Because of the industry's reputation and accountability, they do not alter or remove the stored data without authorization based on presumptions about cloud servers (i.e., insider adversaries).Un-authorised users are the outside enemy.

Their IDs are not included in the data ownership list, therefore thus are unable to pass authentication. CSPs faithfully carry out the protocol and forbid any unauthorised users from successfully manipulating data. Additionally, owners can use data access to challenge clouds to confirm the integrity of their data. The accuracy of SED suggests that data can be effectively recovered by any truthful owner. To sum up, data integrity can be guaranteed via the SED.

5.3.1 The Brute-Force Attack

An adversary may attempt to use the brute-force attack (BFA) to recover the data m with the deduplication tuple ({; k;C). Let us assume that an attacker is requesting data access from CSPs by repeatedly computing a hash value. The enemy starts by creating a dictionary of candidates, such as { = fm1;m2; :::;mng with size n. After that, until Mat({i; {) = 1, where i { n, the attacker attempts to create the tag f{1; {2; :::; ng to seek data access. The opponent then confirms that the guesses about the data were accurate. Using the hash H(mi), he or she decodes the ciphertext C and verifies that H(D(C))?= H(mi), where D is the decryption algorithm. The opponent receives accurate facts if it holds.

Nevertheless, the data space m 2M, or { ~ M, is sufficiently vast for uncertain data. Any PPT opponent would find it difficult to guess the aforementioned mi in our SED since the tag and ciphertext meet the requirements of Theorems 5.1 and 5.2 for semantic security. Additionally, SED can lessen the impact of an attack on predictable data by capping the quantity of requests for the same data sent within a predetermined window of time. In other words, a user can request access to the data m by a maximum of q queries. With query limitation, the SED is thereby protected from a brute force assault.

5.3.2 Poison Attack

One popular technique employed by an external adversary to compromise data integrity is the poison assault (PA). We talk about the poisoning attack in our SED plan. Assume that an adversary with the identical data (m) creates ciphertext C0 maliciously using m0 and uploads ({;C0) to CSPs, where the tag ~ = T(m) and (m0 6= m). In this instance, CSPs are unable to identify the phoney data C0. Nonetheless, in order to ensure the accuracy of their data, other truthful owners can ask for data verification. To be more precise, suppose that the enemy sources data C0 = (C10;C20) with the tag { = (gr; grH(m)). If the backup key k is accurate, that is to say,

The other owners can decode C to obtain m0 if k = gH(m)~s ~ gH(m). However, the verification e(g; grH(m))?= e(gr; gH(m0)) cannot be passed by the datam0. If the backup key is the fraudulent one, that is, is given back to the owners, and the verification fails as well. As a result, it is simple to identify that the data C0 is distorted. The SED is capable of both detecting and thwarting poisoning attacks.

5.3.3 Tampering Attack

Users (the outside adversary) and CSPs (the inside adversary) can both initiate tampering attacks (TAs). The danger model states that CSPs do not alter the data that is outsourced because of accountability and company reputation. Data stored in clouds may be tampered with by an external opponent. Imagine a situation where a data-accessing adversary tries to alter the data so that other authorised owners are unable to accurately decode it. Even still, the adversary cannot alter the data in our scheme SED, even though he is free to request any operation.

due to the fact that changing the data's ownership really implements the update actions. The activities do not alter the data belonging to other owners, regardless of how the attacker updates the data. Furthermore, it is difficult for an unauthorised user to alter data when they are not authorised to do so. Due to the unapproved user's inability to pass the authentication and It is not within his or her authority to alter data. As a result, the suggested SED scheme is resistant to tampering attacks.

5.3.4 Collusion Attacks

A detailed discussion is held of three instances of collusion attacks (CA) involving users and CSPs. The goal of collusion assaults is to access user-outsourced data material beyond the purview of authorized users.

Case 1: CSPs engage in mutual collusion. CSPs can obtain the private key needed to encrypt data by banding together. In particular, CSPs attempt to decode the ciphertext given the auxiliary key k and the secret key the tag. CSPs, however, are unable to retrieve R from C2 in the absence of the crucial data go. As a result, our system SED also withstands the CSPs' collusion attack.

Example #2: Users conspire with one another. Imagine a situation where a few people who do not own data m may conspire to obtain data m. Assume that n users—that is, the outside adversary—collude with one another and that they are not the owners of data m. They then obtain the convergent keys that match the data. However, in order to assure the ciphertext randomization, each user is required to select (s;R) at random when encrypting data. These users who collaborated are unable to learn anything about the data m's secret keys. Therefore, the SED method also withstands user collusion attacks.

Case 3: CSPs and users conspire. The goal of this collusion is to access data beyond the user's privileges, or outside the adversary. Adversaries come in two varieties. 1) The owners of the data who, as a result of deletion or modification actions, no longer possess it. Because they can immediately expose their plaintext to CSPs, SED

views the second instance as a little collusion. Furthermore, other users' data is still not accessible through this collusion.

1.non-owners who lack access to the data's hash value and secret key. In this case, our key discussion topic is the collusion attack. Assume that l CSPs and n users conspire. In other words, users' convergent keys, users' deduplication tuples T; Tn, where T is the tuple of data m and is users' tuple, are encrypted using the secret key.

Even though they own (~; ~; k;C), users and CSPs do not still obtain the right convergent key K of data m. It is not practicable to retrieve R without the essential knowledge of s and h as a result, the ciphertext cannot be successfully decoded by them.

5.3.5. Single-Point-of-Failure

We address whether our SED functions in the event of a single-point-of-failure (SPoF) in this section. Let's look at the following situations where a CSP Pi 2 P fails during our SED's phases.

1.Configuring the system. Examining this phase's process makes it clear that Pi's absence has no effect on the procedures. because parameter creation and authentication can still be handled by the operational CSPs.

2.Outsourcing of data. During this stage, the data was encrypted using the public key g{. The user can upload data in accordance with the protocol and encrypt it, but they are unable to remove the redundant data that failed Pi stored. In other words, functional CSPs can securely store the data, but in this scenario, the deduplication ratio can be lower. Furthermore, this failure has less of an effect on our SED than it did on the earlier plans that were based on the conventional cloud storage system. since under this situation data storage and access services cannot be provided by the prior methods.

3. Access to data. The verified user can decrypt their data on their own during this step. A correctly authenticated user can receive data tuple ({; k;C) if the data is held by operational CSPs. The data from the ciphertext C may be readily decoded by the user with the public key g{. The user cannot access the data pair if the data is saved by the malfunctioning Pi. This problem affects cloud storage methods that are based on the conventional cloud as well. Thankfully, we may request that the permissions to access the failing CSP's storage in the JointCloud system be granted to the operational CSPs in order to resolve this issue.

Furthermore, similar to the data access phase, the problem brought on by Pi failure can be resolved during the data update and sharing phases.

Schemes	Communication overhead		Storage overhead		
	Upload data components	Download data components	Ciphertext size	Key size	Tag size
DupLESS [13]	ID+T+C_m+K	C_m+C_k	$\|m\|$	$\|H\|$	$\|\kappa\|$
Hur et al. [19]	ID+T+CK+C_m+K+A	C_m+CK+K+T+A	$\|m\|+\|H\|+(n-l)log\frac{n}{n-l}\|\kappa\|$	(log$n$+1)$\|H\|$	$\|H\|$
Shin et al. [20]	ID+T+2Sig+C_m+C_k+nA	C_m+C_k+A	$\|m\|+\|\mathbb{G}_T\|+N\cdot\|\mathbb{G}\|$	$\|\mathbb{G}\|$	$\|\mathbb{G}\|$
Our SED	ID+T+C_m+C_k+A	C_m+C_k+A	$\|m\|+\|\mathbb{G}_T\|+\|\mathbb{G}\|$	$\|\mathbb{G}\|$	$\|\mathbb{G}\|$

Table 3: Communication and Storage Complexity Comparison

Schemes / Performance	DupLESS [13]	Hur et al. [19]	Shin et al. [20]	Our SED
Confidentiality	●	●	●	●
integrity	●	●	●	●
Tag consistency	○	●	●	●
Anti-CPA	●	●	●	●
Anti-BFA	●	●	●	●
Anti-PA	○	○	●	●
Anti-TA	●	●	●	●
Anti-CA	○	○	●	●
Anti-SPoF	○	○	○	●
Effeciency	◐	◐	◐	●
Dynamic update	○	●	○	●
Data sharing	○	○	○	●

* ●: Yes. ○: No. ◐: Partly.

Table 4: Advantages of over Prior Schemes

There is no repetition of the first two phases. In conclusion, our SED cannot be rebuilt from scratch due to the failure and revocation of a single cloud server.

6.EFFICIENCY

In this part, we examine the suggested scheme's efficiency. The efficiency will be assessed in relation to communication and storage overhead, as well as computational complexity.

The computational complexity comparison between our scheme SED and a few common schemes [13], [19], and [20] is displayed in TABLE 2. These procedures involve the creation of keys, tags, ciphertext, and deduplication. The comparison findings show that our SED's key and tag generation complexity is H+3Exp, which is clearly lower than that of the earlier methods. Additionally, encryption has a lower complexity than most systems. Additionally, when the number of users increases, the encryption difficulty of schemes [19], [20] will also increase. In summary, our SED satisfies the real client-side needs of the cloud storage system by executing the data deduplication in a concise manner with low computational complexity.

TABLE 3 also provides a summary of the comparative results between the computations of communication overhead and storage overhead. It demonstrates that our approach has lower communication and storage overheads than the majority of earlier schemes. It appears that most techniques require uploading the ciphertext and tag using an additional.

The suggested SED was then contrasted with a few earlier standard schemes [13], [19], and using the assessment standards that were previously established in Section 2.3. TABLE 4 is a summary of the comparison results. The thorough comparison demonstrates that our system is superior to other schemes in terms of security, effectiveness, and utility.

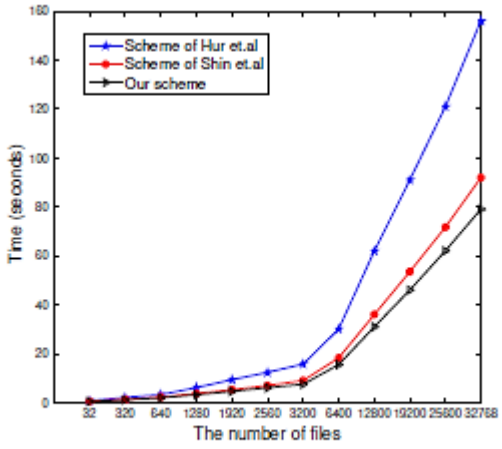


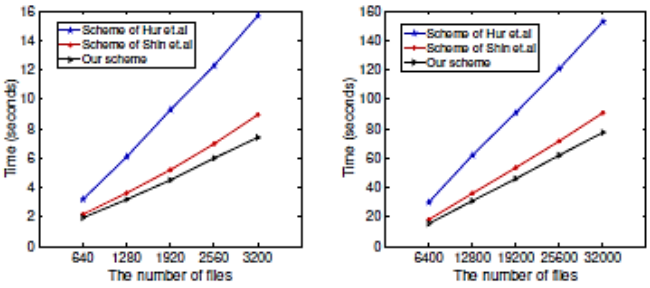
Fig. 2: The running time for different amounts of files.

Our SED offers excellent scalability and can remove data redundancies without the assistance of the centralised key server thanks to JointCloud's network architecture. Because of this, SED is appropriate for decentralised systems, something that the prior plan based on the traditional cloud storage paradigm was unable to accomplish.

7.IMPLEMENTATION AND EVALUATION

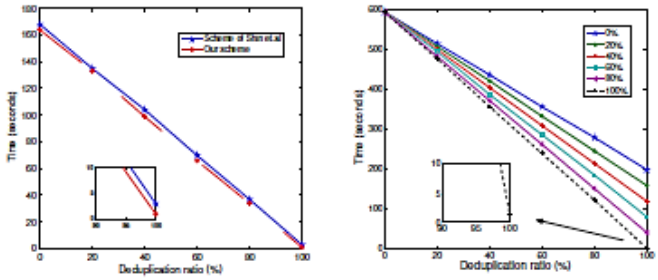
7.1. Implementation

Our SED is built on the Crypto++, GNU , and PBC libraries and is implemented in the 64-bit Ubuntu 16.04 LTS version. The 128-bit security parameter is used in the implementation of the symmetric encryption technique AES and the cryptographic hash function SHA1. We do not take into account in this simulation how bandwidth impacts communication between the cloud server and the client. We model relevant deduplication processes on a PC with a 3.3GHz Intel Core i5-4590 CPU and 3.8GB of RAM. Keep in mind that the same conditions apply to the implementation of the compared scheme. Furthermore, we evaluate whether our approach can effectively remove duplicates rather than focusing on the speed of duplicate detection, since the paper primarily focuses on the security and efficiency of deduplication. In other words, we don't research the traversal algorithm to swiftly identify duplicates. As a result, we confirm the tags using the same inquiry method (i.e., one by one). to verify the copies of the comparing schemes and our own system.



(a) Δ = 640 files (b) Δ = 6400 files

Fig. 3: The running time for different files increment (Δ).



(a) 320 files with different DR (b) Different intra-DR

Fig. 4: The running time for different deduplication ratio (DR).

7.2. The Efficiency of Data Outsourcing

We take into account that consumers outsource their data during various stages of encryption, deduplication, and key and tag generation. To do fine-grained deduplication, which is more space-efficient than file-level deduplication, the file is typically separated into blocks. As a result, if no duplicates in the file-level are found in this simulation, we choose files from the database to conduct file-level deduplication before doing block-level deduplication. Initially, 32768 test files with a 20% duplicate rate are chosen; each file has a 32KB size and can be broken into blocks of 32B in size.

The parameters, tag, and ciphertext are generated by the system on its own for every test file and block. The time it takes for schemes to outsource 32768 files is then tested. The effectiveness of these strategies is assessed using the test results. Fig. 2 presents the comparisons. We discover that as the number of files increases, so does the amount of time needed to implement each scheme's data deduplication. Our SED, however, is more effective than the earlier plans.

Two instances of the incremental file outsourcing execution times for various strategies are displayed in Fig. 3. The results of the increment ~ are shown in Figs. 3(a) and 3(b), respectively. The increment ~ = 640 files indicates that the number of outsourced files increases by 640 for each deduplication. It goes without saying that these systems' execution times grow as the number of files increases isometrically. Moreover, our scheme grows at the slowest rate. In conclusion, the suggested SED plan performs better in terms of data outsourcing efficiency than other earlier schemes.

Figure 4(a) displays the schemes' running times, which indicate that when the DR rises, the schemes' running times decrease. In addition, the rates of descent are almost identical. since we have it configured to stop the process when duplicates are discovered. The traversal time increases with decreasing duplication ratio. For instance, duplicates can be found at the file-level when the duplication ratio is 100% (i.e., file-level duplicates).

Furthermore, we assess the effectiveness of both inter- and intradeduplication. Initially, we choose two file sets, F2 and F3, which contain various 352 and 704 files from the database, respectively. Let's say that CSP Pi stores a user's backup file F2, and other CSPs store the set file F3. Next, in order to do deduplication, the user chooses new files set F1, which consists of 352 files with various deduplication ratios. Let the ratio of duplicates found in F2 be the intra-deduplication ratio (intra-DR), and the ratio of duplicates found in F2 be the inter-deduplication ratio (inter-DR).

For instance, we use six cases to carry out deduplication for F1 with the 20% DR. 1) All duplicates are present in F3, with intra-DR = 0% and inter-DR = 100% indicating that there are no duplicates in F2. 2) 20% for intra-DR and 80% for inter-DR. 3) 40% for intra-DR and 60% for inter-DR. 4) 40% is inter-DR and 60% is intra-DR. 5) 20% is the inter-DR and 80% is the intra-DR. 6) 100% for intra-DR and 0% for inter-DR. Comparably, for F1, there are six examples with DRs of 0%, 40%, 60%, 80%, and 100%, in that order. For the aforementioned scenarios, we performed deduplication, and the outcomes are displayed in Fig. 4 (b).

It goes without saying that a greater intra-DR corresponds to a lower execution time. The method that combines intra- and inter-deduplication may effectively reduce the amount of time spent on deduplication, as shown by the trend of each curve and their rate of decline in Fig. 4(b). Regarding the data deduplication protocol, it is important.

7.3. Deduplication Ratio

Since both the suggested technique and Shin et al.'s scheme remove duplicates by combining intra- and inter-deduplication, we assess their respective efficaciousness in this section. First, we assess the effectiveness using

various deduplication ratios (DR). In particular, 320 test files with various DRs, such as 0%, 20%, 40%, and 100%, have been chosen.

Figure 4(a) displays the schemes' running times, which indicate that when the DR rises, the schemes' running times decrease. In addition, the rates of descent are almost identical. since we have it configured to stop the process when duplicates are discovered. The traversal time increases with decreasing duplication ratio. For instance, duplicates can be found at the file-level when the duplication ratio is 100% (i.e., file-level duplicates).

Furthermore, we assess the effectiveness of both inter- and intradeduplication. Initially, we choose two file sets, F2 and F3, which contain various 352 and 704 files from the database, respectively. Let's say that CSP Pi stores a user's backup file F2, and other CSPs store the set file F3. Next, in order to do deduplication, the user chooses new files set F1, which consists of 352 files with various deduplication ratios. Let the ratio of duplicates found in F2 be the intra-deduplication ratio (intra-DR), and the ratio of duplicates found in F2 be the inter-deduplication ratio (inter-DR). For instance, we use six cases to carry out deduplication for F1 with the 20% DR.

1) All duplicates are present in F3, with intra-DR = 0% and inter-DR = 100% indicating that there are no duplicates in F2. 2) 20% for intra-DR and 80% for inter-DR. 3) 40% for intra-DR and 60% for inter-DR. 4) 40% is inter-DR and 60% is intra-DR. 5) 20% is the inter-DR and 80% is the intra-DR. 6) 100% for intra-DR and 0% for inter-DR. Comparably, for F1, there are six examples with DRs of 0%, 40%, 60%, 80%, and 100%, in that order. For the aforementioned scenarios, we performed deduplication, and the outcomes are displayed in Fig. 4 (b). It goes without saying that a greater intra-DR corresponds to a lower execution time. Based on each curve's trend and rate of decline in Fig. 4(b), The time spent on deduplication can be significantly reduced by using this technique, which combines intra- and inter-deduplication. Regarding the data deduplication protocol, it is important.

8.RELATED WORK

Numerous deduplication techniques are now under consideration. We go over the most recent deduplication schemes in this section.

One of the primary methods for ensuring data security in deduplication is convergent encryption, which helps shield outsourced data from rogue and unreliable CSPs. A rudimentary approach known as message-locked encryption (MLE) was formalised by Bellare et al. [10].

Variants [12], [13] were suggested in light of Bellare's research. But because the keys used to encrypt files are extracted from the files themselves, these MLE-based methods were vulnerable to a wide range of possible threats. For bounded message distributions, Abadi et al. [11] developed a fully randomised scheme and a deterministic encrypted method based on the non-degenerate efficiently computable bilinear map.

on the bilinear map that is non-degenerate and efficiently computable. A method for achieving dependable key management in deduplication was given by Li et al. in [15]. Subsequently, Jiang et al. [14] presented a safe deduplication method using the randomised tag. They did not, however, address the necessity of updating the data. Hur et al. later examined dynamic ownership management for safe deduplication in [19], and in order to accomplish deduplication, each client must store the keys that are situated along the path of a binary tree that contains all of the keys.

Secure deduplication with key management based on secret sharing strategies was proposed by Li et al. in [15], . Following that, a decentralised server-aided encryption for deduplication was created by Shin et al. However, it required several exchanges between users and KS, which provided the attackers with chances to obtain valuable information from the exchange

Miao et al. presented a secure deduplication method for multiserver-aided in . Quick and effective content-defined chunking was created by Xia et al. [3] to accomplish fine-grained data deduplication. In order to save space and lower layer restoration overhead, Zhao et al. presented a deduplication approach based on a Docker registry architecture. Security-wise, [18], covered common attacks against modern deduplication techniques.

CONCLUSION

Without the assistance of the trusted KS, we have developed a safe and effective technique SED for data deduplication in this study. Based on the CDH issue in the JointCloud storage system, the suggested SED has decreased the client-side computation and communication overhead and increased efficiency. Its succinct tag generation and encryption techniques meet semantic security and tag consistency requirements (security and validity), respectively.

Additionally, SED addresses the single-point-of-failure of KS in the traditional cloud storage architecture and increases scalability. SED is very resilient to common threats like brute-force attacks and rogue CSPs working together with unauthorised users. Additionally, SED enhances usefulness and usability by supporting dynamic data actions including sharing, editing, and deleting. To the best of our knowledge, SED is the first scheme that takes into account the scenario in which the data owner shares their contracted data with authorised users.

REFERENCES

[1] P. Christen, "A survey of indexing techniques for scalable recordlinkage and deduplication," IEEE Transactions on Knowledge andData Engineering, vol. 24, no. 9, pp. 1537–1555, 2012.
[2] G. Jia, G. Han, J. J. P. C. Rodrigues, J. Lloret, andW. Li, "Coordinate memory deduplication and partition for improving performancein cloud computing," IEEE Transactions on Cloud Computing, vol. 7,no. 2, pp. 357–368, 2019.
[3] W. Xia, X. Zou, H. Jiang, Y. Zhou, C. Liu, D. Feng, Y. Hua, Y. Hu,and Y. Zhang, "The design of fast content-defined chunking fordata deduplication based storage systems," IEEE Transactions onParallel and Distributed Systems, vol. 31, no. 9, pp. 2017–2031, 2020.
[4] J. Li, J. Li, D. Xie, and Z. Cai, "Secure auditing and deduplicatingdata in cloud," IEEE Transactions on Computers, vol. 65, no. 8,pp.2386–2396, 2016.
[5] L. Liu, Y. Zhang, and X. Li, "Keyd: Secure key-deduplication witidentity-based broadcast encryption," IEEE Transactions on Cloud Computing, pp. 1–1, 2018.
[6] J. Ni, K. Zhang, Y. Yu, X. Lin, and X. S. Shen, "Providing task allocationand secure deduplication for mobile crowdsensing via fog computing," IEEE Transactions on Dependable and Secure Computing, pp. 1–1, 2018.
[7] Y. Zheng, X. Yuan, X.Wang, J. Jiang, C.Wang, and X. Gui, "Towardencrypted cloud media center with secure deduplication," IEEETransactions on Multimedia, vol. 19, no. 2, pp. 251–265, 2017.
[8] H. Wang, P. Shi, and Y. Zhang, "Jointcloud: A cross-cloud cooperationarchitecture for integrated internet service customization," in 2017 IEEE 37th International Conference on Distributed ComputingSystems, 2017, pp. 1846–1855.
[9] K. Huang, X. Zhang, Y. Mu, F. Rezaeibagha, X. Wang, J. Li, Q. Xia,and J. Qin, "Eva: Efficient versatile auditing scheme for iot-baseddatamarket in jointcloud," IEEE Internet of Things Journal, vol. 7,no. 2, pp. 882–892, 2020.
[10] M. Bellare, S. Keelveedhi, and T. Ristenpart, "Message-lockedencryption and secure deduplication," in International Conference on the Theory and Applications of Cryptographic Techniques, 2013, pp.296–312.
[11] M. Abadi, D. Boneh, I. Mironov, A. Raghunathan, and G. Segev, "Message-locked encryption for lock-dependent messages," in Advances in Cryptology – CRYPTO 2013. Springer Berlin Heidelberg, 2013, pp. 374–391.
[12] M. Bellare and S. Keelveedhi, "Interactive message-locked encryption and secure deduplication," in Public-Key Cryptography – PKC 2015, J. Katz, Ed. Springer Berlin Heidelberg, 2015, pp. 516–538.
[13] M. Bellare, S. Keelveedhi, and T. Ristenpart, "Dupless: serveraided encryption for deduplicated storage," in Usenix Conference on Security, 2013, pp. 179–194.
[14] T. Jiang, X. Chen, Q. Wu, J. Ma, W. Susilo, and W. Lou, "Secure and efficient cloud data deduplication with randomized tag," IEEE Transactions on Information Forensics and Security, vol. 12, no. 3, pp. 532–543, 2017.
[15] J. Li, X. Chen, M. Li, J. Li, P. P. C. Lee, and W. Lou, "Secure deduplication with efficient and reliable convergent key management," IEEE Transactions on Parallel and Distributed Systems, vol. 25, no. 6, pp. 1615–1625, 2014.
[16] X. Yang, R. Lu, J. Shao, X. Tang, and A. Ghorbani, "Achieving efficient secure deduplication with user-defined access control incloud," IEEE Transactions on Dependable and Secure Computing, 2020.
[17] H. Yuan, X. Chen, J. Li, T. Jiang, J. Wang, and R. Deng, "Secure cloud data deduplication with efficient re-encryption," IEEE Transactions on Services Computing, pp. 1–1, 2019.
[18] Y. Shin, D. Koo, and J. Hur, "A survey of secure data deduplication schemes for cloud storage systems," ACM Computing Surveys, vol. 49, no. 4, pp. 74:1–74:38, 2017.
[19] J. Hur, D. Koo, Y. Shin, and K. Kang, "Secure data deduplication with dynamic ownership management in cloud storage," IEEE Transactions on Knowledge and Data Engineering, vol. 28, no. 11, pp. 3113–3125, 2016.