SAFE AND EFFECTIVE DATA REDUNDANCY IN COLLABORATIVE CLOUD STORAGE

[1]K.Jaya Krishna, [2]Bandi Ashok,
[1]Associate Professor, Department of Master of Computer Applications,
QIS College of Engineering & Technology, Ongole, Andhra Pradesh, India
[2]PG Scholar, Department of Master of Computer Applications,
QIS College of Engineering & Technology, Ongole, Andhra Pradesh, India

Abstract: Data deduplication is an effective method to remove redundant data in cloud storage, which helps reduce users' bandwidth requirements. However, many previous methods that rely on a trusted key server (KS) have limitations such as information leakage, vulnerability to attacks, and high computational overhead. Furthermore, if the KS fails, the entire system ceases to function, creating a single point of failure. In this paper, we introduce a Secure and Efficient Data Deduplication (SED) scheme for a Joint Cloud storage system. SED provides global services through collaboration with various clouds and supports dynamic data updates and sharing without relying on a trusted KS. This scheme also addresses the single-point-of-failure issue prevalent in traditional cloud storage systems. Theoretical analysis demonstrates that SED ensures semantic security within the random oracle model and possesses robust anti-attack capabilities, including resistance to brute-force and collusion attacks. Additionally, SED can effectively remove data redundancies with minimal computational complexity and low communication and storage overhead. These features enhance the usability of SED for clients. Comparative results indicate that ourscheme outperforms existing methods in terms of performance.

Keywords: Dynamic data, data sharing, a single point of failure, cloud storage, and JointCloud

I. INTRODUCTION

Cloud storage offers extensive data storage and service access on a "pay-as-you-go" basis. However, redundant data often wastes and occupies storage resources. Data deduplication effectively detects and eliminates these redundancies, ensuring that only a single copy of the data is uploaded and stored. This technology reduces bandwidth requirements on the client-side and enhances space utilization on the server-side. It is widely used in various cloud computing services to improve user experience and save storage space.

The classic data deduplication scheme, along with its variants, typically involves a key server (KS), a cloud storage provider (CSP), and users, relying on the trusted KS for security. However, these schemes often face single-point-of-failure and "platform lock-in" issues; if the KS fails, the entire cloud storage system ceases to function, preventing data outsourcing protocols from being executed. Recently, the Joint Cloud computing system has been introduced to address these problems effectively. This model features a

network architecture comprising users and multiple CSPs that collaborate to offer various services without the need for a trusted KS. Users can connect to any of these CSPs for computing services, ensuring efficient cross-cloud operations and meeting the demands of global cooperative cloud services through multilateral collaboration. Additionally, Joint Cloud can be implemented in decentralized systems, garnering significant interest from both academia and industry.

Massive data breaches affecting billions of personal records are increasingly common. To ensure data confidentiality in cloud storage systems, outsourced data is typically encrypted. However, detecting and deleting duplicate copies becomes challenging in the ciphertext domain because traditional encryption results in different ciphertexts for the same plaintext when encrypted by different users. To address this, convergent encryption (CE) and its variants have been proposed, wherein data is encrypted using keys derived from the data itself. This method, however, is vulnerable due to several security issues:

1) the tag reveals the hash value of the plaintext, making it susceptible to chosen-plaintext attacks;
2) the ciphertext lacks semantic security;
3) predictable plaintexts are vulnerable to brute-force attacks;
4) users face significant computational burdens to protect their data against malicious attackers.

Previous solutions to address these issues fall into two categories, each with its own limitations. The first type enhances the security of outsourced data by restricting data access and availability within the classic system model. This involves using key management and access control techniques to manage encryption keys and secure data access. However, these schemes often have high computational complexity and communication costs due to the need for sharing numerous keys and storing auxiliary information. The second type designs new system models to improve security and functionality, requiring users to interact with multiple trusted key servers (KSs) for deduplication. This approach increases the risk of attacks due to the frequent interactions between the client-side and server-sides. Additionally, most prior schemes fail to address the single-point-of-failure issue effectively.

In the JointCloud system, users' data can be stored in one or multiple clouds, which renders classic deduplication schemes ineffective. To address this, we have developed a Secure and Efficient Data Deduplication (SED) scheme tailored for the JointCloud model. The primary goals of our SED are security, functionality, and efficiency.

1) Security: SED ensures secure data outsourcing and deduplication without the need for a trusted key server, maintaining data confidentiality, integrity, and protection against attacks.
2) Functionality: SED supports renewable, shareable, and access-controlled data storage across multiple cloud service providers (CSPs). Data owners can update and share their data, while non-

owners are restricted from accessing and manipulating it.

3) Efficiency: SED performs operations such as uploading, deduplicating, updating, and sharing with minimal computational overhead, ensuring efficient use of resources.

II. LITERATURE SURVEY

Keyd: Secure Key -Deduplication With Identity Based Broadcast Encryption

Deduplication, a process that conserves storage space by storing only one instance of identical data, has become increasingly vital with the exponential growth of data stored in the cloud. To maintain data confidentiality, it is commonly encrypted before being outsourced. However, traditional encryption methods generate different ciphertexts for the same plaintext, making deduplication challenging. Convergent encryption addresses this issue by ensuring identical plaintexts encrypt to the same ciphertexts. Nonetheless, managing a large number of convergent keys poses a challenge. Existing deduplication schemes either require key management servers or involve interactions between data owners. In this study, we introduce a novel client-side deduplication protocol called KeyD, which eliminates the need for an independent key management server by leveraging identity-based broadcast encryption (IBBE). Users interact solely with the cloud service provider (CSP) during data upload and download. Security analysis confirms that KeyD ensures data confidentiality and convergent key security while protecting ownership privacy. Detailed performance comparisons demonstrate that our scheme achieves a better balance between storage cost, communication, and computation overhead.

EVA: Efficient Versatile Auditing Scheme for IoT-Based Datamarket in Jointcloud

Cloud storage is key for the Internet of Things (IoT) by allowing massive numbers of devices to connect and share data. However, it can be more than just storage. Researchers have proposed data marketplaces where IoT data is bought and sold like any other good. Security and efficiency are crucial when storing and using this data, especially considering the risk of a single cloud server failure. To address these challenges, this article proposes a new system called a Versatile Auditing Scheme (EVA) that allows for secure, efficient, and dynamic data storage in a joint cloud system, while also supporting data trading using blockchain technology. The authors provide analysis and experiments to show that EVA can efficiently handle large files.

Secure Cloud Data Deduplication with Efficient Re-Encryption

Cloud storage uses data deduplication to save space by storing only unique data chunks. However, when this data is encrypted for security, it becomes difficult to update ownership or access rights. This paper focuses on re-encryption for encrypted deduplication. It highlights a vulnerability in a recent scheme (REED) and proposes a new secure and efficient solution. This new scheme leverages a technique called CAONT and the Bloom filter to achieve efficient re-encryption, reducing the computational burden. Unlike prior approaches, it only re-

encrypts a small portion of the data, improving overall system performance. Security analysis and experiments confirm the effectiveness of this approach.

Achieving Efficient Secure Deduplication with User-Defined Access Control in Cloud

Cloud storage lets users easily store and share data securely. A key technique used is "deduplication," which removes redundant copies of encrypted data to save space and bandwidth. However, existing secure deduplication schemes struggle to balance four key security features: data privacy, data integrity, access control, and resistance to brute-force attacks. This paper proposes a new efficient and secure deduplication scheme that allows user-defined access control. It achieves this by letting the cloud provider authorize data access on the owner's behalf. This approach maximizes deduplication benefits while ensuring user privacy and data integrity. Security analysis and simulations show that this scheme outperforms existing solutions in terms of efficiency, storage savings, and deduplication effectiveness.

An efficient and provably secure authenticated key agreement scheme for mobile edge computing

Mobile cloud and edge computing (MCC and MEC) offer greater convenience for mobile services, but security issues like user verification, anonymity, and intractability remain. Existing security protocols for MCC and MEC have vulnerabilities. This paper focuses on Jia et al.'s recent authentication scheme for MEC and exposes its weaknesses to attacks. It then shows similar vulnerabilities in Li et al.'s scheme, derived from Jia's work. To address these shortcomings, the paper proposes a new secure key agreement protocol based on Jia's scheme. This new protocol offers strong security, low computational cost, and efficient communication compared to existing solutions. Formal security proofs using the AVISPA tool and simulations in NS-3 demonstrate the effectiveness and practicality of the proposed scheme in MEC environments.

III. METHODOLOGY

Input Design is a crucial aspect throughout the software development life cycle, demanding meticulous attention from developers. Its primary goal is to ensure accurate data feeding into the application. Effective input design aims to minimize errors during data input. As per the principles of Software Engineering Concepts, input forms or screens are crafted to incorporate validation controls for input limits, ranges, and other relevant criteria.In this system, input screens are present in nearly all modules. Error messages are created to notify users of any mistakes and guide them in correcting them to prevent invalid entries. Let's delve deeper into this aspect in the module design phase. Input design involves the conversion of user-generated input into a computer-readable format. Its objective is to ensure logical data entry that is devoid of errors. Input design controls errors in the input process. The application has been developed with user-friendliness in mind. Forms are designed to position the cursor appropriately for data entry. Users are also

given the option to select suitable inputs from various alternatives in certain cases. Validation is necessary for all entered data. When a user inputs erroneous data, an error message appears, allowing the user to proceed to subsequent pages only after completing all entries on the current page.

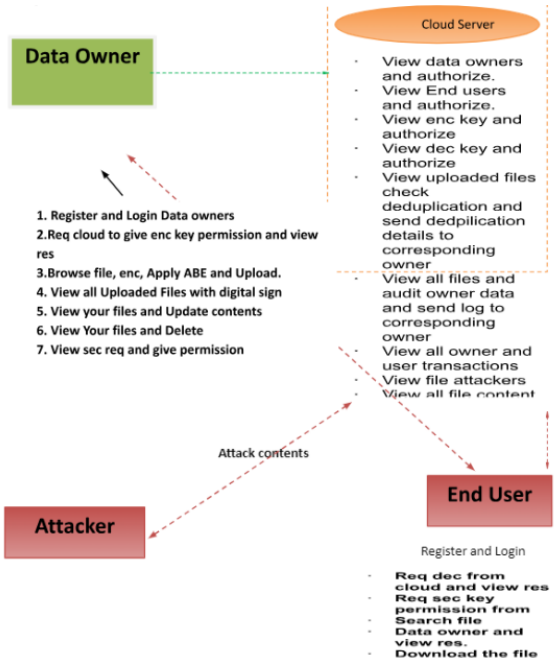


Fig 1 : System Architecture

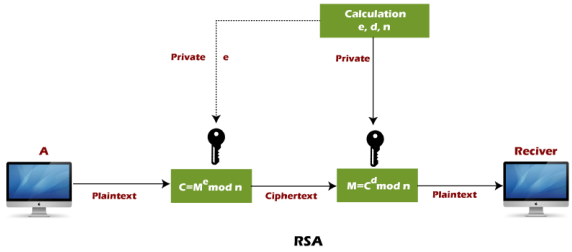
The computer's output is primarily needed to facilitate efficient communication within the company, especially between the project leader, team members, administrators, and clients. The output of the Virtual Private Network (VPN) system enables the project leader to manage clients by creating new client profiles, assigning projects, maintaining project records, and granting folder-level access based on the allocated projects. Upon project completion, new projects may be assigned to clients. User authentication procedures are initiated at the outset, allowing the administrator to create new users or users to self-register, with project assignment and validation tasks reserved for the administrator. Upon initial execution, the application starts running, requiring the server and Internet Explorer as the browser. The project operates within the local area network, with the server serving as the administrator and connected systems acting as clients. The developed system prioritizes user-friendliness, ensuring easy comprehension even for first-time users.

Convergent encryption stands as a primary method for ensuring data security in deduplication processes, safeguarding outsourced data from untrusted or malicious Cloud Service Providers (CSPs). Bellare et al. introduced a concept termed Message-Locked Encryption (MLE), further expanded upon in subsequent works. However, MLE-based approaches faced potential risks due to key derivation from the files themselves. Abadi et al. proposed both fully randomized and deterministic encryption schemes for bounded message distributions, leveraging efficiently computable non-degenerate bilinear maps. Li et al. introduced a scheme focusing on reliable key management in deduplication, followed by Jiang et al. presenting a secure deduplication scheme utilizing randomized tags, albeit without addressing data updating requirements. Hur et al. later explored dynamic ownership management for secure deduplication, requiring clients to store keys along a binary tree path for deduplication. Li et al. proposed secure deduplication with key management based on secret sharing schemes. Shin et al. designed a decentralized server-aided encryption approach for deduplication, albeit requiring multiple interactions between users and Key Servers (KS), potentially exposing

communication to attackers. Additionally, Miao et al. proposed secure deduplication for multi-server scenarios. Xia et al. introduced a Fast and Efficient Content-Defined Chunking method for achieving fine-grained data deduplication. Zhao et al. presented a deduplication scheme utilizing a Docker registry architecture, optimizing space usage by deduplicating layers and reducing layer restoration overhead. Discussion on common attacks against recent deduplication schemes was addressed. Furthermore, emerging technologies like blockchain are increasingly explored for deduplication across various applications, attracting recent attention from researchers. In the Data Owner module, individuals upload their data to the cloud server, encrypting both the file and index names for security before storing them. The data owner, acting as the encryptor, has the capability to selectively delete specific files and can also monitor transactions related to the files uploaded to the cloud. In the Data User module, users log in using their credentials and initiate search requests within the cloud, searching for files based on index keywords, viewing search ratios and top documents, and subsequently downloading the files. The Cloud Server module manages cloud infrastructure to offer data storage services. Data owners encrypt their files for sharing with remote users, who then download and decrypt the files from the cloud. The cloud server facilitates authorization for both data owners and users, processing search requests and offering personalized and interest-based search models, while also providing visibility into potential file attackers.

IV. ALGORITHMS

RSA Algorithm: RSA is the most common public-key algorithm, named after its inventors Rivest, Shamir, and Adelman (RSA).



RSA

RSA algorithm uses the following procedure to generate public and private keys:

Select two large prime numbers, p and q.
Multiply these numbers to find n = p x q, where n is called the modulus for encryption and decryption.
Choose a number e less than n, such that n is relatively prime to (p - 1) x (q -1). It means that e and (p - 1) x (q - 1) have no common factor except 1. Choose "e" such that 1<e < φ (n), e is prime to φ (n),
gcd (e,d(n)) =1
If n = p x q, then the public key is <e, n>. A plaintext message m is encrypted using public key <e, n>. To find ciphertext from the plain text following formula is used to get ciphertext C.
C = me mod n
Here, m must be less than n. A larger message (>n) is treated as a concatenation of messages, each of which is encrypted separately.
To determine the private key, we use the following formula to calculate the d such that:
De mod {(p - 1) x (q - 1)} = 1
Or
De mod φ (n) = 1
The private key is <d, n>. A ciphertext message c is decrypted using private key <d, n>. To calculate plain text m from the

ciphertext c following formula is used to get plain text m.
m = cd mod n

SHA Algorithm:
SHA stands for secure hashing algorithm. SHA is a modified version of MD5 and used for hashing data and certificates
SHA algorithm uses the following procedure

1. Append padding bits
The first step in our hashing process is to add bits to our original message to make it the same length as the standard length needed for the hash function. To accomplish so, we begin by adding a few details to the message we already have. The amount of bits we add is determined so that the message's length is precisely 64 bits less than a multiple of 512 after these bits are added.
This can be expressed mathematically in the following way:

n x 512 = M + P + 64
M is the original message's length.
P stands for padded bits.

2. Append length bits
Now that we've added our padding bits to the original message, we can go ahead and add our length bits, which are equal to 64 bits, to make the whole message an exact multiple of 512.

We know we need to add 64 extra bits, so we'll compute them by multiplying the modulo of the original message (the one without the padding) by 232. We add those lengths to the padded bits in the message and get the complete message block, which must be a multiple of 512.

3. Initialize the buffers
We now have our message block, on which we will begin our calculations in order to determine the final hash. Before we get started, I want to point out that we'll need certain default settings to get started with the steps we'll be taking.

a = 0x6a09e667
b = 0xbb67ae85
c = 0x3c6ef372
d = 0xa54ff53a
e = 0x510e527f
f = 0x9b05688c
g = 0x1f83d9ab
h = 0x5be0cd19

Keep these principles in the back of your mind for now; all will fit together in the following phase. There are a further 64 variables to remember, which will operate as keys and are symbolized by the letter 'k.'

Let's go on to the portion where we calculate the hash using these data.

4. Compression Function
As a result, here is where the majority of the hashing algorithm is found. The whole message block, which is 'n x 512' bits long, is broken into 'n' chunks of 512 bits, each of which is then put through 64 rounds of operations, with the result being provided as input for the next round of operations.
The 64 rounds of operation conducted on a 512-bit message are plainly visible in the figure above. We can see that we send in two inputs: W(i) and K(i). During the first 16 rounds, we further break down the 512-bit message into 16 pieces, each consisting of 32 bits. Indeed, we must compute the value for W(i) at each step.

$W(i) = W^{i-16} + \sigma^{0} + W^{i-7} + \sigma^{1}$
where,
σ^{0} = (W^{i-15} $ROTR^{7}(x)$) XOR (W^{i-15} $ROTR^{18}(x)$) XOR (W^{i-15} $SHR^{3}(x)$)

σ^{1} = (W^{i-2} $ROTR^{17}(x)$) XOR (W^{i-2} $ROTR^{19}(x)$) XOR (W^{i-2} $SHR^{10}(x)$)

$ROTR^n(x)$ = Circular right rotation of 'x' by 'n' bits

$SHR^n(x)$ = Circular right shift of 'x' by 'n' bits

In this paper, we propose a secure and efficient data deduplication scheme (SED) for the Joint Cloud storage system, eliminating the need for a trusted Key Server (KS). Our SED incorporates sub-algorithms inspired by a fully randomized tag generation algorithm, aiding in duplicate detection and safeguarding outsourced data from collusion attacks. Unlike previous schemes, SED ensures that both ciphertext and tags meet semantic security standards, preventing adversaries from extracting any useful information. Notably, SED is the first scheme to securely support data updates and sharing. Our encryption algorithm facilitates deduplication, updating, and sharing of data. SED uniquely allows data owners to share their outsourced data with authorized users, utilizing a master encryption key generated collaboratively by participating Cloud Service Providers (CSPs), ensuring secure and flexible key generation. Data access control based on authentication further supports secure data updates and sharing. SED combines intra-deduplication and inter-deduplication techniques to enhance deduplication efficiency within the Joint Cloud system. Theoretical analyses demonstrate that SED offers superior performance in terms of data confidentiality, integrity, and resistance to strong attacks and collusion. Experimental implementation and simulation using Crypto++, GNU, and PBC libraries on Ubuntu show that SED is both efficient and computationally cost-effective.

V. RESULTS

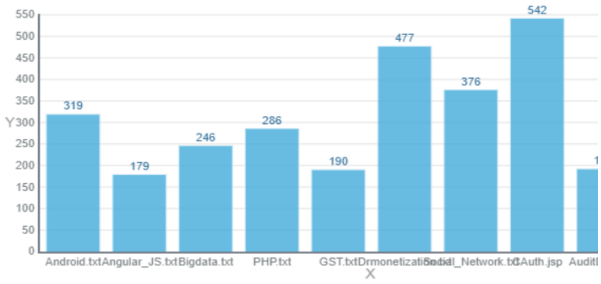


Fig 1:File Upload/Store time Delay Results

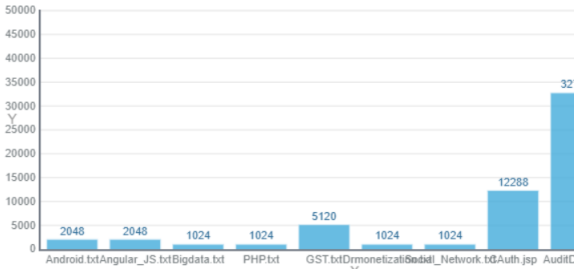


Fig 2:File Upload Throughput Results

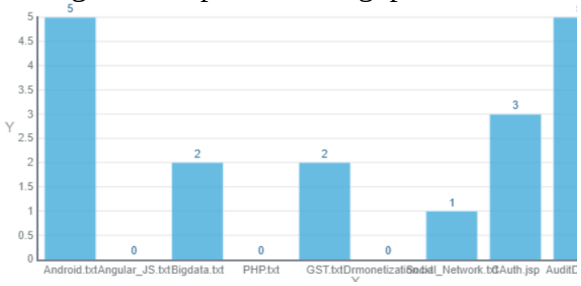


Fig 3:File Rank Results

VI. CONCLUSION

In this paper, we present a secure and efficient data deduplication scheme (SED) that operates without relying on a trusted Key Server (KS). The proposed SED minimizes client-side communication and computation overhead, enhancing efficiency based on the Computational Diffie-Hellman (CDH) problem in the JointCloud storage system. Its streamlined encryption and tag generation algorithms ensure semantic security and tag consistency, encompassing both security and validity. SED also improves scalability and addresses the single-point-of-failure issue common in traditional cloud storage systems. It offers robust protection against typical attacks, such as brute-force and collusion

between malicious Cloud Service Providers (CSPs) and unauthorized users. Additionally, SED supports dynamic data operations, including deletion, modification, and sharing, enhancing functionality and usability. Notably, SED is the first scheme to allow data owners to securely share their outsourced data with authorized users. Theoretical and experimental analyses confirm that SED is secure and exhibits low computational, communication, and storage complexity. Compared to previous schemes, SED is more secure, efficient, and functional.

VII. REFERENCES

[1] P. Christen, "A survey of indexing techniques for scalable record linkage and deduplication," IEEE Transactions on Knowledge and Data 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 performance in 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 for data deduplication based storage systems," IEEE Transactions on Parallel and Distributed Systems, vol. 31, no. 9, pp. 2017–2031, 2020.

[4] J. Li, J. Li, D. Xie, and Z. Cai, "Secure auditing and deduplicating data 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 with identity-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 allocation and 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, "Toward encrypted cloud media center with secure deduplication," IEEE Transactions on Multimedia, vol. 19, no. 2, pp. 251–265, 2017.

[8] H. Wang, P. Shi, and Y. Zhang, "Jointcloud: A cross-cloud cooperation architecture for integrated internet service customization," in 2017 IEEE 37th International Conference on Distributed Computing Systems, 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-based datamarket in jointcloud," IEEE Internet of Things Journal, vol. 7, no. 2, pp. 882–892, 2020.

Authors

[1] Mr. K. Jaya Krishna, currently working as an Associate Professor in the Department of Master of Computer Applications, QIS College of Engineering and Technology, Ongole, Andhra Pradesh. He did his MCA from Anna University, Chennai, M.Tech (CSE) from JNTUK, Kakinada. He published more than 10 research papers in reputed peer reviewed Scopus indexed

journals. He also attended and presented research papers in different national and international journals and the proceedings were indexed IEEE. His area of interest is Machine Learning, Artificial intelligence, Cloud Computing and Programming Languages.

[2] Mr. Bandi Ashok, currently pursuing Master of Computer Applications at QIS College of engineering and Technology (Autonomous), Ongole, Andhra Pradesh. He Completed B.Sc. in Computer Science from NNR & CL Degree College, Ongole, Andhra Pradesh. His areas of interests are Cloud Computing & Machine learning.