Encrypted Cloud Data Deduplication Technique for Secure Data Storage Optimization Over Cloud

Suhas A. Lakade **Yuvraj R. Gurav** **Harsh Lohia**

Department of Computer Science and Engineering
Sri Satya Sai University of Technology and Medical Science
Sehore, Madhya Pradesh

ABSTRACT

Data duplication is a significant concern in cloud-fog storage integrated environments, leading to wasteful storage usage. Traditional techniques, like backup and archive systems designed for static data, are inadequate due to the dynamic nature of cloud and integrated cloud settings. To address this issue, data deduplication techniques are employed to efficiently eliminate redundant data in cloud storage systems. Implementing data deduplication (DD) on encrypted data presents a substantial challenge in secure integrated cloud-fog storage and computing environments. This paper introduces a novel approach utilizing Convergent and Modified Elliptic Curve Cryptography (MECC) algorithms in cloud and fog environments to create secure deduplication systems. The method primarily focuses on two critical objectives: minimizing data redundancy and developing robust encryption methods to ensure data security. It is well-suited for tasks such as uploading new files to fog or cloud storage. The process begins by encrypting the file using Convergent Encryption (CE) and then further encrypting it with the Modified Elliptic Curve Cryptography (MECC) algorithm. This approach efficiently identifies redundancy at the block level, resulting in more effective data redundancy reduction. Test results demonstrate that the proposed method surpasses several state-of-the-art methods in terms of computational efficiency and security.

KEYWORDS *: Convergent encryption (CE), Modified elliptic curve cryptography (MECC), Edge computing, Integrated cloud and fog networks, Hash tree, Secure hash algorithm (SHA).*

INTRODUCTION

The exponential growth of data from various sources and the widespread adoption of the Internet of Things across applications have led to a massive increase in data volume, reaching from petabytes to yottabytes. To handle this data, cloud computing and fog networks have become essential for data processing and storage.

Cloud computing (CC) offers a network-centric environment, providing users access to a collective pool of programmable resources, including servers, software, storage, and services that can be easily accessed over the internet. Data processing in cloud servers is done remotely via internet connectivity. In contrast, fog computing provides local infrastructure for processing applications locally before connecting to the cloud. This reduces latency compared to processing applications directly in the cloud. To ensure data security, communication protocols, and resources are crucial for accessing information stored in both cloud and fog environments.

Smart applications, often relying on sensors and actuators, store data in the cloud while utilizing edge computing facilities, referred to as "fog," for low-latency data processing. The Internet of Things aims to create cyber-physical systems, extending user mobility with edge computing for on-the-go data processing.

Traditionally, sending data to the cloud was the dominant trend, but it is evolving to include fog, edge, and cloudlet solutions to address the demands of delay-sensitive and context-aware services. This transition from centralized cloud computing to distributed edge cloud computing enhances transparency.

Fog computing offers an attractive solution for

distributed edge cloud computing, particularly for low-latency, mobility, and geodistributed services. Several schemes focus on offloading tasks from the cloud to reduce latency, improve energy efficiency, and enhance reliability.

Effective resource allocation methods in both cloud and fog environments further boost performance and application processing reliability. Edge computing, which brings cloud capabilities closer to mobile devices, helps address various challenges in the Internet of Things, such as latency, limited bandwidth, energy efficiency, and data security.

Data deduplication (DD) is a widely used technique for removing data redundancy. It involves chunking, hashing, and comparing hashes to identify redundant data. Implementing encryption and deduplication together is crucial for achieving optimized and secure storage. DD can be applied in various scenarios, such as archiving, backup systems, databases, and network storage.

This paper proposes a secure data deduplication system using convergent and MECC algorithms in the integrated cloud-fog-edge environment. While convergent encryption is suitable for deduplication with encryption in cloud storage, it may be vulnerable to dictionary attacks. To address this, the paper suggests using the modified elliptic curve cryptography (MECC) technique to encrypt the deduplicated data. This combined approach ensures efficient deduplication and secure encryption with lower computational overhead compared to existing methods.

The significant contributions of this paper can be summarized as follows:

- A novel approach for establishing a secure data deduplication (DD) system is introduced, utilizing Convergent and Modified Elliptic Curve Cryptography (MECC) algorithms, as published by P. G. et al. in the Journal of Cloud Computing: Advances, Systems, and Applications in 2020. This method is designed to operate within the cloud and fog/edge environments.
- The proposed technique's performance is assessed, focusing on its computational efficiency and security level.
- We have confirmed the capability of the proposed deduplication method to identify data redundancy at the block level. This results in a more efficient reduction of data redundancy and a consequent reduction in cloud storage space usage.

The manuscript's outline is organized as follows: Section 2 examines related research, Section 3 outlines the proposed methodology, Section 4 delves into the initial results, and Section 5 encompasses the conclusion and outlines potential avenues for future research.

RELATED WORKS

This The secure deduplication system eliminates duplicate data copies while ensuring data security. Convergent Encryption (CE) is employed to encrypt and decrypt data at the file level using a key derived from the file's content [20]. Users retain the encryption key and outsource the ciphertext (CT) to the Cloud Server (CS) to conserve storage space. Consistent privacy is maintained by updating the CT in the central cloud and user-level public keys without revealing the private keys [21].

Kwon et al. introduced a secure deduplication framework with user revocations, consisting of three phases: upload, revocation, and download. This framework utilizes a privilege-centric re-encryption method over convergent encryption.

Li et al. recommended "Dekey," which securely distributes convergent key shares across multiple servers. Dekey supports both block-level and file-level deduplication, improving storage efficiency. Security analysis confirmed Dekey's security and minimal overhead.

Kwon et al. proposed a server-side deduplication framework for encrypted data, allowing the Cloud Server to control access to outsourced data as ownership changes dynamically. It prevents data leakage to revoked users and ensures data integrity.

Yuan et al. developed a randomized framework (R-MLE2) with static and dynamic schemes, achieving high-level performance for data equality tests. Han et al. introduced a multi-bit secret channel in cloud storage, simplifying data upload and enhancing security. Tawalbeh reconsidered security and privacy for cloud and fog environments using a health care system case study, improving performance and trust among users.

Zhang introduced indexing for high-performance deduplication of data, providing quick responses to fingerprint queries. Hou suggested checking the truthfulness of cloud data when the remote server stores only a single copy of the same file from different users. Deduplication offers significant space and cost savings for storage providers.

Enhanced Secure Threshold Data Deduplication Pattern helps maintain end-to-end encryption, and Proxy re-encryption (PRE) is a flexible admission control tool with efficiency in computational cost and ciphertext size.

A confidentiality-preserving deduplication technique for public cloud services is discussed, supporting public reliability and auditing in the cloud storage system. A chaotic fuzzy transformation method is proposed for fog systems, improving user data privacy, confidentiality, and resource savings.

A comprehensive study on security issues related to cloud data and their solutions is described, with a focus on access control models for the cloud computing environment. A framework is introduced to statically and dynamically mine structures from malware, accurately classifying malware with low false alarms. Public-key-based schemes are presented to address security vulnerabilities in symmetric-key-based μTESLA-like schemes, although their signature verification is time-consuming.

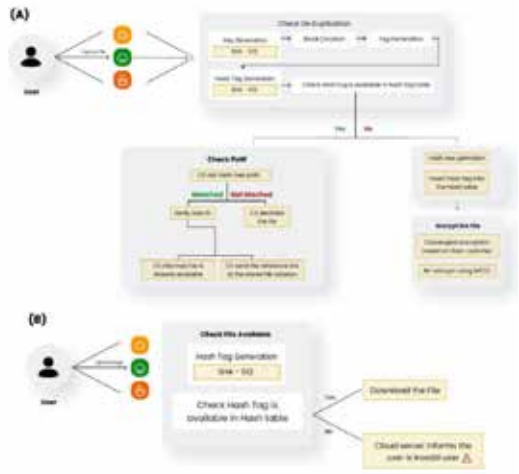

Fig. 1 a Uploading the file into the cloud server using the proposed system. b Downloading the file from the cloud server using the proposed system

PROPOSED SECURE DEDUPLICATION APPROACH

To The Cloud Service Provider offers various resources to users in the form of services, such as extensive storage capacity. Handling the continuously growing volumes of data within the cloud presents a significant challenge. Data Deduplication (DD) is an effective technique for enhancing data management scalability in Cloud Computing (CC). However, the primary concern in Data Deduplication is security. To address this issue, this paper introduces a secure data deduplication system that employs convergent and Modified Elliptic Curve Cryptography (MECC) algorithms in an integrated cloud-fog environment.

The proposed methodology is evaluated in four scenarios: a) when a new user attempts to upload a new file, b) when the same user tries to upload a duplicate file, c) when different users seek to upload identical files to the cloud server, and d) when users aim to download the file. The detailed explanation of the proposed methodology can be found in the block diagram illustrated in Figure 1a and 1b.

When a new user tries to upload a new file

To begin, when a new user selects a file for upload to the Cloud Server (CS), the CS generates a unique hash code (HC) key for the file using the SHA-512 algorithm. This input file is then modified by adding padding and a fixed 128-bit length field. The resulting message is divided into blocks, and a 64-bit word is computed for each message block using eight constants derived from the square roots of the first eight prime numbers. Subsequently, a 512-bit buffer is updated in the SHA-512 operation. The process of SHA-512 can be better understood by referring to Figure 2.

The original input file is then segmented into blocks, and tag values are assigned to each block. A hash code (HC) is generated for each tag value of a specific block using the same SHA-512 algorithm. The Cloud Server (CS) checks whether the hashtag exists in the Hash Table (HT). If it is not present, the hash tree is created for Proofs of Ownership (PoW) based on the hashtag value.

Next, the file is encrypted using a convergent encryption (CE) method. This CE method takes the HC

key of the file as input. Subsequently, the convergent-based encrypted file undergoes an additional layer of encryption using the MECC algorithm.

This encryption process is implemented to ensure the secure uploading of data to the CS. The details of the CE and MECC-based encryption of the specific file are elaborated as follows.

Convergent encryption (CE)

Within data deduplication, Convergent Encryption (CE) significantly enhances data confidentiality. The Convergence Key (CK) is represented by the hash code (HC) value generated from the file. Utilizing this CK, all data copy blocks are encrypted. To identify duplicate files within the Cloud Service Provider (CSP), a tag is created for each data block. If two data files are identical, they will receive the same tags.

Before storing the data file in the CSP, its tag is sent to the CSP to identify duplicate data files.

Ultimately, this encrypted data block, along with its tag, is stored within the CSP. The phases involved in Convergent Encryption (CE) are explained as in Figure 2.

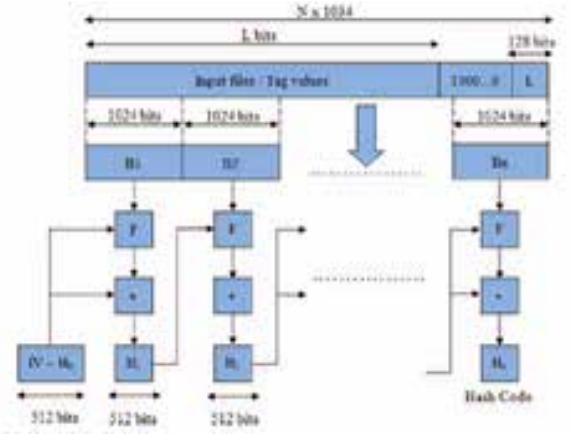

Fig. 2 Structure of the SHA-512 algorithm [35]

Convergent encryption

In the encryption process, the original file (f) and κh are provided as input to the encryption algorithm, and the encryption function Ey is applied. As a result, the encryption algorithm produces ciphertext (Ct) as its output.

$$Ct = Ey(f, kh) \quad (1)$$

Convergent decryption

During decryption, the encrypted file f or Ct is inputted to the decryption algorithm. Finally, this decryption algorithm outputs f and Ct.

$$f = Dy(Ct, kh) \quad (2)$$

The CE algorithm demonstrates superior performance in comparison to other existing methods. However, it does have a drawback in terms of security, as it remains vulnerable to dictionary attacks. To address this vulnerability, the proposed approach introduces an additional layer of encryption for the previously convergent-encrypted file, employing the Modified ECC algorithm, which is elaborated as follows.

The Elliptic Curve Cryptography (ECC) algorithm is founded on a curve featuring specific base points and a prime number function, which serves as an upper limit. ECC is a cryptographic algorithm employed in public-key cryptography. The mathematical model of ECC, with "g" and "e" as integers, is provided below.

$$w^2 = v^3 + gv + e,\ 4g^3 + 27e^2 \neq 0 \quad (3)$$

In cryptographic procedures, the effectiveness of encryption primarily relies on the method employed for key generation. In the proposed system, three types of keys need to be generated. The initial step involves generating the public key (αk) from the server and encrypting it. Subsequently, a private key (βk) is generated on the server-side for message decryption. The final step is to generate a secret key (ok) using αk, βk, and a point on the curve (ρc). The generation of αk follows the equation provided below.

$$\alpha_k = \beta_k * p_c \quad (4)$$

The eq. (5) elucidates α_k generation,

$$O_k = \alpha_{k*\beta_k*P_c} \quad (5)$$

After O_k generation, the file is encrypted. This encrypted file contains two CTs, and mathematically,they are depicted as,

$$C_1 = ((K:=1, 2, \ldots,(n-1)*p_c) + O_k \quad (6)$$

$$C_2 = (f + ((K:=1, 2, \ldots, (n-1))*\alpha_k)) + O_k \quad (7)$$

Here, C1 and C2 represents the two CTs, K is the random number generated in (1, ..., (n − 1)) interval. During encryption, O_k is added to the CTs. During decryption,

O_k is subtracted with the two CTs, and the original file f is given by,

$$f = (((C_2 - \beta_k) * C_1) - o_k) \quad (8)$$

When the same user attempts to upload the same file once more, the Cloud Server (CS) computes the hash value using the Convergence Key (CK) with the SHA-512 algorithm. For each input file, the binary representation of the file is divided into fixed-sized blocks. The size of these data blocks determines the granularity of deduplication. Smaller data block sizes lead to higher deduplication levels, but they can also introduce complexity in metadata management. The proposed approach considered file block sizes of 5MB, 10MB, 15MB, 20MB, and 25MB. Subsequently, a tag key is generated for each of the segmented blocks, and the hash value is calculated for all tag keys using the same SHA-512 algorithm.

In the uploading phase, the CS examines the hashtag (HT) for a specific input file. If the hashtag value of the input file is found in the HT, the CS requests the hash tree path from the users. If a user provides the correct path, the CS verifies the user's ID. If the ID matches, the CS refrains from storing the file again. Generally, the hash tree path adheres to the following format:

$$P(H_t) = \{ RLL, RLR, etc. \} \quad (9)$$

Here, $P(H_t)$ signifies the hash tree path, with RLL representing "Root, Left, Left," and RLR indicating "Root, Left, Right." The leaf node is excluded from the hashed tree path. Mathematically, the scenario where the same user attempts to upload the same file is denoted as follows:

$$S_u \xrightarrow{Uploads} S_f \; (CS \xrightarrow{Informs} A) \quad (10)$$

When different users attempt to upload identical files to the Cloud Server (CS), the file is divided into multiple blocks, and a tag is generated to check for duplicate data copies within the CSP. Subsequently, each tag is transformed into a Hash Code (HC), referred to as a hashtag value. The CS examines the Hash Table (HT) for the input file based on this hashtag value. If the hashtag value is present in the HT, the CS requests the hash tree path of the input file. If a user provides the correct path, the CS verifies the user's ID. If the ID is distinct, the CS shares the reference link to the specific stored file's location with the user. The scenario of different users attempting to upload the same file to the CS can be represented as follows:

$$D_u \xrightarrow{Uploads} S_f \; (CS \xrightarrow{asks} P(H_t)) \quad (11)$$

$$P(H_t) \quad matched \rightarrow (CS \xrightarrow{send}) R_l(f) \quad (12)$$

$$P(H_t) \quad Not\ Matched \rightarrow (CS \xrightarrow{Informed} I_u) \quad (13)$$

Where D_u denotes the different users. R_l is the reference link and I_u denotes an invalid user.

When User tries to download the file

In this scenario, the user transmits the tag value of the designated file. Subsequently, the CS calculates the hash value using the SHA-512 algorithm. The CS proceeds to verify the existence of the hashtag value within the Hash Table (HT). If the value is found, the CS grants the user permission to download the file; otherwise, the CS categorizes them as an unauthorized user. This situation can be expressed mathematically as follows:

$$H(T) \quad matched \rightarrow D_u \; (\downarrow) \quad (14)$$

$$H(T) \quad Not\ Matched \rightarrow I_u \quad (15)$$

Where H(T) denotes the hashtag value. Pseudocode for the proposed secure deduplication system is evinced below,

Algorithm 1 Uploading file into the cloud server

Input: Original file

Output: Upload the file into the CS

begin

initialize keyk, tagT, hashtagH(T),

blocks(B1, B2,..., B,,), and hash tree H

for-number of files do

{

generatek using SHA-512

dividef into blocks (B_1, B_2,..., B_1)

generate tag T

generate H(T) using SHA-512 if (H(T) == Hash_ table) then

```
    check PoW
    else
        generate Ht and insert H(T) into the-hash table
        and encrypt the file f
        store the file in Cloud Server (CS)
    end if
    }
end for
end
```

Algorithm 2 Downloading file from the cloud server

```
Input: Tag value
Output: Download the original file
begin
initialize tag T, hashtag H(T), original-
file f
for all tags do
{
    generateH(T) using SHA-512
    if(H(T) Hashtable) then
    Cloud Server allows the user to download-the
     file f(↓)
    else
    Cloud Server informs as invalid user
    end if
}
end for
end
```

RESULT AND DISCUSSION

The implemented deduplication method has been executed within the JAVA programming environment, utilizing the system configuration outlined below. The performance assessment of the system focuses on various file sizes ranging from 5 MB to 25 MB, with an increment of 5 MB after each iteration.

In this section, we conduct a performance analysis of the proposed system. Initially, we compare the performance exhibited by the proposed MECC security algorithm to that of existing security algorithms, such as Diffie-Hellman (DH), ECC, and Rivest Shamir Adelman (RSA). This comparison encompasses encryption time (Et), decryption time (Dt), key generation time, and security analysis.

Performance analysis of proposed encryption technique

Encryption time

E_t represents the duration taken by an encryption algorithm to produce encrypted data from the provided input data. Encryption time is determined by calculating the time difference between the end of the encryption process and the start of the encryption process. This is assessed as follows:

Table 1 Performance Comparison of Proposed MECC and Existing Techniques in terms of Encryption Time

File Size in MB	Encryption Time in Seconds				
	Proposed MECC	DH	Existing ECC	Existing RSA	
5	6.21		10.22	14.47	12.44
10	10.04		19.64	22.56	21.76
15	15.54		28.32	28.00	30.35
20	21.00		36.33	36.65	35.61
25	27.55		46.29	44.32	47.28

Table 2 Performance Comparison of Proposed MECC and Existing Techniques in terms of Decryption Time

File Size in MB	Decryption Time in Seconds			
	Proposed MECC	DH	Existing ECC	Existing RSA
5	5.28	12.21	13.5	11.51
10	10.22	18.11	22.66	20.53
15	16.74	26.43	29.43	28.54
20	20.36	34.56	38.64	36.84
25	25.44	45.44	45.81	48.43

$$Et = Ee - Es \quad (16)$$

Where Et is the encryption time, Ee is the encryptionending time and Es is the encryption starting time.

Decryption time

D_tis defined as the difference between the decryption ending time and decryption starting time. It is evaluated as,

$$D_t = D_e - D_s \quad (17)$$

where D_t is the decryption time, D_e is the decryption ending time and D_s is the decryption starting time.

Security

Security is highly essential for cloud storage. The security level is computed by dividing the hacked data with the number of the original text. The security level of the system is expressed as,

$$S = H_d/O_d \quad (18)$$

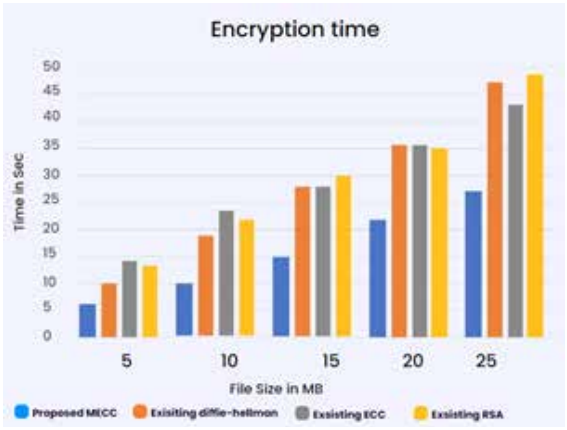

Fig. 3 (a) Performance comparison of proposed MECC with existing techniques in terms of encryption time

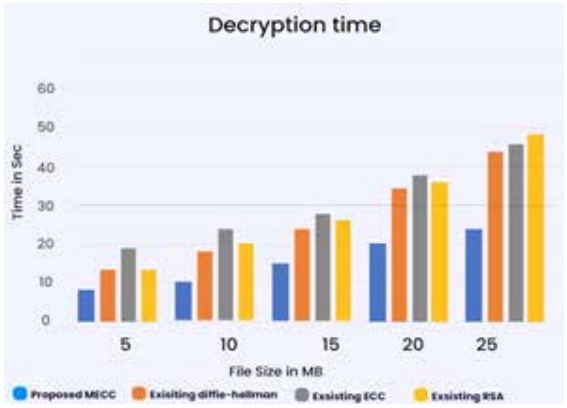

Fig. 3 (b) Performance comparison of proposed MECC with existing techniques in terms of encryption time

Table 3 Performance Comparison of Proposed MECC and Existing Techniques in terms of Key Generation Time and Security Level

SI. No.	Encryption Algorithm	Key Generation Time (ms)	Security (%)
1	Proposed MECC	425.21	96
2	ECC	612.32	90
3	RSA	765.54	87.5
4	DH	856.33	85

The time taken for encryption, denoted as Et, is measured as the duration required for an encryption algorithm to transform input data into encrypted data. Et is calculated as the difference between the ending time and starting time of the encryption process. It is computed as follows:

E_t = Encryption Ending Time - Encryption Starting Time

Tables 1 and 2 provide a performance comparison of the proposed MECC algorithm with the existing Diffie-Hellman (DH), Rivest Shamir Adelman (RSA), and Elliptic Curve Cryptography (ECC) techniques in terms of Et and Dt (encryption and decryption times). The Et and Dt values are presented in seconds (s). For instance, the proposed MECC takes 6 seconds to encrypt a 5MB file, while the existing DH, ECC, and RSA take 10, 14, and 12 seconds, respectively, for encrypting the same 5MB file.

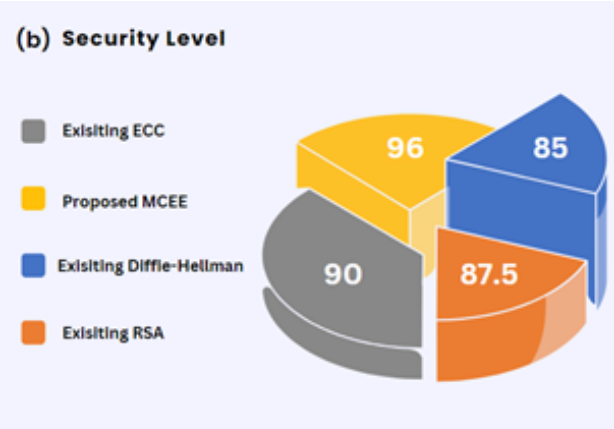

Fig. 4 a Performance comparison of proposed MECC with existing techniques in terms of key generation time. b Performance comparison of proposed MECC with existing techniques in terms of security level

The trend continues for other file sizes (10MB to 25MB) where the proposed method consistently exhibits shorter encryption and decryption times compared to the alternatives. This indicates that the MECC algorithm outperforms the other methods in terms of Et and D_t.

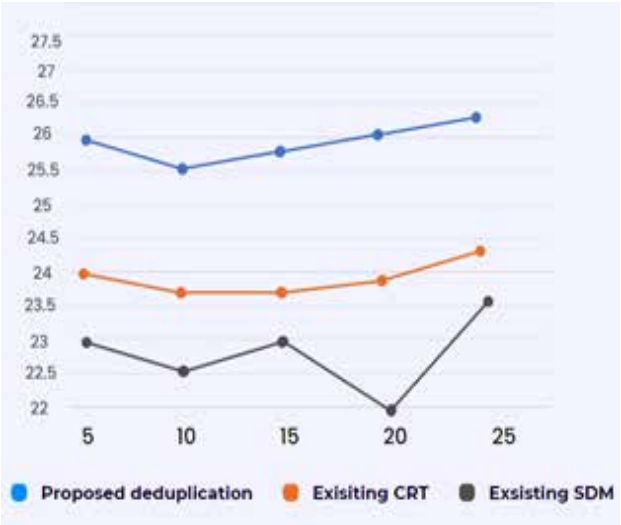

Fig. 5 Performance analysis of the proposed deduplication scheme with existing techniques in terms of deduplication rate

Table 3 compares the key generation time and security level between the proposed MECC technique and the existing methods. The proposed MECC takes 425 milliseconds (ms) to generate a key, whereas the existing ECC, RSA, and DH methods take 612 ms, 765 ms, and 856 ms, respectively. The MECC algorithm demonstrates faster key generation compared to the other techniques. In terms of security, the proposed MECC achieves the highest security value at 96%, while the existing ECC, RSA, and DH methods achieve security levels of 90%, 87.5%, and 85%, respectively. Thus, the MECC algorithm offers superior performance in both key generation and security.

Performance analysis of the proposed deduplication technique involves comparing it to other methods such as Chinese Remainder Theorem (CRT)-based secret sharing and Smart Deduplication for Mobile (SDM) in terms of deduplication rate, as shown in Figure 5. The proposed deduplication method consistently outperforms CRT and SDM across various file sizes, demonstrating a higher deduplication rate.

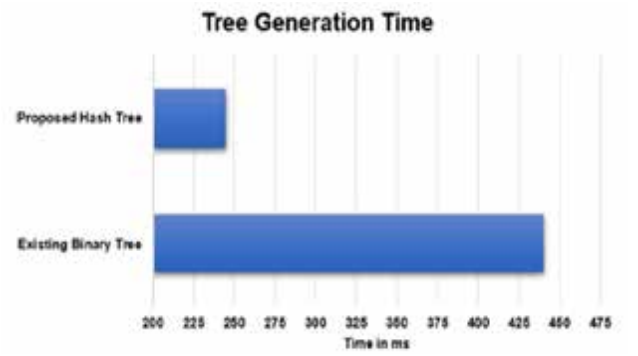

Fig. 6 Performance comparison of the proposed hash tree with existing binary tree

Furthermore, Figure 6 illustrates the performance comparison between the proposed hash tree used in deduplication and the existing binary tree in terms of tree generation time. The proposed hash tree exhibits quicker tree generation, taking 245 ms, compared to the existing binary tree, which takes 440 ms. This indicates that the proposed hash tree approach outperforms the binary tree generation methodology.

Overall, the proposed MECC algorithm and deduplication technique offer superior performance compared to existing methods, as evidenced by their faster encryption, decryption, key generation, higher security, and better deduplication rates.

CONCLUSIONS

Deduplication stands as the most prominent data compression technique. While various existing methods have introduced different deduplication approaches, they often lacked in terms of security. In response, this paper puts forth a secure deduplication system that leverages convergent and MECC algorithms within the cloud-fog environment. The proposed method is assessed in four scenarios: a) when new users attempt to upload a new file, b) when the same user seeks to upload an identical file, c) when distinct users aim to upload the same file, and d) when different users intend to download a file.

Performance evaluation was carried out by considering different file sizes, ranging from 5 MB to 25 MB, with an incremental increase of 5 MB in each iteration. The results of the performance analysis substantiate that the proposed system attains a remarkable 96% security

level, surpassing many other existing encryption methods.

These findings underscore the high degree of security and efficiency offered by the proposed system for data deduplication within an integrated cloud environment. Looking ahead, this model has the potential for extension to various Internet of Things (IoT) applications that rely on dynamic resource management at the edge. Furthermore, it can contribute to the development of cyber-physical systems, accommodating diverse use cases with varying data payloads and formats. The proposed technique holds promise for enhancing security, optimizing computational efficiency, and streamlining storage management within integrated environments, including IoT and cyber-physical systems.

REFERENCES

1. Lohstroh M, Kim H, Eidson JC et al (2019) On enabling Technologies for the Internet of important things. IEEE Access 7:27244–27256. https://doi.org/10.1109/ACCESS.2019.2901509
2. Abbas N, Zhang Y, Taherkordi A, Skeie T (2018) Mobile edge computing: a survey. IEEE Internet Things J 5:450–465. https://doi.org/10.1109/JIOT.2017.2750180
3. Ren J, Zhang D, He S et al (2019) A survey on end-edge-cloud orchestrated network computing paradigms: transparent computing, mobile edge computing, fog computing, and cloudlet. ACM Comput Surv:52. https://doi.org/10.1145/3362031
4. Zhang P, Liu JK, Richard Yu F et al (2018) A survey on access control in fog computing. IEEE Commun Mag 56:144–149. https://doi.org/10.1109/MCOM.2018.1700333
5. Menon VG, Jacob S, Joseph S, Almagrabi AO (2019) SDN powered humanoid with edge computing for assisting paralyzed patients. IEEE Internet Things J:1. https://doi.org/10.1109/jiot.2019.2963288
6. Menon VG, Prathap J (2017) Vehicular fog computing. Int J Veh Telemat Infotain Syst 1:15–23. https://doi.org/10.4018/ijvtis.2017070102
7. Liu J, Zhang Q (2018) Offloading schemes in Mobile edge computing for ultra-reliable low latency communications. IEEE Access6:12825–12837.https://doi.org/10.1109/ACCESS.2018.2800032
8. Li S, Zhang N, Lin S et al (2018) Joint admission control and resource allocation in edge computing for internet of things. IEEE Netw 32:72–79.https://doi.org/10.1109/MNET.2018.1700163
9. Nadesh RK, Aramudhan M (2018) TRAM-based VM handover with dynamic scheduling for improved QoS of cloud environment. Int J Internet Technol Secur Trans:8. https://doi.org/10.1504/IJITST.2018.093340
10. Ning Z, Kong X, Xia F et al (2019) Green and sustainable cloud of things: enabling collaborative edge computing. IEEE Commun Mag 57:72–78. https://doi.org/10.1109/MCOM.2018.1700895

    P. G. et al. Journal of Cloud Computing: Advances, Systems and Applications (2020) 9:61 Page 11 of 12
11. Rajesh S, Paul V, Menon VG, Khosravi MR (2019) A secure and efficient lightweight symmetric encryption scheme for transfer of text files between embedded IoT devices, pp 1–21
12. Zhang J, Chen B, Zhao Y et al (2018) Data security and privacy-preserving in edge computing paradigm: survey and open issues. IEEE Access 6:18209– 18237. https://doi.org/10.1109/ACCESS.2018.2820162
13. Nadesh RK, Srinivasa Perumal R, Shynu PG, Sharma G (2018) Enhancing security for end users in cloud computing environment using hybrid encryption technique. Int J Eng Technol 7
14. Abbasi M, Rafiee M, Khosravi MR et al (2020) An efficient parallel genetic algorithm solution for vehicle routing problem in cloud implementation of the intelligent transportation systems. J Cloud Comput 9. https://doi.org/10.1186/s13677-020-0157-4
15. Subramanian N, Jeyaraj A (2018) Recent security challenges in cloud computing. Comput Electr Eng 71:28–42. https://doi.org/10.1016/j.compeleceng.2018.06.006
16. Jiang S, Jiang T, Wang L (2017) Secure and efficient cloud data Deduplication with ownership management. IEEE Trans Serv Comput 12:532–543. https://doi.org/10.1109/TSC.2017.2771280
17. Yoon MK (2019) A constant-time chunking algorithm for packet-level deduplication. ICT Express 5:131–135. https://doi.org/10.1016/j.icte.2018.05.005
18. Wang L, Wang B, Song W et al (2019) Offline privacy preserving proxy reencryption in mobile cloud computing. Inf Sci (Ny) 71:38–43. https://doi.org/10.1016/j.jksuci.2019.05.007

19. Wang L, Wang B, Song W, Zhang Z (2019) A key-sharing based secure deduplication scheme in cloud storage. Inf Sci (Ny) 504:48–60. https://doi.org/10.1016/j.ins.2019.07.058

20. Kwon H, Hahn C, Kim D, Hur J (2017) Secure deduplication for multimedia data with user revocation in cloud storage. Multimed Tools Appl 76:5889–5903. https://doi.org/10.1007/s11042-015-2595-4

21. Akhila K, Ganesh A, Sunitha C (2016) A study on Deduplication techniques over encrypted data. Procedia Comput Sci 87:38–43. https://doi.org/10.1016/j.procs.2016.05.123

22. Kwon H, Hahn C, Kang K, Hur J (2019) Secure deduplication with reliable and revocable key management in fog computing. Peer-to-Peer Netw Appl12:850–864. https://doi.org/10.1007/s12083-018-0682-9

23. Li J, Chen X, Li M et al (2014) Secure deduplication with efficient and reliable convergent key management. IEEE Trans Parallel Distrib Syst 251615–1625. https://doi.org/10.1109/TPDS.2013.284

24. Koo D, Hur J (2018) Privacy-preserving deduplication of encrypted data with dynamic ownership management in fog computing. Futur Gener Comput Syst 78:739–752. https://doi.org/10.1016/j.future.2017.01.024

25. Liu J, Wang J, Tao X, Shen J (2017) Secure similarity-based cloud data deduplication in ubiquitous city. Pervasive Mob Comput 41:231–242.https://doi.org/10.1016/j.pmcj.2017.03.010

26. Li S, Xu C, Zhang Y (2019) CSED: client-side encrypted deduplication scheme based on proofs of ownership for cloud storage. J Inf Secur Appl 46:250–258. https://doi.org/10.1016/j.jisa.2019.03.015

27. Tawalbeh LA, Saldamli G (2019) Reconsidering big data security and privacy in cloud and mobile cloud systems. J King Saud Univ - Comput Inf Sci.https://doi.org/10.1016/j.jksuci.2019.05.007

28. Zhang P, Huang P, He X et al (2017) Resemblance and mergence based indexing for high performance data deduplication. J Syst Softw 128:11–24.https://doi.org/10.1016/j.jss.2017.02.039

29. Hou H, Yu J, Hao R (2019) Cloud storage auditing with deduplication supporting different security levels according to data popularity. J Netw Comput Appl 134:26–39. https://doi.org/10.1016/j.jnca.2019.02.015

30. Khanaa V, Kumaravel A, Rama A (2019) Data deduplication on encrypted big data in cloud. Int J Eng Adv Technol 8:644–648. https://doi.org/10.35940/ijeat.F1188.0886S219

31. Stanek J, Kencl L (2018) Enhanced secure Thresholded data Deduplication scheme for cloud storage. IEEE Trans Dependable Secur Comput 15:694–707. https://doi.org/10.1109/TDSC.2016.2603501

32. Zeng P, Choo KKR (2018) A new kind of conditional proxy re-encryption for secure cloud storage. IEEE Access 6:70017–70024. https://doi.org/10.1109 ACCESS.2018.2879479

33. Wu J, Li Y, Wang T, Ding Y (2019) CPDA: a confidentiality-preserving Deduplication cloud storage with public cloud auditing. IEEE Access 7:160482–160497. https://doi.org/10.1109/ACCESS.2019.2950750

34. Awad A, Matthews A, Qiao Y, Lee B (2018) Chaotic searchable encryption for Mobile cloud storage. IEEE Trans Cloud Comput 6:440–452. https://doi.org/10.1109/TCC.2015.2511747

35. Shynu P G, John Singh K (2016) A comprehensive survey and analysis on access control schemes in cloud environment. Cybern Inf Technol 16:19–38.https://doi.org/10.1515/cait-2016-0002

36. Alazab M (2015) Profiling and classifying the behavior of malicious codes. J Syst Softw 100:91–102. https://doi.org/10.1016/j.jss.2014.10.031

37. Benzaid C, Lounis K, Al-Nemrat A et al (2016) Fast authentication in wireless sensor networks. Futur Gener Comput Syst 55:362–375. https://doi.org/10.1016/j.future.2014.07.006 Systems, IEEE, 2002, pp. 617–624.