https://doi.org/10.1186/s13677-020-00214-6

Journal of Cloud Computing:
Advances, Systems and Applications

RESEARCH

Open Access

A secure data deduplication system for integrated cloud-edge networks

Shynu P. G.[1], Nadesh R. K.[1], Varun G. Menon[2], Venu P.[3], Mahdi Abbasi[4*] and Mohammad R. Khosravi[5,6]

Abstract

Data redundancy is a significant issue that wastes plenty of storage space in the cloud-fog storage integrated environments. Most of the current techniques, which mainly center around the static scenes, for example, the backup and archive systems, are not appropriate because of the dynamic nature of data in the cloud or integrated cloud environments. This problem can be effectively reduced and successfully managed by data deduplication techniques, eliminating duplicate data in cloud storage systems. Implementation of data deduplication (DD) over encrypted data is always a significant challenge in an integrated cloud-fog storage and computing environment to optimize the storage efficiently in a highly secured manner. This paper develops a new method using Convergent and Modified Elliptic Curve Cryptography (MECC) algorithms over the cloud and fog environment to construct secure deduplication systems. The proposed method focuses on the two most important goals of such systems. On one side, the redundancy of data needs to be reduced to its minimum, and on the other hand, a robust encryption approach must be developed to ensure the security of the data. The proposed technique is well suited for operations such as uploading new files by a user to the fog or cloud storage. The file is first encrypted using the Convergent Encryption (CE) technique and then re-encrypted using the Modified Elliptic Curve Cryptography (MECC) algorithm. The proposed method can recognize data redundancy at the block level, reducing the redundancy of data more effectively. Testing results show that the proposed approach can outperform a few state-of-the-art methods of computational efficiency and security levels.

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

Introduction

The data gathered through different sources and the Emergence of the Internet of Things in all aspects of applications increases data volume from petabytes to yottabytes, necessitating cloud computing paradigm and fog networks to process and store the data. Cloud computing (CC) produces a network-centered environment vision to users which provides access to the internet, to a collective pool of programmable grids, servers, software, storage, and amenities that could be quickly freed, with less supervision and communication to the cloud service provider. Data processing in all ways is carried out remotely in the cloud server with the help of internet connectivity. Fog computing provides the local infrastructure to process the application locally and then connects to the cloud. The fog environment reduces delay when compared to the application connected to the cloud for processing. The application developed to process and store the data needs end-to-end security, communication protocols, and resources to access information stored in the cloud and fog environments. Smart applications are built with the help of sensors and actuators, and the data is stored in the cloud environment; and edge computing facilities are also used along with the local infrastructure, termed as fog, to process the data without delay. Internet of Things does not end up with an information system but tries to build a cyber-physical system [1]. Edge computing provisions the

* Correspondence: Abbasi@basu.ac.ir
[4]Department of Computer Engineering, Engineering Faculty, Bu-Ali Sina University, Hamedan 65178-38695, Iran
Full list of author information is available at the end of the article

feature of mobility for the user to process and store data on the move. Mobile edge computing provides seamless integrity among multiple applications, vendors, mobile subscribers, and enterprises [2].

Sending data to the cloud was the bulbous trend in the past decades, which is now changing to fog, edge, and cloudlet due to delay-sensitive and context-aware services. To address these challenges, the centralized cloud computing paradigm is moving to distributed edge cloud computing and this makes computing transparent [3]. Fog computing is an attractive solution to the distributed edge cloud computing for any type of applications and benefits in low-latency, mobility, and geo-distributed services distinguished from the cloud with several access control schemes [4–6]. When fog computing is considered the one-step solution for reducing computation tasks' latency, some schemes are described for offloading the task focusing on reducing the latency, energy efficiency, and reliability [7]. Admission control, computational resource allocation and power control are some of the critical parameters considered before offloading the intensive task from the cloud. The performance can be further improved only by the efficient resource allocation methods available for cloud and fog environment, thus increasing the reliability and transparency in application processing [8]. Resource allocation in a period allows the moving user to offload the task to the nearest cloudlet and extend the services from the fog environment. This type of offloading reduces delays in computational tasks with more significant mobility features [9].

Various application services impulse the possibility of edge computing by offering cloud capabilities at the network edge closer to mobile devices. Edge computing is an encouraging paradigm to decide several vital challenges in the Internet of Things in all domains, such as delay, low bandwidth, energy issues, latency in transmission, and data security and privacy [10, 11]. A comprehensive study of information security and privacy requirements, tasks, and tools in edge computing, cloud computing, fog computing and the cryptography-based technologies for solving security and privacy issues are analyzed before incorporating the cloud and fog networks [12, 13]. Hybrid encryption techniques using AES in CBC Mode and HMAC-SHA-1 (Hash-based Message Authentication Code) with lightweight procedures improve the robust encryption at user-level security in a cloud computing environment [14]. There are many more technical developments, but they exhibit other issues that have to be resolved, encompassing processing and storing data, securing sensitive information, and protecting user privacy [15].

Data deduplication (DD) stands as a universal data redundancy removal technology. DD primarily classifies identical data, stores one copy of data, and substitutes other similar copies with undirected references instead of keeping full copies [16]. DD involves three major processes: a) *chunking*, b) *hashing*, and c) *comparing* hashes to recognize redundancy. The chunking process breaks a file into many smaller files termed as chunks. The chunk level deduplication method ameliorates the storage of unique chunks by contrasting it with all incoming chunks for duplicate recognition [17]. Once the data is being uploaded to the cloud, the owner could not assure the security of the data in remote storage systems. Performing encryption is necessary to make data secure; simultaneously, performing deduplication is imperative for attaining optimized storage. Hence, encryption and deduplication should be done simultaneously for ensuring optimized and secured storage [18]. DD could be employed within a file, across files, across applications, or across clients over a particular period of time. It is utilized in archiving and backing up the file systems, databases with low change rate, Network Attached Storage, VMware environment, Local Area Network, and Storage Area Network. By adopting them, the key utilized for encryption and also decryption is itself attained from the data and would resist further attacks [19].

This paper proposes a secure data deduplication system using convergent and MECC algorithms over the integrated cloud-fog-edge environment. The convergent encryption appears to be the right choice for the implementation of deduplication with encryption in the cloud storage domain. But there is the possibility of *dictionary attacks* concerning this scheme as the encryption key is formed using the plaintext. An intruder who has gained access to the storage can compare the ciphertexts produced after the encryption of distinguished plaintext values from a dictionary where the ciphertexts are being stored. Moreover, even if encryption keys are encrypted with the users' private keys and stored somewhere else, the cloud provider, who has no access to the encryption key but has access to the encrypted chunks (blocks), can efficiently perform offline dictionary attacks and determine the expected data. Hence, to solve the above problem, the encrypted and deduplicated data using the CE are once again encrypted by the proposed modified elliptic curve cryptography (MECC) technique. The combined CE and MECC technique ensure efficient deduplication and secured encryption of cloud-fog-edge storage with less computational overhead compared to existing data deduplication techniques.

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

- A new method of constructing a secure deduplication (DD) system using Convergent and Modified Elliptic Curve Cryptography (MECC) algorithms over the cloud and fog/edge environment.

- Performance evaluation of the proposed technique, based on its computational efficiency and level of security is done.
- We validated the proposed deduplication technique's ability to recognize data redundancy in the level of blocks, which can reduce the redundancy of data more effectively to minimize the storage space in the cloud environment.

The draft structure of the manuscript is arranged as follows: section 2 surveys the associated works, section 3 provides the proposed methodology, section 4 explores the tentative outcome and section 5 contains the conclusion and scope for future work.

Related works

The secure deduplication system abandons the duplicate copies of data, and it also proffers security to the data. Convergent Encryption (CE) is utilized to encrypt or decrypt the data to the file level with a convergent key that is generated as the file content itself [20]. After encrypting such files, the cloud user just holds the encryption key and outsources the ciphertext (CT) to the CS to save storage space. By updating the CT saved in the central cloud and user-level public keys without knowing the private keys, consistent privacy is rendered [21]. Kwonet al [22]. proffered a secure deduplication framework with user revocations. The system comprises of '3' phases, namely: upload, revocation, and download. The proffered framework is executed via a privilege-centric re-encryption methodology over convergent-encryption. Liet al [23]. recommended Dekey, a construction wherein users need not handle any keys but rather securely disseminate the convergent key shares across multi servers. Dekey upholds the block-level and file-level deduplication. File-level deduplication eradicated the storage of any redundant files, and block-level deduplication separated the files into a smaller variable or fixed-sized blocks and wiped out the storage of any redundant blocks. Security analysis delineated that Dekey was secure. The Dekey centric Ramp secret centered framework elucidated that Dekey incurred limited overhead in factual environments.

Kwon et al. [24] recommended a deduplication framework at the server-side for the encrypted data. It permits the Cloud Server to control the access to the outsourced data even while the ownership altered dynamically with the secured ownership group key distributions and exploited randomized CEs. It can avert data leakage to the revoked users though they formerly owned the data and even to the cloud storage. The system assures data integrity like the tag inconsistency attack. The efficiency estimation results corroborated that the scheme was almost as effectual as the former framework, while the extra overhead in computations was insignificant. Yuanet al [25]. developed a primitive termed and a fully randomized framework (R-MLE2). It comprises of '2' schemes: i) static and ii) dynamic, where the latter one permitted tree adjustment by elevating specific computation cost. The primary trick of the framework was to utilize the interactive protocol centered on dynamic or static decision trees. The security and performance analyses evinced that the frameworks were Path-PRV-CDA2 secured which attained multiple orders of magnitude and high-level performance for the data equality tests, more than the R-MLE2 framework, when the count of data items was comparatively large. Han et al. [26] propounded a multi-bit secret channel in the cloud storage service, and also recommended a framework that attained good security and high-level data transmission rate. In the recommended algorithm, the data upload was simplified via multi-bit file depiction. It eradicated the need to upload "0" to diminish the number of uploaded files, thereby made it hard for the attacker to spot the covert channel and also effectually ameliorated the security of cloud user data upload. Tawalbeh et al. [27] reconsidered the security and privacy for cloud and fog environments with the case study of health care systems using fog simulator and enhanced the performance and trust among the end-users. Similarity and emergence centered indexing for high-performance deduplication of data was introduced by Zhanget al [28]. which provides quick responses to fingerprint queries. Houet al [29]. suggested to check the truthfulness of cloud data beneath the condition that the remote server stores only a single copy of the same file from different users.

Deduplication has confirmed to achieve great space and cost investment, and a higher number of distributed storage suppliers are currently embracing it. Deduplication can weaken capacity needs by up to 90–95% for corroboration [30]. As more users outsource their data to remote server storage, the latest data breach occurrences make end-to-end encryption increasingly desirable. Enhanced Secure Threshold Data Deduplication Pattern for remote storage helps to maintain end-to-end encryption [31]. A flexible admission control tool called Proxy re-encryption (PRE) has been recently hosted. PRE is an effective tool for creating cryptographically imposed admission control systems [32]. These schemes show competence in computational cost and ciphertext size.

A confidentiality-preserving deduplication technique for remote storage in public cloud services is discussed in [33]. The authors have proposed a secure file deduplication mechanism on the encrypted file, supporting public reliability and auditing in the deduplication of the cloud storage system. A chaotic fuzzy transformation method is projected to provision protected fuzzy keyword indexing, storage, and query for fog systems that aid in raising the privacy and confidentiality of the end-user data and also by saving the resources of the mobile user devices [34]. A comprehensive study on various security problems associated with outsourced data on the cloud and their existing solutions is described using access control models for the cloud computing environment [35].

A framework to mine structures statically and dynamically from malware that imitates the performance of its code, such as the Windows Application Programming Interface (API) classifies malware with high accuracy and low false alarm rates [36]. The public-key-based schemes obviate the security vulnerability inherent to symmetric-key-based μTESLA-like schemes. But their signature verification is time-consuming [37].

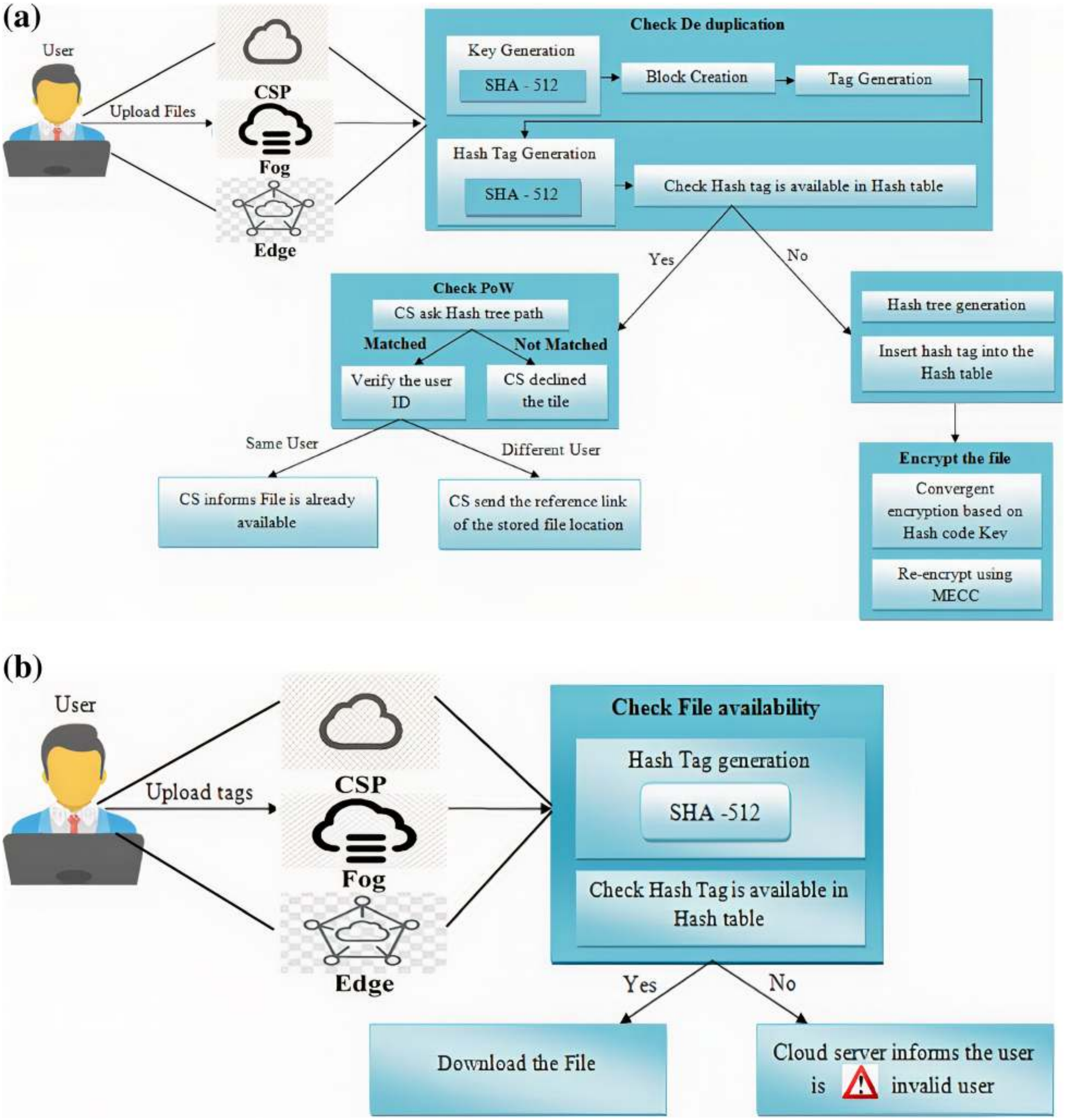

Fig. 1 a Uploading the file into the cloud server using the proposed system. **b** Downloading the file from the cloud server using the proposed system

Proposed secure deduplication approach

Cloud Service Provider provides many resources to users as a service, for instance, vastly available storage space. Managing the ever-elevating volumes of data in the cloud is a noteworthy task. The DD technique makes data management more scalable in CC. But security is the main problem in Data Deduplication. To overcome this problem, this paper proposes a secure data deduplication system using convergent and MECC algorithms over the integrated cloud-fog environment.

The proposed methodology is analyzed in four ways, i.e., a) when a new user tries to upload a new file, b) when the same user tries to upload the same file c) when different users try to upload the same file to the cloud server and d) when the users try to download the file. The proposed methodology could be expounded in detail using the block diagram evinced in Fig. 1a and b.

When a new user tries to upload a new file

Initially, the new user browses a file and uploads it to the CS. Then, the CS generates the hash code (HC) key for the appropriate file using SHA 512 algorithm. The input file is then appended with padding and fixed 128bit length field. The enlarged message is partitioned as blocks. A 64-bit word is derived as of the current message block utilizing 8 constants based on the square root of the first 8 prime numbers. In the subsequent level, a 512-bit buffer is updated. SHA-512 operation can be comprehended using Fig. 2, and then, the original input file is split into blocks. Next, tag values are assigned for each block and hash code (HC) is created for each tag value of a particular block utilizing the same SHA 512 algorithm. Cloud server (CS) verifies whether the hashtag is available in the HT. If it is unavailable, then the hash tree is generated for Proofs of Ownership (PoW) grounded on the hashtag value. Next, the file is encrypted using a convergent encryption (CE) method. The CE takes the HC key of the file as input. Next, the convergent based encrypted file is again encrypted utilizing the MECC algorithm.

The encryption process is done for securely uploading the data to the CS. The CE and MECC based encryption of the particular file is expounded as follows.

Convergent encryption (CE)

In data deduplication, CE improves data confidentiality. The convergence key (CK) is denoted as the generated hash code (HC) value of the file. Using this CK, all blocks of data copy are encrypted. To detect the duplicate file in the CSP, a tag will be derived for each data block. If two data files are the same, the same tags will be provided. Before storing the data file to the CSP, its tag would be forwarded to the CSP for detecting duplicate data files. At last, this encrypted data block, along with its tag, is saved in the CSP. The phases in Convergent encryption (CE) are described as follows.

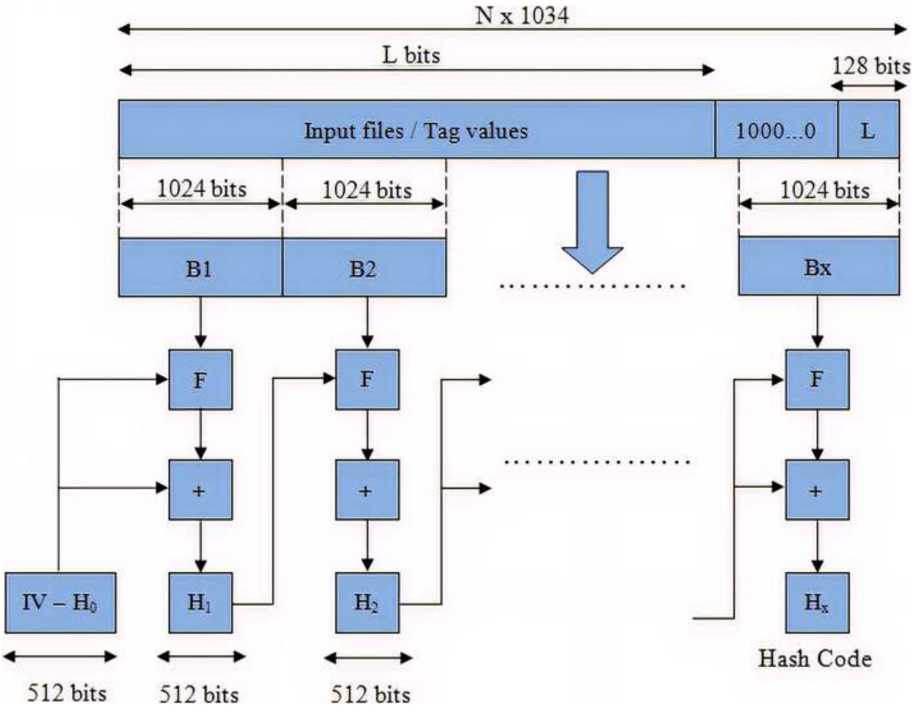

Fig. 2 Structure of the SHA-512 algorithm [35]

Convergent encryption

During encryption, the original file (*f*) and κ_h are given as input for the encryption algorithm and E_y is the encryption function. Finally, this encryption algorithm gives ciphertext (C_t) as output.

$$C_t = E_y(f, \kappa_h) \tag{1}$$

Convergent decryption

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

$$f = D_y(C_t, \kappa_h) \tag{2}$$

The CE algorithm gives better performance when compared with other existing methods, but it has a limitation as the CE is not secure since it may be affected by the dictionary attack. To avoid this, the proposed methodology once again encrypts the above convergent encrypted file utilizing the Modified ECC algorithm, which is discussed as follows.

Elliptic Curve Cryptography (ECC) algorithm is centered on a curve with specific base points and a prime number function. This function is utilized as a maximum limit. ECC is a kind of algorithm that is used in the implementation of public-key cryptography. The mathematical model of the ECC with *g* and *e* as integers is given below.

$$w^2 = v^3 + gv + e, 4g^3 + 27e^2 \neq 0 \tag{3}$$

In a cryptographic procedure, the potency of the encryption technique depends mainly on the mechanism that is deployed for the key generation. In the recommended system, three types of keys have to be generated. The main step is to generate the public key (α_k) from the server and encrypting it. In the next step, a private key (β_k) is produced on the server-side, and the message is decrypted. The last step is to generate a secret key (o_k) from α_k, β_k, and point on the curve (ρ_c). Using the succeeding equation, the α_k is generated,

$$\alpha_k = \beta_k * \rho_c \tag{4}$$

The eq. (5) elucidates o_k generation,

$$o_k = \alpha_k * \beta_k * \rho_c \tag{5}$$

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

$$C_1 = \left((K \coloneqq 1, 2, \dots, (n-1)) * \rho_c\right) + o_k \tag{6}$$

$$C_2 = (f + ((K \coloneqq 1, 2, \dots, (n-1)) * \alpha_k)) + o_k \tag{7}$$

Here, C_1 and C_2 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,

$$f = \left(\left(\left(C_2 - \beta_k\right) * C_1\right) - o_k\right) \tag{8}$$

When the same user tries to upload the same file

When the same user tries to upload the same file again, the CS calculates the hash value with the CK by utilizing the SHA 512 algorithm. Next, for every single input file, the binary depiction of the file is split into fixed-sized blocks. The size of the data block finds the level of granularity of deduplication. As the data block size decreases, the level of deduplication increases. Meanwhile, it might bring complex metadata management. The proposed approach considered the file block-sizes of 5 MB, 10 MB, 15 MB, 20 MB, and 25 MB. Then, the tag key is created for each of the divided blocks. Next, the hash value is computed for all the tag keys utilizing the same SHA-512 algorithm.

In the uploading phase, the CS checks the hashtag (HT) for a particular input file. If the hashtag value of the input file is in that HT, then the CS queries the path of the hash tree to the users. If a user sends the correct path, then the CS verifies the user id. If the id is the same, then the CS does not store the file again. Generally, the hash tree path has the succeeding format,

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

Where ,$P(H_t)$ denotes the path of the hash tree, *RLL* represents the "Root, Left, Left", *RLR*, denotes the "Root, Left, Right." The leaf node is not added to the hashed tree path. The same user trying to upload the same file is mathematically denoted as,

$$S_u \overset{Uploads}{\rightarrow} S_f\left(CS \overset{Informs}{\rightarrow} A\right) \tag{10}$$

Where, S_u denotes the same user, S_f represents the same file, and *CS* means the cloud server, which informs the file is previously available (A).

When different users try to upload the same file to the cloud server

When different users try to upload the same files to the CS, the file is split into several blocks, and a tag is created for checking the duplicate data copies in CSP. Then, each tag is converted into HC, and it is called a hashtag value. The CS checks the HT for the input file grounded on the hashtag value. If the hashtag value is available in the HT, then the CS asks the path of the hash tree of the input file. If a user sends the correct path, then the CS verifies the user id. If the id is different, then the CS sends the reference link of the particular stored file's location to the user. Different users trying to upload the same file to the CS is expressed as,

$$D_u \overset{Uploads}{\rightarrow} S_f\left(CS \overset{asks}{\rightarrow} P(H_t)\right) \tag{11}$$

$$P(H_t)_{matched} \rightarrow \left(CS \overset{Send}{\rightarrow} R_l(f)\right) \tag{12}$$

$$P(H_t)_{Not\ \ matched} \rightarrow \left(CS \overset{Informed}{\rightarrow} I_u\right) \tag{13}$$

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

When users try to download the file

Here, the user sends the tag value of the specified file. Then, the CS generates the hash value utilizing the SHA512 algorithm. The CS now checks the hashtag value, whether it is in the HT. If the value is available, then the CS lets the user download the file, else the CS considers them as an invalid user. It is mathematically denoted as,

$$H(T)_{matched} \rightarrow D_u(\downarrow) \tag{14}$$

$$H(T)_{Notmatched} \rightarrow I_u \tag{15}$$

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 key$k$, tag$T$, hashtag$H(T)$,
        blocks$(B_1, B_2, \ldots, B_n)$, and hash tree $H_t$
for$n$-number of files do
        {
        generate$k$ using SHA-512
        divide$f$ into blocks $(B_1, B_2, \ldots, B_n)$
        generate tag $T$`
        generate $H(T)$ using SHA-512
        if $(H(T) == Hash\_table)$then
        check PoW
        else
              generate$H_t$ 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
        {
        generate$H(T)$ using SHA-512
        if$(H(T) == Hashtable)$then
        Cloud Server allows the user to download-the
        file $f(\downarrow)$
        else
        Cloud Server informs as invalid user
        end if
        }
        end for
end
```

Result and discussions

The implemented deduplication methodology is deployed in the JAVA programming environment with the following system configuration. The system performance is analyzed, centered on different file sizes varying from like 5 MB to 25 MB with an increase of 5 MB after each iteration. In this section, the performance scrutiny is done on the proposed system. First, the performance revealed by the proposed MECC security algorithm is contrasted to the existing security algorithms, say, Diffie-Hellman (DH), ECC and Rivest Shamir Adelman (RSA) in respect of encryption time (Et), decryption time (Dt), key generation time, and security analysis.

Performance analysis of proposed encryption technique

Encryption time

E_t is considered as the time that an encryption algorithm consumes to generate encrypted data as of the inputted data. Encryption time is computed as the difference between the encryption ending time and encryption starting time. It is evaluated as,

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

File Size in MB	Encryption Time (sec)			
	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	21.0	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 (sec)			
	Proposed MECC	DH	Existing ECC	Existing RSA
5	5.28	12.21	13.50	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	36.87
25	25.44	45.44	45.81	48.43

$$E_t = E_e - E_s \tag{16}$$

Where E_t is the encryption time, E_e is the encryption ending time and E_s is the encryption starting time.

Decryption time

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

$$D_t = D_e - D_s \tag{17}$$

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,

$$S = {}^{H_d}\big/_{O_d} \tag{18}$$

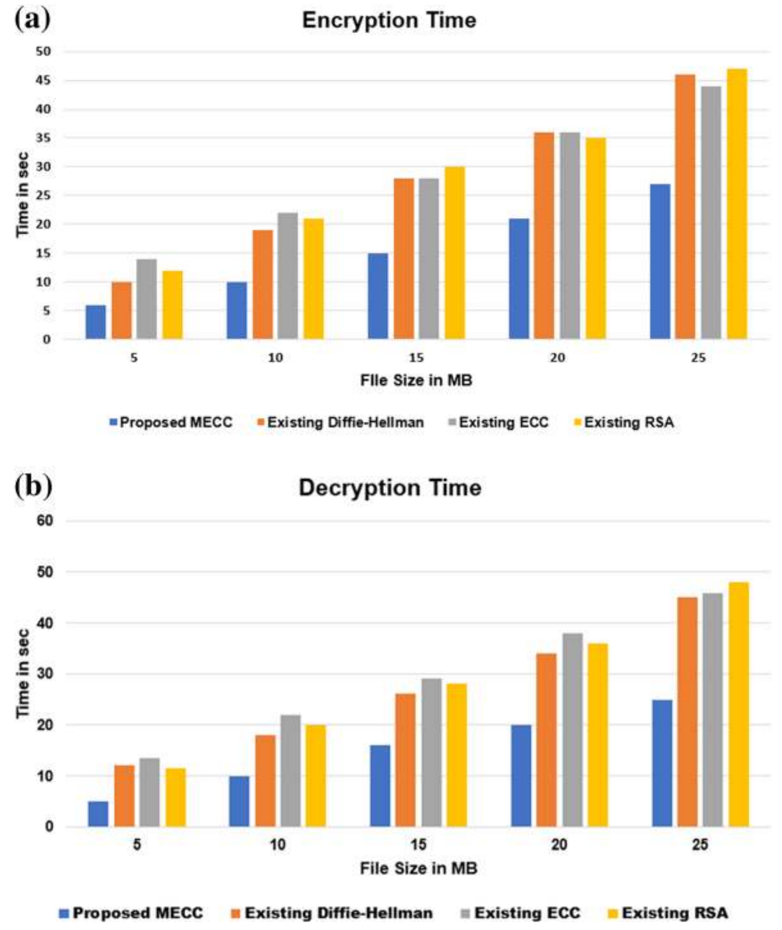

Fig. 3 a Performance comparison of proposed MECC with existing techniques in terms of encryption time. **b** Performance comparison of proposed MECC with existing techniques in terms of encryption time

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

Sl. No	Encryption Algorithms	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

where S denotes the security level, H_d is a hacked data, and O_d denotes the number of original data.

Tables 1 and 2 elucidate the performance comparison of proposed MECC with the prevailing DH, RSA, and ECC techniques concerning E_t and D_t. The comparison is performed, centered on the uploaded file size. The E_t and D_t are denoted in seconds (s). The proposed MECC takes 6 s to encrypt the 5mb file, whereas the existing DH, ECC, and RSA take 10, 14, and 12 s to encrypt the same 5mb file, which is high when contrasted to the proposed MECC. Similarly, for the remaining file sizes (10 to 25 MB), the proposed method takes less time to encrypt the data. For the same 5 Mb file, the proposed MECC takes 5 s to decrypt the data, but the prevailing DH, RSA, and ECC takes 12, 13.5, and 11.5 s to decrypt the data. So, it is inferred that the suggested MECC

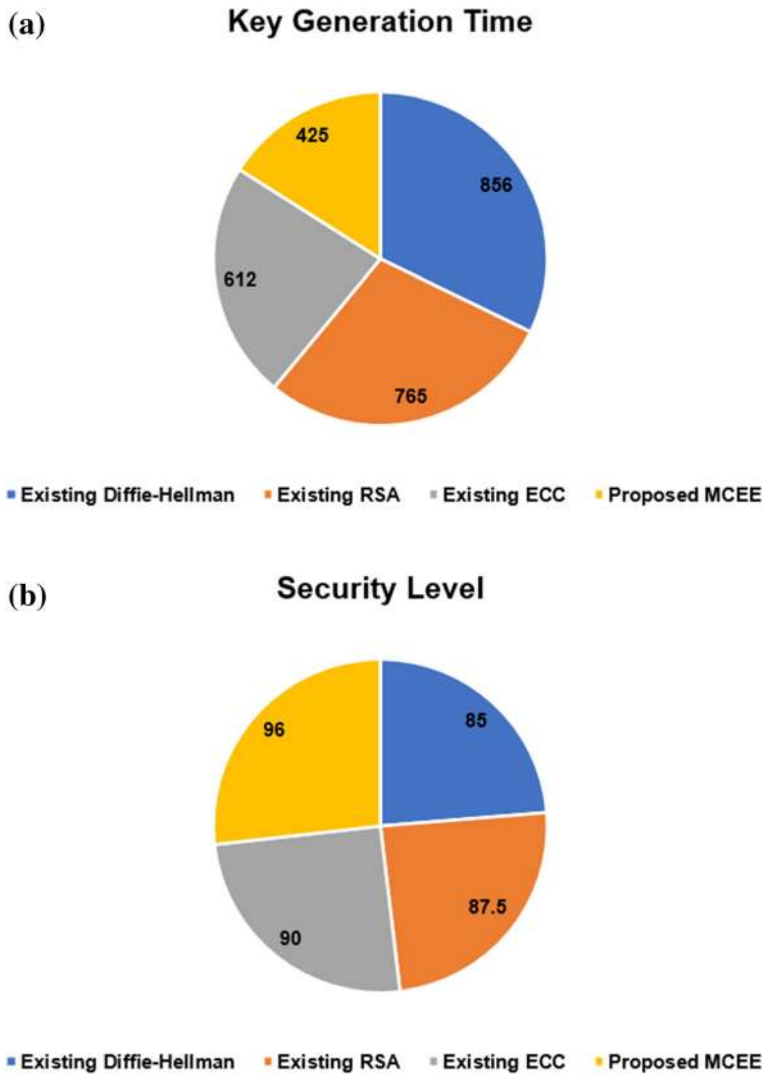

Fig. 4 a Performance comparison of proposed MECC with existing techniques in terms of key generation time. **b** Performance comparison of proposed MECC with existing techniques in terms of security level

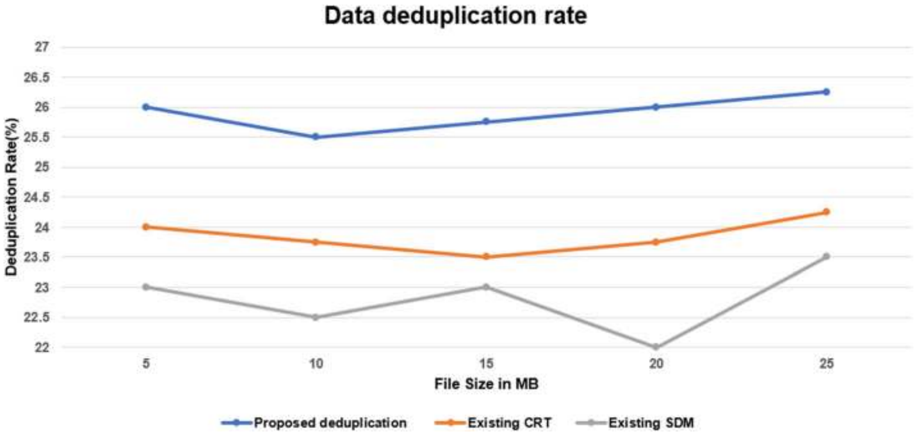

Fig. 5 Performance analysis of the proposed deduplication scheme with existing techniques in terms of deduplication rate

algorithm takes less E_t and D_t when contrasted to the remaining techniques. Tables 1 and 2 are graphically plotted and are displayed in Fig. 3a and b.

Fig. 3a and b analyze the performance proffered by the proposed MECC approach with other existing techniques. The E_t and D_t time varies centered on the file sizes. Here, the file sizes range from 5mb to 25mb. For 10mb file size, the E_t and D_t of the MECC are 10s, but the existing DH, RSA, and ECC take 19 s, 21 s, and 22 s for encryption and 18 s, 20s, and 22 s for decryption. Similarly, for all file sizes, the proposed MECC takes lesser Et and Dt. So, it is deduced that the MECC attains the best performance when contrasted to others.

Table 3 compared the performance rendered by the proposed MECC technique with the prevailing methods concerning the key generation time and security level. The proposed MECC takes 425 ms to generate a key, whereas the existing ECC, RSA, and DH methods take 612 ms, 765 ms, and 856 ms to generate a key. Here, the existing Diffie-Hellman (DH) method takes more time for key generation. But the proposed MECC takes less time to generate a key when contrasted to other techniques. Furthermore, the security level of the proposed and existing methods is compared with existing techniques. The proposed MECC gives the highest security value (96%), but the existing ECC, RSA, and DH methods give 90%, 87.5%, and 85% of security. So, it is inferred that the proposed MECC proffers high performance for both key generation and security. Table 2 is graphically illustrated as displayed in Fig. 4a for the Key Generation Time and Fig. 4b for the Security Level.

Performance analysis of proposed deduplication technique

The proposed deduplication scheme is contrasted to existing techniques such as the Chinese Remainder Theorem (CRT) centered secret sharing and Smart Deduplication for Mobile (SDM) in respect of deduplication rate, which is evinced in Fig. 5. Here, the proposed deduplication scheme is contrasted to other techniques concerning the deduplication rate and tree generation time.

Figure 5 contrasts the performance of the proposed deduplication with the prevailing methods, say CRT and SDM. The deduplication rate varies centered on the file size. For the 5mb file, the deduplication rate of the proposed method is 26%, whereas the existing SDM and CRT give 23% and 24%, which are relatively low when contrasted to the proposed method. For the 25mb file, the existing SDM and CRT give 23.5% and 24.2% of the

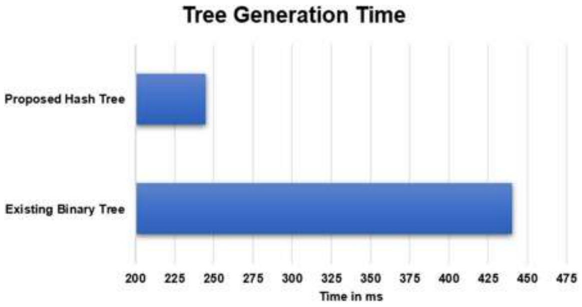

Fig. 6 Performance comparison of the proposed hash tree with existing binary tree

deduplication rate, but the deduplication rate of the proposed method is 26.2%. Similarly, for other file sizes such as 10mb, 15mb, and 20mb, the proposed deduplication scheme gives superior results contrasted to CRT and SDM. The performance comparison of the proposed hash tree used in deduplication with the existing binary tree in respect of tree generation time is evinced in Fig. 6.

Figure 6 contrasts the performance shown by the proposed hash tree with the existing binary tree regarding tree generation time. The proposed hash tree takes 245 ms to generate a tree, whereas the existing binary tree takes 440 ms to generate a tree, which is high when contrasted to the proposed hash tree generation approach. So it is deduced that the proposed hash tree approach shows high-level performance compared to binary tree generation methodology.

Conclusion

Deduplication is the utmost notable Data compression methodology. Many existing methods introduced different deduplication methods, but they provided low security. This paper proposed a secure deduplication system using convergent and MECC algorithms over the cloud-fog environment. The proposed method is analyzed in four ways: a) when new users try to upload the new file, b) when the same user tries to upload the same file, c) when different users try to upload the same file, and d) when different users try to download the file. The performance of the recommended system was analyzed by using various file sizes ranging from 5 MB to 25 MB, with an incremental of 5 MB each in every iteration. The performance analysis corroborated that the recommended system has 96% security, which is a promising result and higher than the other existing encryption methods.

The assessment result elucidates that the recommended system is extremely secure and effective for data deduplication for an integrated cloud environment. This proposed model may be extended in the future for any kind of Internet of Things (IoT) applications that use dynamic resources management at the edge environment. It can also be used in building cyber-physical systems by studying the different use cases having different payload with the variant data formats. The proposed technique would certainly be a promising model for increasing the security and optimizing the computation time and storage in an integrated environment such as IoT or cyber-physical systems.

Abbreviations

DD: Data deduplication; MECC: Modified Elliptic Curve Cryptography; PoW: Proofs of Ownership; CE: Convergent Encryption; SHA: Secure Hash Algorithm; CS: Cloud Server; CC: Cloud Computing; HMAC: Hash-based Message Authentication Code; CT: Ciphertext; PRE: Proxy re-encryption; ECC: Elliptic Curve Cryptography; CRT: Chinese Remainder Theorem (CRT); SDM: Smart Deduplication for Mobile; DH: Diffie Hellman Algorithm

Acknowledgements

Authors thank editor and reviewers for their time and consideration.

Authors' contributions

All authors have participated in the design of the proposed method and practical implementation. SPG and MRK have coded the method. SPG, NRK, VGM, VP, MA and MRK have completed the first draft of this paper. All authors have read and approved the manuscript.

Authors' information

Not applicable.

Funding

Not applicable.

Availability of data and materials

Not applicable.

Competing interests

The authors declare that they have no competing interests.

Author details

[1]School of Information Technology and Engineering, Vellore Institute of Technology, Vellore 632014, India. [2]Department of Computer Science and Engineering, SCMS School of Engineering and Technology, Ernakulum, Kerala 683576, India. [3]Department of Mechanical Engineering, SCMS School of Engineering and Technology, Ernakulum, Kerala 683576, India. [4]Department of Computer Engineering, Engineering Faculty, Bu-Ali Sina University, Hamedan 65178-38695, Iran. [5]Department of Computer Engineering, Persian Gulf University, Bushehr 75169-13817, Iran. [6]Department of Electrical and Electronic Engineering, Shiraz University of Technology, Shiraz 71555-313, Iran.

Received: 23 March 2020 Accepted: 5 November 2020
Published online: 19 November 2020

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 Access 6:12825–12837. 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

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 re-encryption 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 Appl 12:850–864. 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 25:1615–1625. 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

Publisher's Note

Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.