ISSN 2347 - 3983
Volume 11. No.9, September 2023
International Journal of Emerging Trends in Engineering Research
Available Online at http://www.warse.org/IJETER/static/pdf/file/ijeter011192023.pdf
https://doi.org/10.30534/ijeter/2023/011192023

A Secure Deduplication Technique for Data in the Cloud

Khulood Al-lehaibi [1], Afnan A.Alharbi[2]
[1] Department of Computer and Information Technology, Technical College for Girls in Makkah, Technical and Vocational Training Corporation, Saudi Arabia, khulood.a@tvtc.gov.sa
[2] Deanship of Scientific Research, Umm Al-Qura University, Saudi Arabia, aafharbi@uqu.edu.sa

Received Date: July 29, 2023 Accepted Date: August 21, 2023 Published Date : September 07, 2023

ABSTRACT

The tremendous growth of digital data in cloud storage systems is a critical issue, as many duplicate data in storage systems cause extra load. Cloud Service Providers (CSPs) often employ Data Deduplication techniques to eliminate redundant data and store only one copy of data to save storage space and reduce transmission costs. Data Deduplication is mostly effective when multiple clients outsource the same data to cloud storage, but it raises security and ownership issues. This paper proposes a secure, Proof of Ownership (PoW)-based Data Deduplication scheme that has a low communication overhead and ensures that only valid cloud clients can download and decrypt ciphertext from cloud storage. The Advanced Encryption Standard (AES) is used as the encryption algorithm in the proposed scheme. It utilizes two modes of AES encryption, namely, Cipher Block Chaining (CBC) and Galois Counter Mode (GCM), with single-threading and multi-threading to upload and download ciphertext between the client and the server to measure the effect of upload and download times. We present a new approach for PoW to reduce communication overhead. PoW enables owners of the same data to prove to the cloud server that they own the data in a robust way. The comparison between CBC and GCM is implemented in a Java environment with two scenarios: single-threading and multi-threading. The simulation results show that AES-GCM with multi-threading is better during the uploading and downloading times.

Key words: AES, Cloud Storage, CBC, Data Deduplication, GCM, PoW

1. INTRODUCTION

One of the most widely used cloud computing applications nowadays is remote data storage. Cloud data storage provides consumers with a large pool of shared resources (such as computing and storage) that they can use on a pay-per-use basis to meet their needs [1]. The use of cloud services has significantly increased as a result of the rapid evolution of Internet technology. The cloud computing platform has risen to prominence in the business due to its low operational and maintenance costs. As data rates increase, it is becoming increasingly important for Cloud Service Providers (CSPs) to invest in new solutions capable of optimizing storage and network bandwidth [2, 3]. CSPs use different techniques to obtain these solutions, and one of the most important techniques is data deduplication [4].

Data deduplication has been proven to achieve high-cost savings. It can reduce storage needs by 90%–95% for backup applications and up to 68% for standard file systems [5]. The data deduplication technique eliminates the same data and stores only one copy in the cloud to reduce storage space and network bandwidth. It was initially used as secondary storage before it was adapted for use as primary storage. Data deduplication is widely used by various CSPs, such as Dropbox, Amazon S3, and Google Drive [1, 5]. CSPs use data deduplication techniques to eliminate data duplication, which reduces the required storage space and network bandwidth and increases storage efficiency [1]. Data deduplication techniques based on Convergent Encryption (CE) are widely used in cloud storage systems to eliminate data duplication [6]. CE is a mechanism for ensuring data security while implementing data deduplication. When a data owner wants to store an encrypted file on a remote storage server, they must generate the enciphering key from the plaintext hash, and the file is encrypted using this key. The server receives the ciphertext, and the client keeps the encryption key to decrypt it later. Since CE is deterministic, identical files, regardless of who encrypts them, are always encrypted into identical ciphertext. As a result, the cloud server is able to perform Data Deduplication on the ciphertext. The CE scheme, on the other hand, has security flaws with tag consistency, PoW, and dynamic ownership management.

The security and PoW challenges are manifested as follows: original data content that is outsourced to cloud storage should not be revealed to anyone except the clients who own that data. The PoW process in the cloud efficiently allows data owners to prove to the cloud server that they still own the data. Most studies in this area focus on secure deduplication to provide

space efficiency and data security against inside and outside adversaries. However, the PoW approaches used in these studies increased the communication overhead, thus affecting their performance.

Security is a major issue in cloud-based deduplication systems. As clients share files, there is a security risk in which clients' private data can be leaked to unauthorized clients. As the majority of cloud services are provided in public domains, it is critical to ensure their security because public platforms can be accessed by a large number of users. Therefore, service providers must prioritize cloud data security and privacy when providing these services to cloud consumers. Ensuring the privacy of consumers data is one of the most concerns for security and privacy in cloud-based systems [7].

To overcome these problems, this study proposes a secure, PoW-based data deduplication scheme with a low communication overhead. The main objective of this paper is to implement a cloud-based Data Deduplication scheme capable of achieving a high degree of security and enhancing overall system performance. The main contributions of this paper are as follows:

- Design a file-level deduplication scheme to satisfy the following security requirements: data privacy, data integrity, forward secrecy, and backward secrecy.
- Select the best mode of the Advanced Encryption Standard (AES) encryption algorithm to achieve high security with high performance.
- Improve the performance of uploading and downloading in cloud computing using multi-threading.
- Present a new approach for PoW to reduce communication overhead.

2. RELATED WORK

Secure data deduplication with high performance has become an essential issue in modern storage systems, particularly in the cloud storage environment. In recent years, there has been a significant increase in research interest in data integrity and file deduplication. Several CE-based deduplication schemes have been proposed to address the issue of data deduplication on cloud data storage [1]. However, there are several security risks associated with CE-based deduplication schemes such as tag consistency, PoW, and dynamic ownership management.

In [8], a Message-Locked Encryption (MLE) with Randomized Convergent Encryption (RCE) scheme was introduced to address the tag consistency problem by introducing an additional integrity check phase for decrypted data. However, the revocation of ownership poses a security risk in this scheme. As a result, revoked cloud clients can access the corresponding data in the cloud storage as long as they keep the encryption key, regardless of the validity of their ownership. The study in [9] introduces a cryptographic method for secure PoW, based on CE and the Merkle-based Tree, to improve data security in the cloud, provide dynamic sharing between users, and ensure efficient Data Deduplication. Their approach used the Merkle-based Tree over encrypted data to derive a unique identifier. It is used to check the availability of the same data in a cloud server. If the file does not exist, the client sends the encrypted file. If the file exists, the CSP verifies client ownership by sending random leaves indices of the associated Merkle Tree. The client will compute each leaf's associated sibling path to prove his ownership. Thus, their approach raises communication overhead due to the transmission of the complete Merkle Tree. It requires high computation at the cloud client side and cloud server side. The CSP identifies clients as data owners while outsourcing the same data. The scheme also allows a data-sharing process. The client stores data in the cloud and authorizes a group of users to access the data by enciphering the decryption key with the public keys of authorized users and sending it with the encrypted file that they want to store in the cloud and share. The authorized user will be decrypting the key using his private key. Then, the user uses the derived key to decrypt the requested data file. In [10], a PoW solution was proposed that depends on previous data owners and a trusted third party. The server provides a file to previous owners and a trusted third party for defining PoW.

In [11], the researchers proposed a decentralized block-level data de-duplication framework for big data management in cloud Systems. In order to improve deduction efficacy and reduce workload, the proposed approach employed a two-level routing decision for directing the file after clients based on data similitude and locality. Chouhan et al. [12] presented a secure data de-duplication framework for encrypted data. The proposed method improves data and key privacy and reliability by securely storing their fragments in a distributed fashion based on the duplicateless encryption for a simple storage scheme. According to the results, the proposed approach achieved reliability while incurring an average storage overhead of 66.66%. In [13], the researchers presented a block-level data deduplication technique. The CE algorithm is used to check for duplicate data copies in the CSP. Then, to ensure secure data storage, an enhanced symmetric key encryption algorithm is used. The proposed approach also employs the Spider Monkey Optimization algorithm to optimally select the secret key. The results of simulations revealed that the proposed approach performed well in terms of encryption time, decryption time, and computation time.

In [14] Wu et al. proposed a secure file deduplication method for encrypted files to enhance availability to the cloud data storage. The proposed solution maintains data confidentiality during file deduplication by combining convergent encryption and random masking techniques. In [15], the researchers proposed a secure data deduplication technique that utilizes convergent and modified Elliptic Curve Cryptography algorithms. The proposed method determines data redundancy at the block level, files are first encrypted using convergent encryption, and then re-encrypted using the Modified Elliptic Curve Cryptography algorithm. According to the performance analysis, the recommended system has 96% security.

3. METHODOLOGY AND IMPLEMENTATION

This section presents the methodology and implementation of the proposed system model.

3.1 Component of the proposed System Model

The system model as illustrated in Figure 1 has two entities, namely, the cloud client and the cloud server, which are involved in file-level deduplication. Cloud client: This is an entity that stores files in cloud storage and accesses these files later. It uploads encrypted files and requests to download the uploaded files later. Cloud server: This server provides cloud storage services, including deleting redundant files and storing only a single copy of these files. The cloud server manages the metadata that includes tags of the stored files, the identities of cloud clients' and the corresponding signatures. Furthermore, it checks the cloud client's identification before it downloads the file. The cloud server is assumed to be honest but curious. It executes the protocol honestly and is curious about the contents of the stored files.

3.2 Methodology of the Proposed System Model

In general, the proposed system model consists of two main parts: the client side, which is managed by the cloud user, and the server side, which is managed by the cloud server. They utilize multi-threading to ensure parallel running in the upload and download processes to enhance overall system performance. The proposed model as shown in Figure 2 executes the deduplication workflow stages between the client side and the server side, as follows:

- The first three stages of the deduplication workflow (hashing, chunking, and encryption) are set on the client side.
- Both indexing and PoW verification are set on the server side.

The rest of this section discusses the methodology used to build the proposed model.

3.2.1 Hashing Stage

The hashing stage is developed on the client side. The proposed scheme is based on CE, that is, deriving an encryption key by applying a one-way hash function on data content. The hash function adopted in the proposed scheme is SHA-256, which is used on the file to generate an encryption key. The encryption key is then used with AES symmetric encryption to encrypt the file. Afterward, SHA-256 is used to generate a tag from the encrypted file. This hash value is also used with a private client key to generate the signature. Figure 3 shows how SHA-256 converts files into a fixed hash value.

3.2.2 Indexing Stage

This stage indexes the fingerprints produced to distinguish between duplicate and non-duplicate files by comparing the new tag with those in the index. The duplicated files are deleted, leaving only a unique file to be stored. This can be considered the most critical stage in the deduplication workflow. In the proposed scheme, the client sends the tag to the cloud server during the upload request and searches the index table for the presence of the tag. If the tag is not present, the client uploads encrypted chunks of the file. The client is called the "First Uploader." Any other file with the same tag will not be allowed to be uploaded. A new record for the client in the metadata will be created without uploading the file to avoid duplicating data. By doing so, the storage space will be saved, and the transfer time will be reduced.

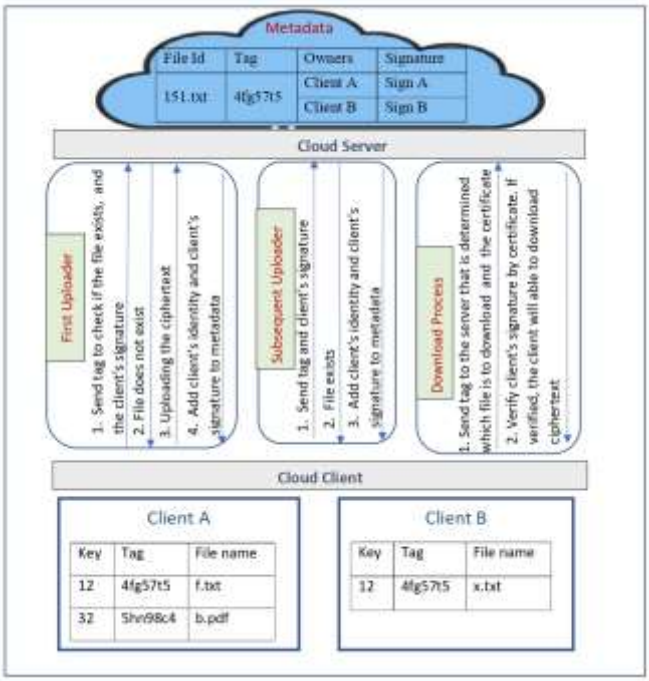

Figure 1: The Proposed System Model.

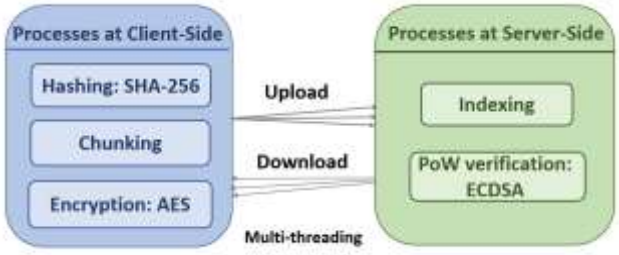

Figure 2: The Data Deduplication Stages.

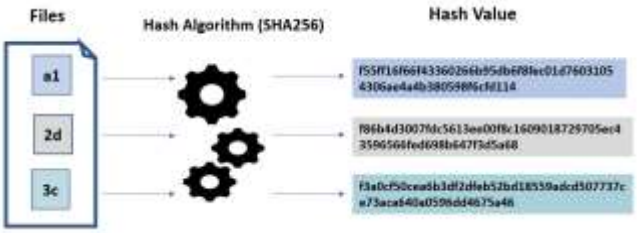

Figure 3: The SHA-256 Algorithm.

3.2.3 Chunking Stage

Chunking involves breaking a large file into small pieces called "chunks" or "blocks." The proposed scheme only considers file-level deduplication and uses chunking to improve performance. As encryption and decryption take

time, the file is encrypted in chunks through multi-threading, which will reduce encryption and decryption times. Then, the client uploads the encrypted chunks to the server. The upload and download processes are performed in parallel mode for higher speed. The same procedures are applied for downloading the file. In the end, all chunks will be merged.

3.2.4 Cryptography Stage

Cryptography is an important stage in deduplication-based cloud storage systems to provide confidentiality. The original data content outsourced to cloud storage should not be revealed to anyone except the client that owns the data. There are many cryptographic algorithms in the literature, and deciding which one to use is an important issue. The method used in this study is based on In-House key management. The encryption keys are stored on the client side without sending them to the server side. An encryption algorithm capable of quick encryption using the two modes of AES, namely, Cipher Block Chaining (CBC) and Galois Counter Mode (GCM), is chosen. The reason for selecting these two modes is that CBC is considered the most common mode for general use, while GCM is parallelizable [16]. Although the AES-CBC and AES-GCM modes involve a block cipher and an exclusive-OR (XOR), they work differently internally.

1. In the CBC mode, a block of data is encrypted by taking the current plaintext block and XORing it with the previous ciphertext block.

2. In the GCM mode, a counter is assigned for each block of data, and the current value of the counter is sent to the block cipher. The output of the block cipher is XORed with the plaintext to obtain the ciphertext. The counter mode of the operation is designed by transforming block encryption into stream encryption.

3.2.5 PoW Stage

PoW in the proposed scheme is implemented by ECDSA using the NIST P256 curve. ECDSA is used for the verification process without retrieving data, which reduces the communication overhead. The steps involved in ECDSA are the formation of the key pair generator, signature generation, and signature verification.

1. Key Pair Generator

The ECDSA method has the advantage of having a lower key size. It helps to minimize the computation overhead and computation time of the integrity check [17]. ECDSA allows constructing strong public and private key pairs with key lengths that are far shorter than those of Rivest, Shamir, and Adleman encryption algorithms (RSA). Furthermore, a 256-bit Elliptic Curve Cryptography (ECC) key should provide similar security to a 3072-bit RSA key [18]. This algorithm generates a private–public key pair for every client. The client's key pair is constant.

2. Signature Generation

The sender is the client. The client signs the file hash with its private key to generate the signature and sends it to the cloud server. The cloud server then stores this signature in the metadata. If another client tries to upload the same file, it needs to generate a signature and send it to the cloud server to be saved in the metadata. As each file has a distinct signature, any client that does not have the file cannot convince the cloud server that it owns the file.

3. Signature Verification

The cloud server verifies the client's signature, which is present in the metadata against the client certificate during the download request. If the signature is verified, the client is authorized and will be able to download the file, as the client has been confirmed to be the file owner. Decryption is conducted on the client side. When a client deletes a file, the cloud server deletes the client's identity and his/her signature from the metadata. This means that the client cannot retrieve the deleted file. If a new client uploads a file, the cloud server inserts the client's identity and the client's signature into the metadata. Note that a client cannot access and download the file until it verifies its certificate. Moreover, any client cannot download another client's files.

3.3 Design Proposed Scheme Algorithms

This section presents the algorithms of the proposed scheme for uploading, downloading, and deleting operations.

3.3.1 Upload Operation

Algorithm 1 shows a detailed overview of the upload operation in the proposed scheme, in which the file is transferred from a client machine and uploaded to cloud storage. First, the proposed scheme uses the hash of a file and splits a file into multiple chunks. Second, it encrypts all chunks in parallel. Finally, it uploads all chunks to cloud storage in parallel. Algorithm 2 provides an overview of the cloud server to verify the tag presence in the metadata. Algorithm 3 shows an overview of the cloud server verifying the corresponding tag with the encrypted file for the "First Uploader." The cloud server calculates tag1 for the received encrypted file and then compares tag1 with the provided tag.

Algorithm 1 Upload File

procedure CLOUD CLIENT: UPLOAD FILE($File$)
  $key \leftarrow Hash(File)$ Split File into small chunks $C_i, where i = 1, 2, ..., n$
  for each chunks in C_i **do**
    $C_c \leftarrow Encrypt(Key, C_i)$ ▷ where C_c is encrypted chunks
  end for
  $C_f = C_{c1} + ... + C_{cn}$ ▷ where C_f is encrypted file
  $Tag \leftarrow Hash(C_f)$
  EC generate(Public key, Private key) ▷ where EC is elliptic curve
  $Sign$ ←ECDSA Signature Generate(Tag, Private key)
  $Presence \leftarrow$ Verify Presence(Tag, Sign)
  if $Presence = FALSE$ **then**
    $Upload(Tag, Sign, C_c)$
  else if $Presence = TRUE$ **then**
    Save the client's signature $(Sign)$ in metadata
  end if

3.3.2 Download Operation

Algorithms 4 and 5 show a detailed overview of the download operation in the proposed scheme, in which the file is transferred from cloud storage until it is downloaded to a client machine. First, the cloud server checks which file the client wants to download through the tag and verifies the client's signature, which is presented in the metadata against the client's certificate sent during the download process. Second, if it passes the verification process, the server downloads the file chunks from cloud storage in parallel. Finally, the client decrypts and assembles all chunks to obtain the original file.

3.3.3 Delete Operation

Algorithm 6 provides a detailed overview of the delete file operation in the proposed scheme. First, the cloud server checks which file the client wants to delete through the tag and verifies the client's signature, which is presented in the metadata against the client's certificate. Second, if it passes the verification process, the cloud server deletes the client's identity and the client's signature from the metadata. However, the file remains until the last owner deletes it.

Algorithm 2 Verify Presence

procedure CLOUD SERVER: VERIFY PRESENCE($Tag, Sign$)
  $key \leftarrow Hash(File)$
  if ($Tag \in metadata$) **then**
    return $True$
  else
    return $False$
  end if

Algorithm 3 Verify Presence

procedure CLOUD SERVER: VERIFY CORRESPOND(Tag, C_f)
  $Tag_1 \leftarrow Hash(C_f)$
  if ($Tag_1 == Tag$) **then**
    Save C_f
  else
    Cloud server deletes the encrypted file, cancel upload request
  end if

Algorithm 4 Download File

procedure CLOUD CLIENT: DOWNLOAD FILE($Tag, Certificate$)
  $Response \leftarrow VerifyDownload(Tag, Certificate)$
  if ($Response = True$) **then**
    for (all chunks in C_i) **do**
      $C_i = Decrypt(Key, C_c)$
    end for
  end if
  Assemble all chunks C_i to get the file

4. EXPERIMENTAL RESULTS AND COMPARISON

4.1 Investigation of the Proposed Scheme's Security Requirements

This section presents four theorems for the security requirement investigation in the proposed scheme: data privacy, data integrity, forward secrecy, and backward secrecy.

4.1.1 The Proposed Scheme Data Privacy Investigation

Data privacy is defined as keeping data secret from the cloud server and unauthorized clients that cannot prove ownership. In the proposed scheme, the cloud server is considered honest but curious. Two scenarios were applied to verify data privacy. First, assuming that an unauthorized client wants to download a file, the client sends a tag and a certificate along with the request. The server verifies whether the client's signature is present in the metadata or not through the certificate. The server then rejects it as an invalid signature. In this case, the client's signature does not exist in the metadata, thus preventing the client from downloading the encrypted file. Moreover, the encryption key is only on the authorized client side. The second scenario can be launched by the cloud server. The cloud server can know the encrypted file but cannot guess the key because it is derived from the hash of plaintext. Therefore, the cloud server cannot decrypt the encrypted file and obtain the original file. This shows that data privacy has been verified for the proposed scheme.

4.1.2 The Proposed Scheme Data Integrity Investigation

Integrity of the data means it must validate all information, and it must be what it claims to be, originally sent. The deduplication algorithm guarantees tag consistency against any poison attacks and allows valid clients to verify that the data downloaded from the cloud have not been altered. Let us assume that an attacker and a cloud client have the same data M. The attacker uses the correct tag T and other ciphertexts to damage data integrity. Initially, T is generated from M and ciphertext C$^\wedge$ from M$^\wedge$ = M, and then C$^\wedge$ is uploaded with T . In the proposed scheme, a poison attack on tag consistency can be detected, as the tag is generated from ciphertext instead of plaintext. Therefore, when the "First Uploader" wants to upload a file, it will send the tag (T) and the encrypted file. The cloud server will compute the tag T$^\wedge$ from the encrypted file. Then, If T$^\wedge$ = T , then it will store the tag with the corresponding encrypted file. If T$^\wedge$ = T , then it will delete the encrypted file and cancels the upload request.

4.1.3 The Proposed Scheme Forward Secrecy Investigation

Forward secrecy means that when a group of clients shares the same data in cloud storage, some clients may request this data deletion from cloud storage. The cloud server should prevent them from accessing the data after deletion. To some extent, the proposed scheme can guarantee forward secrecy.

When a cloud client deletes a file, the owners of the file are updated immediately. Hence, the cloud server deletes the client's identity and the client's signature from the metadata. Therefore, the cloud client cannot pass on the PoW to download the deleted file.

4.1.4 The Proposed Scheme Backward Secrecy Investigation

Backward secrecy means that the cloud server should prevent any client from accessing the data before uploading. The proposed scheme can guarantee backward secrecy to some extent. It means that before the second cloud client uploads the file, this file already exists in cloud storage (by the "First Uploader"), and the client identity and signature of the second cloud client do not exist in the metadata. As a result, this client cannot pass on the PoW and is prevented from accessing and downloading the file.

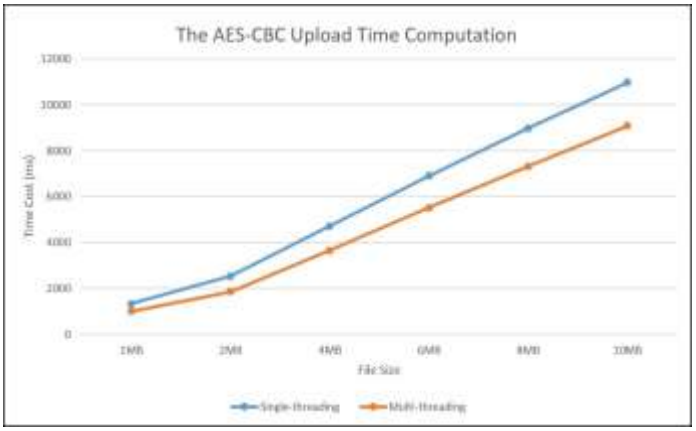

Figure 4: AES-CBC Upload Time Computation Using Single-threading and Multi-threading.

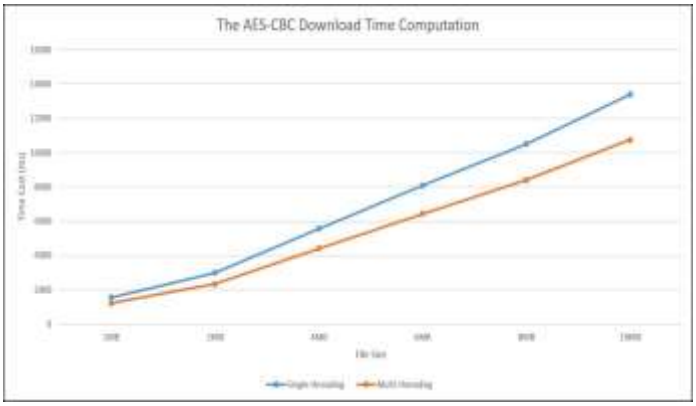

Figure 5: AES-CBC Download Time Computation Using Single-threading and Multi-threading.

4.2 Performance of the Proposed Scheme

The performance was improved using multi-threading and AES-GCM. The effects of multi-threading were measured by comparing it with single-threading. The AES encryption was changed from CBC to GCM in order to measure its effect on performance. The performance improvement was evaluated by comparing the upload and download computation times for different file sizes using single-threading and multi-threading. The AES-CBC upload time and download time computations using single-threading and multi-threading are shown in Figures 5 and 5, respectively. The AES-GCM upload time and download time computations using single-threading and multi-threading are shown in Figures 4 and 5, respectively. The AES-CBC comparisons between the upload and download times when single-threading and multi-threading are used to upload and download different file sizes are 1, 2, 4, 6, 8, and 10 MB. In the multi-threading mode, the chunks were partitioned by 200 KB. As shown in Figures 5 and 5, AES-CBC with a multi-threading mode is better than AES-CBC with a single-threading mode in terms of upload and download time costs.

The AES-GCM comparisons between the upload and download times when single-threading and multi-threading are used to upload and download different file sizes are 1, 2, 4, 6, 8, and 10 MB. In the multi-threading mode, the chunks were partitioned by 200 KB. As shown in Figures 6 and 7, AES-GCM with a multi-threading mode is better than AES-GCM with a single-threading mode in terms of upload and download time costs. Therefore, the AES-GCM multi-threading mode is considered to be better than the AES-CBC multi-threading mode in terms of upload and download time costs as highlighted in Figures 8 and 9.

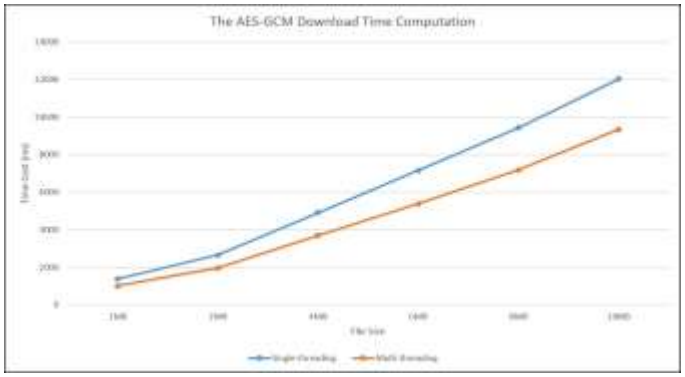

Figure 7 AES-GCM Download Time Computation Using Single-threading and Multi-threading.

4.3 Comparison with Related Works

This section compares the familiar and different aspects of this study with other works. The common feature between this work and other works in the literature is that they both focus on data security and file-level data deduplication in the cloud and providing PoW. In what follows, we compare our propose approach with previous works. We found that the main difference lies in the PoW technique. Specifically, this work is based on ECDSA, which reduces the communication overhead because the verification process does not involve retrieving the data. Conversely, most previous research used PoW approaches that enable the cloud server to act like a challenger, which sends challenges to the cloud client and waits for its response. Table 1 shows a comparison between the existing schemes and the proposed scheme based on the

type of Data Deduplication and the pow.

As shown in Table 1, the PoW technique used for the scheme [9] is the Merkle tree. This approach increases the communication overhead due to the transmission of the complete Merkle tree. It requires high computation on the client side and the server side. Conversely, the PoW used in SDDOM [19] is KEK, which does not support new client cloud joining. If a new client wants to insert a new leaf node of the binary KEK tree, it will take considerable overhead to reconstruct the binary KEK tree. In the DedupDUM [20] scheme, PoW is the ElGamal algorithm. Before a cloud client can download data, the cloud server encrypts a random number using the cloud client's public key to check its identity. The cloud client with the corresponding secret key can obtain the encrypted number. If the cloud client passes the verification process, the cloud server transmits the ciphertext to the cloud client. This process increases both communication and computational overhead. After measuring the effect of multi-threading and AES-GCM on reducing both upload and download times, the effectiveness of ECDSA in reducing upload and download time was determined by comparing it with an existing scheme [9]. The smaller scope was chosen for two reasons. First, the two previous studies in Table 1, SDDOM [19] and DedupDUM [20], did not explain their algorithms; thus, they could not be reconstructed. Second, the two previous studies [19, 20] in Table 1 did not measure upload and download times. As mentioned before, Java environment was used to implement the proposed scheme and rebuild the existing scheme [9]. The experiment was conducted on a Windows 10 laptop with an Intel (R) Core (TM) i7-7500 CPU at 2.70 GHz and 16.0 GB RAM.

The upload times of the different file sizes of 1, 2, 4 6, 8, and 10 MB were evaluated for the proposed scheme and existing scheme [9]. The result of the upload time computation is shown in Figure 10, which indicates that the proposed scheme takes less time to upload the same files than the existing scheme [9]. The download times of the different file sizes of 1, 2, 4 6, 8, and 10 MB were evaluated for the proposed scheme and existing scheme [9]. The result of the download time computation is shown in Figure 11, which indicates that the proposed scheme takes less time to download the same files than the existing study [9].

Figure 8: The Upload Time Comparison for AES-GCM vs. AES-CBC.

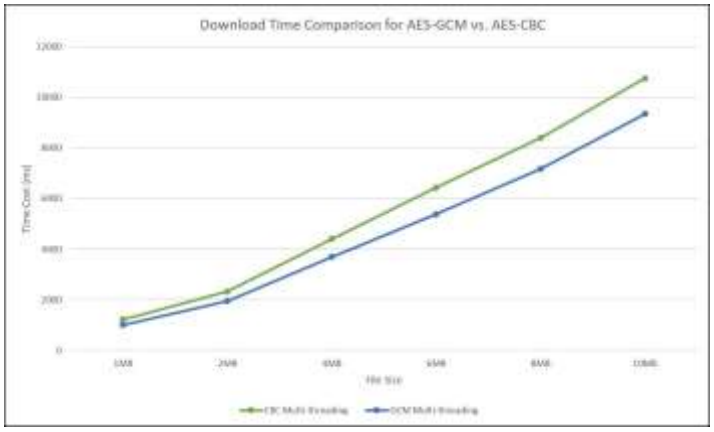

Figure 9: The Download Time Comparison for AES-GCM vs. AES-CBC.

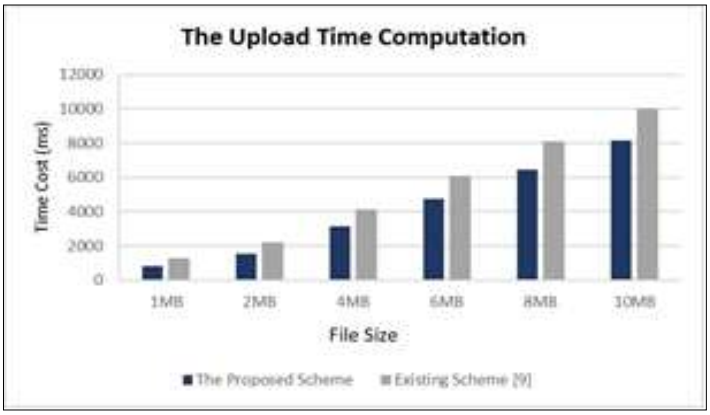

Figure 10: Upload Time Computation of the Proposed Scheme vs. Existing Scheme [9].

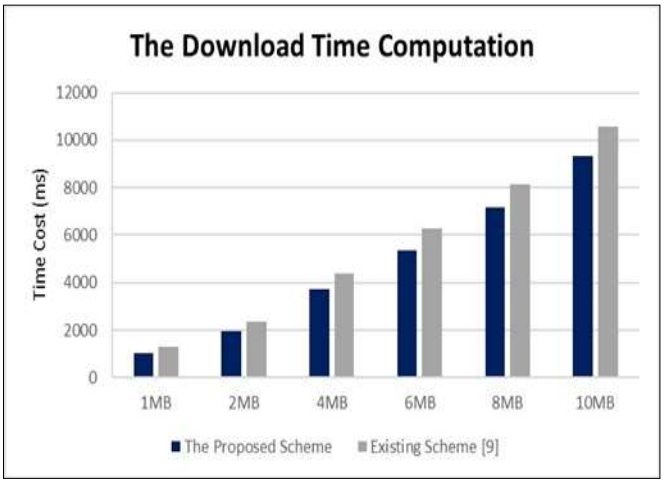

Figure 11: Download Time Computation of the Proposed Scheme vs. Existing Scheme [9].

5. CONCLUSION

Cloud services have become popular with the significant growth of digital data, as they provide clients with efficient and convenient storage services. CSPs usually employ data deduplication techniques to eliminate data duplication and save space on their platforms. Data deduplication is most effective when multiple clients store the same data in cloud storage, but it raises security and ownership issues. This paper proposed a secure, with PoW Data Deduplication scheme that has a low communication overhead and restricts unauthorized cloud clients from accessing and downloading data owned by valid clients. Two modes of AES encryption, CBC and GCM, were studied using single-threading and multi-threading to upload and download ciphertext between the client and the server to measure the effect of upload and download times. The ciphertext was uploaded from the client to the server, and the encryption key was saved on the client side without affecting the deduplication process. PoW was provided using the signature algorithm ECDSA. The simulation results show that AES-GCM with multi-threading is better during the uploading and downloading times. The security requirements that have been verified in this proposed scheme are data privacy, data integrity, forward secrecy, and backward secrecy for data deduplication in the cloud storage environment. The proposed scheme outperforms the existing work in both upload and download times.

Table 1: Features of the Proposed Scheme and Other Schemes.

Scheme	Scheme [9]	DedupDUM [20]	SDDOM [19]	Proposed Scheme
Deduplication type	File	File	File	**File**
Hashing type	SHA-256	SHA-128	MD5	**SHA-256**
Encryption type	AES-CBC	AES-CBC	AES- ECB	**AES-GCM**
PoW technique	Merkle tree	ElGamal	KEK	**ECDSA**

REFERENCES

1. P. Prajapati and P. Shah, **A review on secure data deduplication: Cloud storage security issue**, *Journal of King Saud University-Computer and Information Sciences*, vol. 34, no. 7, pp. 3996–4007, 2022.
2. A. Shakarami, M. Ghobaei-Arani, A. Shahidinejad, M. Masdari, and H. Shakarami, **Data replication schemes in cloud computing: a survey,** *Cluster Computing*, vol. 24, no. 3, pp. 2545–2579, 2021.
3. H. Tabrizchi and M. Kuchaki Rafsanjani, **A survey on security challenges in cloud computing: issues, threats, and solutions,** *The journal of supercomputing*, vol. 76, no. 12, pp. 9493–9532, 2020.
4. X. Yu, H. Bai, Z. Yan, and R. Zhang, **Veridedup: A verifiable cloud data deduplication scheme with integrity and duplication proof,** *IEEE Trans. on Dependable and Secure Computing*, vol. 20, no. 1, pp. 680–694, 2023.
5. P. Puzio, R. Molva, M. Önen, and S. Loureiro, **Cloudedup: Secure deduplication with encrypted data for cloud storage** in 2013 *IEEE 5th International Conference on Cloud Computing Technology and Science*, vol. 1. IEEE, 2013, pp. 363–370.
6. G. Zhang, Z. Yang, H. Xie, and W. Liu, **A secure authorized deduplication scheme for cloud data based on blockchain,** *Information Processing & Management*, vol. 58, no. 3, p. 102510, 2021.
7. W. Ahmad, A. Rasool, A. R. Javed, T. Baker, and Z. Jalil, **Cyber security in iot-based cloud computing: A comprehensive survey,** *Electronics*, vol. 11, no. 1, p. 16, 2022.
8. M. Bellare, S. Keelveedhi, and T. Ristenpart, **Message-locked encryption and secure deduplication,** in *Advances in Cryptology–EUROCRYPT 2013: 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques*, Athens, Greece, May 26-30, 2013. Proceedings 32. Springer, 2013, pp. 296–312.
9. N. Kaaniche and M. Laurent, **A secure client side deduplication scheme in cloud storage environments,** in 2014 6th *International Conference on New Technologies, Mobility and Security (NTMS). IEEE*, 2014, pp. 1–7.
10. C. Yang, J. Ren, and J. Ma, **Provable ownership of files in deduplication cloud storage,** *Security and Communication Networks*, vol. 8, no. 14, pp. 2457–2468, 2015.
11. J. Gnana Jeslin and P. Mohan Kumar, **Decentralized and privacy sensitive data de-duplication framework for convenient big data management in cloud backup systems,** *Symmetry*, vol. 14, no. 7, p. 1392, 2022.
12. V. Chouhan, S. K. Peddoju, and R. Buyya, **dualdup: A secure and reliable cloud storage framework to deduplicate the encrypted data and key,** *Journal of Information Security and Applications*, vol. 69, p. 103265, 2022.
13. S. Elkana Ebinazer, N. Savarimuthu, and S. Mary Saira Bhanu, **Eskea: enhanced symmetric key encryption algo- rithm based secure data storage in cloud networks with data deduplication,** *Wireless Personal Communications*, vol. 117, no. 4, pp. 3309–3325, 2021.
14. J. Wu, Y. Li, T. Wang, and Y. Ding, **Cpda: A confidentiality-preserving deduplication cloud storage with public cloud auditing,** *IEEE Access*, vol. 7, pp. 160 482–160 497, 2019.
15. S. PG, N. RK, V. G. Menon, M. Abbasi, M. R. Khosravi et al., **A secure data deduplication system for**

integrated cloud-edge networks, *Journal of Cloud Computing*, vol. 9, no. 1, pp. 1–12, 2020.

16. S. Almuhammadi and I. Al-Hejri, **A comparative analysis of aes common modes of operation,** in 2017 *IEEE 30th Canadian conference on electrical and computer engineering (CCECE). IEEE*, 2017, pp. 1–4.
17. S. Pathakamuri, B. R. Reddy, and A. S. Kumar, **Elliptic curve digital signature algorithm for the third party auditing,** *International Journal of Engineering and Advanced Technology (IJEAT)*, vol. 9, no. 2, pp. 33–37, 2019.
18. D. Mahto and D. K. Yadav**, Performance analysis of rsa and elliptic curve cryptography.** *Int. J. Netw. Secur.*, vol. 20, no. 4, pp. 625–635, 2018.
19. J. Hur, D. Koo, Y. Shin, and K. Kang, **Secure data deduplication with dynamic ownership management in cloud storage,** *IEEE Trans. on Knowledge and Data Engineering*, vol. 28, no. 11, pp. 3113–3125, 2016.
20. H. Yuan, X. Chen, T. Jiang, X. Zhang, Z. Yan, and Y. Xiang, **Dedupdum: Secure and scalable data deduplication with dynamic user management,** *Information Sciences*, vol. 456, pp. 159–173, 2018.