Cloud Computing Security: Assured Deletion

Sarmad Mahmood Ahmed, Baban Ahmed Mahmood
Department of Computer Science, College of Computer Science and Information Technology, Kirkuk University, Kirkuk, Iraq
E-mail: stcm22006@uokirkuk.edu.iq, babanmahmood@uokirkuk.edu.iq
*Corresponding Author

Student paper

Keywords: cloud service provider, cloud storage, assured deletion, policy revocation

Received: May 22, 2024

With the advent of cloud computing, many organizations, institutions, and individuals have chosen to store their data in the cloud as a way to compensate for limited local storage capabilities and reduce expenses. However, the process of uploading data to the cloud results in relinquishing control over it, leaving the data owner unaware of the details of its storage and location. Hence, ensuring data confidentiality and integrity has emerged as a critical concern, especially with regard to cloud employees and other potential attacks. A significant amount of investigation has been conducted on the security aspects of cloud computing, with a particular focus on permanent removals. Hence This study investigates the deployment of deletion assurance in cloud computing on two separate cloud platforms, employing strategies such as cryptographic key partitioning, cryptographic-based random writing, and link deletion within confirmed deletion scenarios. The characteristics of this approach include security safeguards like asymmetric encryption (specifically Elliptic Curve Cryptography (ECC)) and multiple hashing algorithms like hash 256 are used to fortify the security and confidentiality of data stored in the public cloud and the secrecy of files kept in the cloud is unaffected by invasions from the outside or from within an encrypted file prevents a threat from reading it and the owner may easily encrypt each file with a fresh key by managing and storing the keys using the key management center also authorized users are able to request this key thanks to secure access enforcement and In the event of assure deletion, the file is overwritten with random encryption and uploaded to the cloud, updating all copies while deleting all links and keys from the key management.

Povzetek: Študija se ukvarja z varnostjp oblačnega računalništva s poudarkom na zagotovitvi brisanja podatkov. Uporabljeni so algoritmi za kriptografsko brisanje, ECC šifriranje in delitev ključev za varno in zanesljivo odstranitev podatkov iz oblaka.

1 Introduction

Cloud computing facilitates effective and economical provision of resources to users upon request through the internet. The new operational framework of the organization promotes the utilization of the public cloud for online data retention and collaboration with external entities and institutions. Users are solely charged for the resources they utilize, resulting in diminished initial establishment and information technology infrastructure expenses. Cloud services empower individuals to stockpile and retrieve substantial volumes of data from a remote location, thereby diminishing the necessity for data maintenance and administration [1], [2], [3], [4]. With the use of commodity clusters, cloud computing has brought about a framework that makes it easier to assign data processing responsibilities to outside parties. By empowering data owners to trust third parties with their sensitive information, this approach reduces the computational and administrative costs involved in granting authorized users access to and manipulation of the data. When individuals and companies upload data to the cloud. You lose direct control over this data, so it is necessary to implement a security access policy for this data to ensure its security and integrity [1]. The purpose of secure deletion is to ensure that data is not used by unauthorized parties [5]. This goal is achieved by taking steps to prevent unauthorized access to your data after the deletion process is complete [6], [7]. Data must be encrypted before being sent to the cloud server. Therefore, most existing data deletion techniques rely on decryption key deletion to achieve the desired result of secure data deletion. Nevertheless, numerous challenges have arisen in this respect, such as the intricacy linked with key management, the protracted procedure of data destruction, and the restricted authority over data access [6]. Guaranteed deletion can protect the confidentiality of user data while also maintaining its accuracy [8]. All of these methods, meanwhile, fall short when it comes to the validation process used to evaluate the results of deleting data in

the context of cloud computing. Thus, it is necessary to investigate the application of strong procedures for safe data removal as well as validation in the context of cloud computing [9]. There exist three primary methodologies. The most straightforward and highly efficient approach to achieve data eradication is via dissociation, or elimination. Eradication by superimposition: In order to supersede the original data within the cloud by updating it, the proprietor of the data engenders a data unit haphazardly that bears identical same name, classification, and magnitude as the data that necessitates erasure. The fundamental tenet underlying eradication by cryptology, or the annihilation of the decryption key, lies in the fact that the possessor of the data enciphers the dossier prior to externalization, and subsequently obliterates the decryption key of the enciphered dossier to render the encoded text infeasible for use [10], [11], [12], [13].

1.1 Contribution

In this paper, we put forward a hypothesis for a theoretical framework and plan for the encryption of information, the administration of cryptographic keys, and the preservation of keys. Additionally, we argue for the use of two separate cloud infrastructures to house both keys and documents. Furthermore, we propose the execution of a reliable method for data removal, employing a technique of overwriting using randomized encryption. Moreover, we suggest the elimination of keys and the revocation of the association policy.

The rest of the paper is organized as follows. Section 2 presents the most recent related work. The proposed method is presented in section 3. In section 4 Implementation and thorough analysis of the proposed method are presented. Section 5 concludes the paper.

2 Literature review

Researchers have suggested many deletion schemes based on policy revocation technology Overwriting, and other techniques. Below are some of the recent related works.

Tang et al. [14] designed a system for assured file deletion by revoking the file access policy. They depend on the encryption of data files and a data key to guarantee the privacy of files Initially, each data file is associated with a single policy or multiple policies through logical connections. In order to delete the file, the owner of the data revokes the policy associated with the file, and subsequently the key manager erases the private key rendering the data key irretrievable.

Yuchuan Luo et al. [15] proposed a new scheme for assured deletion called Permutation-based Assured Deletion Scheme (PADS) for the purpose of deleting data in cloud storage. This particular system involves the cloud generating data blocks in a random manner in order to overwrite the original data and ensure its recoverability. Additionally, it allows the data owner to confirm the results of the overwrite operation, ensuring successful deletion of data in the cloud.

Tyne et al. [8] A safe and effective ordered overwrite and erase scheme (SEAD-OO) is proposed. This method provides a multi-copy storage system for cloud data and adopts an ordered coverage method. The technology is composed of four basic components: Cloud Service Provider (CSP), Key Manager, Hyperledger Fabric, and Data Owners (including Authoritative Data Owner (ADO) and General Data Owner (GDO)).). ADO encrypts the file, uploads the ciphertext to the CSP, and passes the corresponding key to key management. ADO and GDO use the Diffie-Hellman protocol to establish session keys. Data accessibility is exclusively authorized based on the attributes of the data owner.

Zakaria and Mustapha [16] proposed a system to produce and store encryption keys locally rather than by a third party which was based on the Trusted Platform Module (TPM). Secure storage and cryptographic operations are offered by TPM, a hardware device often installed on the motherboard of a laptop or computer. By using the TPM chip's cryptographic features, the data is encrypted throughout the guaranteed deletion process, and a distinct key is created for it. Following the completion of the encryption procedure, only the encrypted data remains in the cloud storage with the original data being completely wiped. Because the encryption key is safely kept inside the TPM chip, FADETPM guaranteed deletion ensures that data cannot be accessed or recovered by unauthorized parties. This is what makes the technology unique.

Zhenjie Xie and colleagues [17] introduced a novel technique for ensuring the deletion of cloud data. By incorporating the XOR operation within a block cipher, the resulting cipher data becomes highly nonwearable, which presents an appealing approach to achieving assured deletion. The non recoverability of the original data subsequent to a deletion operation can be significantly enhanced by eliminating the key and certain encrypted data through the involvement of a trusted third party (TTP).

Wang and Luo [18] introduced a classification system for secure cloud data removal that is based on cryptographic methods. They examined this classification from two perspectives: one that involves a third-party key management centre and another that does not.

Joshi and Panchal [19] conducted a study on the requirements that need to be taken into account when ensuring assured data deletion in cases where the client does not have trust in the cloud server provider. These requirements include fine-grained deletion, cloud computation, availability of services, timeliness, complete deletion, and deletion acknowledgement. Furthermore,

they presented existing techniques for ensuring assured deletion in the cloud, along with their limitations.

Table 1 highlights several strategies and mechanisms that have been utilized in the literature bringing out the most important features of each method.

Table 1: Comparison of various file assured deletion mechanisms based on common techniques

References	Assure deletion	Proof of deletion	Policy	Overwriting	KM	TTP
[8]	✓	✓	✓	✓	×	×
[9]	✓	×	×	✓	×	×
[14]	✓	✓	✓	×	✓	×
[15]	✓	✓	×	✓	✓	×
[16]	×	×	×	✓	×	×
[17]	×	×	×	×	×	✓

3 Methods and implementation

As we noted in Section 2, we propose a conjecture for a conceptual framework and blueprint for the encryption of data, the management of cryptographic keys, and the retention of said keys in a key custodian. Furthermore, we advocate for the utilization of two distinct cloud infrastructures to store both keys and files. Moreover, we put forward the implementation of a secure approach for data eradication, employing a process of overwriting utilizing randomized encryption. Additionally, we recommend the eradication of keys and the annulment of the association policy.

3.1 Preliminaries

In this section, we give four preliminaries encryption, Secure Hash Algorithm (SHA-256), Encryption Standard Algorithm (AES) and Rivest-Shamir-Adelman (RSA).

3.1.1. ECC encryption

Elliptic curve cryptography (ECC), a variant of the public key cryptosystem RSA, distinguishes itself by its enhanced adaptability and its provision of a desirable alternative for cryptographic algorithm researchers. In addition, the capacity of ECC to ensure a comparable level of security to RSA, despite using smaller key sizes, is noteworthy; for example, a 160-bit ECC key provides a level of security equivalent to that of a 1024-bit RSA key [20], [21].

3.1.2. Secure hash algorithm (SHA-256)

There are several options for cryptographic hash functions, including secure hashing algorithms. This algorithm is known as a "one-sided" cryptographic function, meaning that the original text cannot be recovered through decryption. The deterministic and consistent properties of a hash function ensure that it always produces the same hash result every time it processes a given message [22], [23].

3.1.3. Encryption standard algorithm (AES)

Known alternatively as private key cryptography, the Advanced Encryption Standard method (AES) is a cryptographic method based on symmetric cryptography principles. In this approach, the encryption and decryption operations are carried out using the same key. Several lengths of cryptographic keys are used to ensure safe data encryption. There are three different key size options: 128, 192, or 256 bits. Data is encrypted using AES in blocks of 128 bits each, functioning as a block cipher. As a result, it takes an input of the same length and outputs a 128-bit encrypted ciphertext[24].

3.1.4. Rivest-Shamir-Adleman (RSA)

the public key, which is used for encryption, is distributed to all users, while the private key, which is used for decryption, is kept secret. This is an example of an asymmetric cryptography algorithm. Usually, the keys have a length of 1024 or 2048 bits. Digital signatures and

public key encryption are two applications for this method. The difficulty of factoring in huge prime numbers, which are used to produce the keys, is the foundation of its security[25], [26], [27].

3.2 System model

The principal constituents of a secure file deletion system, as delineated in Figure 1, are derived from the FADE-ECC model. The model comprises four distinct entities, namely the data owner, key manager, cloud service provider1, and cloud service provider2

3.2.1. Data owner - cloud users (DO)

The clients consist of both corporate entities and individual users. These clients engage in the rental of cloud services that are offered by the cloud service provider. These services entail a diverse selection of offerings, such as storage spaces, resources, infrastructure, and other related services. The purpose of availing these services is to leverage their economic and administrative merits, thereby alleviating the local burden associated with the utilization of these resources. Service consumers, in turn, establish an account with the cloud providers to make use of the most suitable cloud services available.

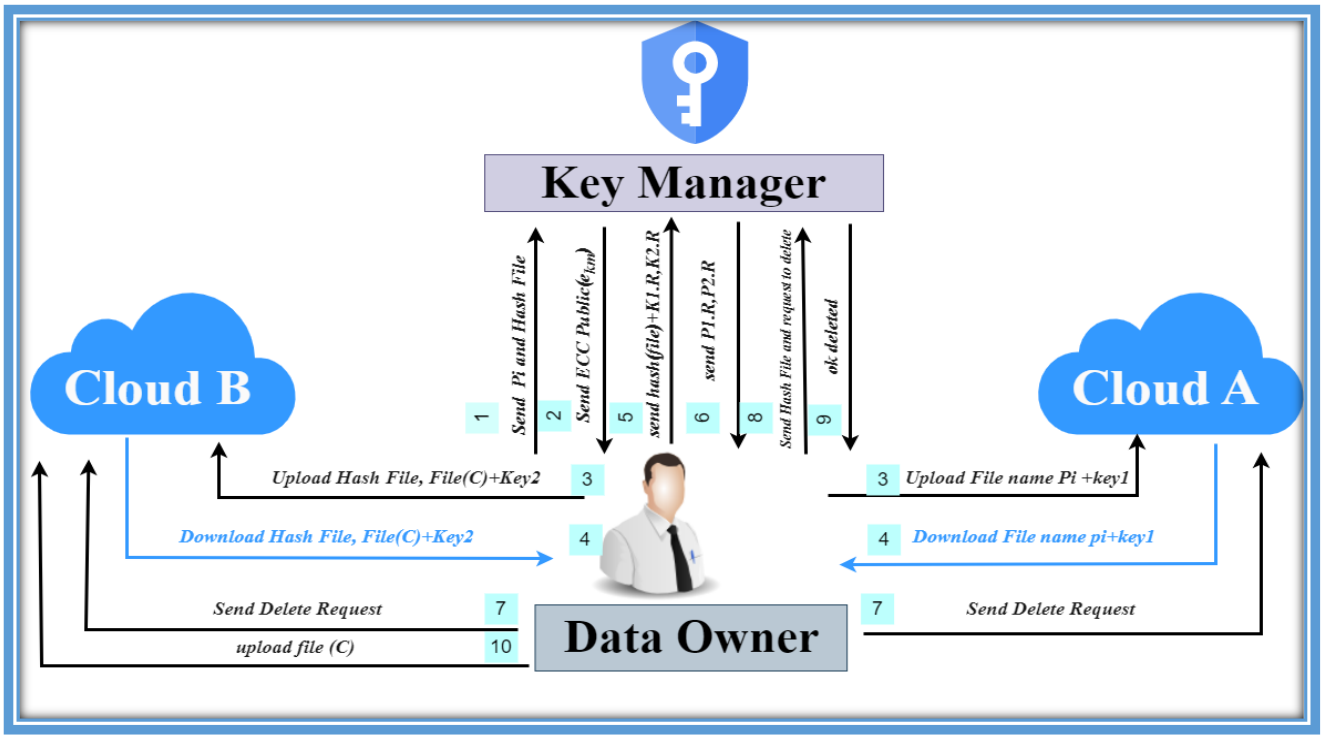

Figure 1: System model

3.2.2. Key manager (KM)

The key manager is responsible for generating and managing the ECC private-public keys (control keys). Generates control keys based on the password hash value provided by the data owner, sends the public key to the data owner for use in encryption and keeps the private key for decryption. It deletes the private key when the data owner asks to delete it.

3.2.3. Cloud service providers 1 (CSP1) and 2 (CSP2)

These are two different businesses that provide elastic computing resources that are pay-as-you-go accessible to operations throughout the network.

3.3. System description

This section shows how the various components of the architecture work together and provides a high-level overview of its data activity, including uploads, downloads, deletions, and post-deletion verification.

3.3.1. Data encryption and uploading process

The public/private key (ei, di)' creation of the ECC is performed by the data owner (DO). The file is encrypted using the public key (ei), while the private key is split into two parts (d1f, d2f) based on the data owner's policy. Additionally, a SHA-256 hash of the file name is created and sent to the key manager. Key Manager (KM), on the other hand, creates two ECC type keys (ekm, dkm), maintains the private key (dkm) with a hash of the file name, and sends the public key (ekm) to the data owner. The keys that were divided are then encrypted using the

public key of the key manager (KM) through ECC encryption. Afterwards, the encoded document and the initial encoded key are transferred to the initial cloud, while the checksum of the document is transferred with the second encoded key to the second cloud. Finally, all keys are erased, as illustrated in Figure 2

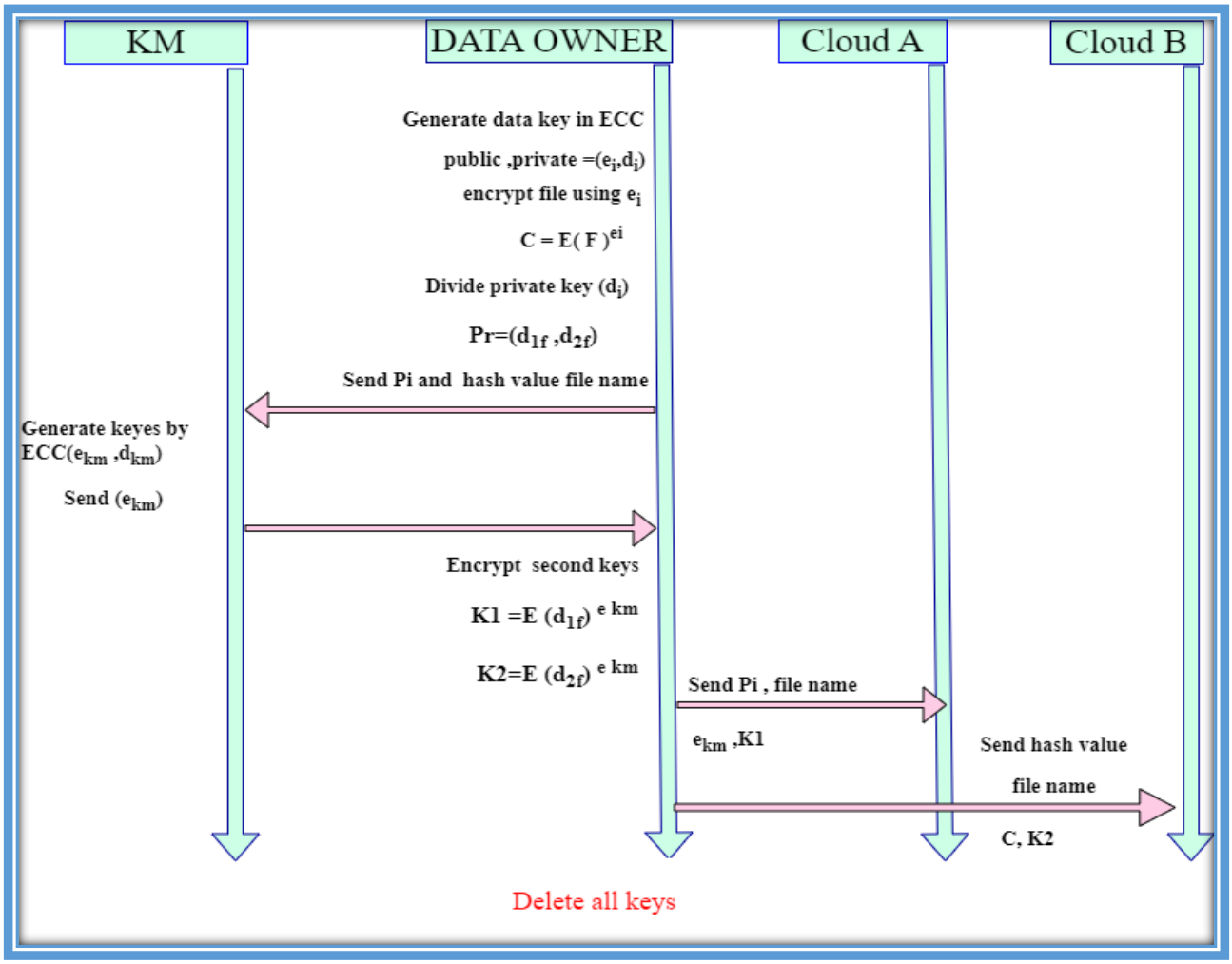

Figure 2: Data encryption and uploading process

3.3.2. File downloading and integrity checking process

The encrypted file (C) and the first encrypted key (K1) are downloaded by the data owner (DO) from the first cloud. Additionally, the data owner (DO) downloads the second encrypted key (K2) along with the file hash and the policy (pi) from the second cloud. Subsequently, a random value (R) is selected by the data owner (DO) and combined with the encrypted keys. These encrypted keys, along with the Hash-256 of the file name, are then transmitted to the key manager (KM).

Upon receipt, the key manager (KM) performs a comparison with the stored data. If a hash match is found, the sent keys are decrypted by the key manager (KM) using the ECC private key that is stored with him. The decrypted keys are then returned to the data owner. Following this, the random value (R) is removed and the keys (di) are collected based on the specified policy. Subsequently, the file is decrypted using ECC and finally opened. It is evident as depicted in the Figure 3.

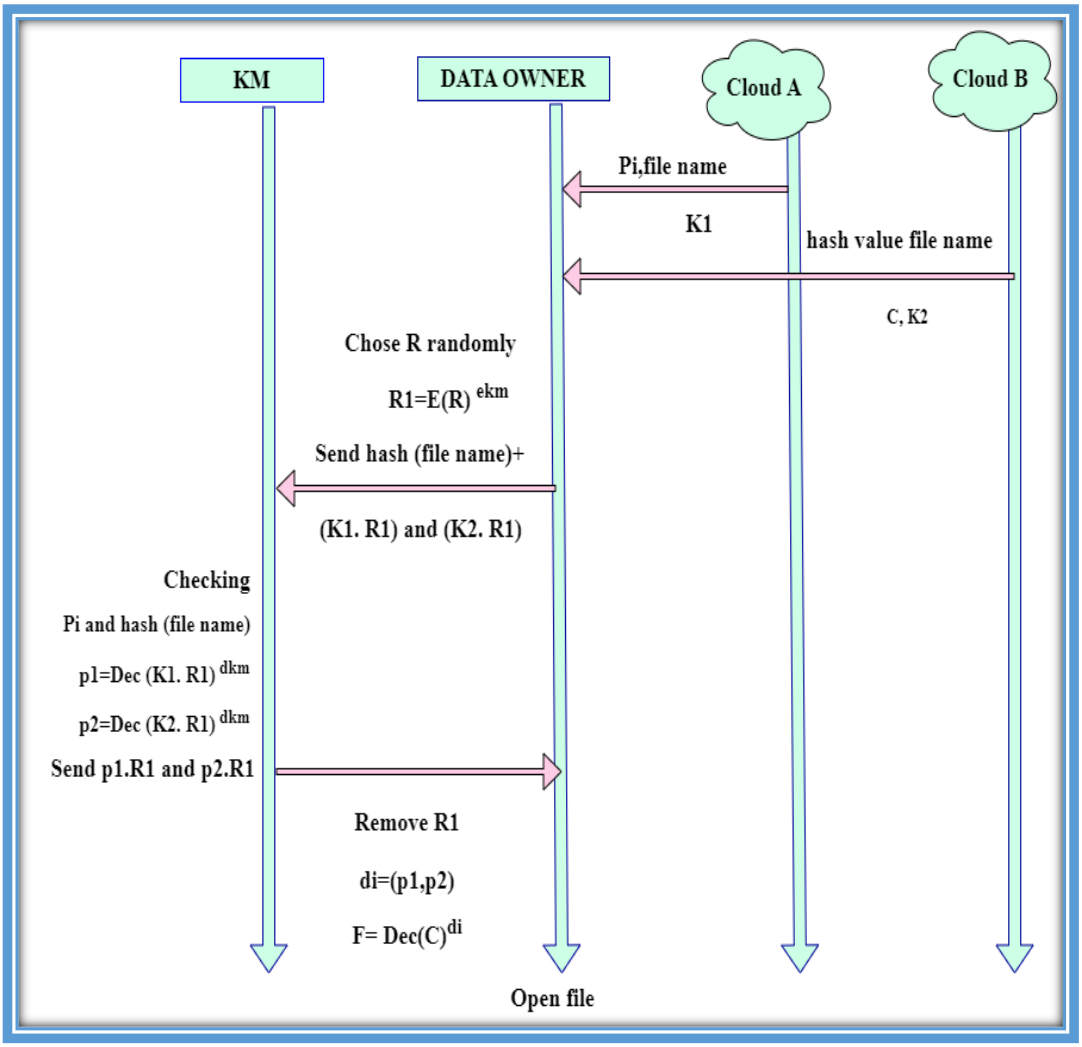

Figure 3: File downloading and integrity checking process

3.3.3 File deletion and verification process

In the process of confirmed deletion, the data owner (DO) employs a technique known as overwriting of the file. This involves encrypting the file a second time, utilizing a random public key of the ECC type without a private key. Simultaneously, the DO encrypts the file using the public key (PupR) and sends the Hash-256 of the file name to the key manager (KM). The KM is requested to erase the private key that they possess and subsequently re-uploads the file to the cloud, where the provider updates their copies. Following this, the DO asks the provider to erase the file. As a result, it can be ensured that the file stored in the cloud is rendered unusable since it is encrypted with a random encryption and lacks a first decryption key which has also been erased from key manager (KM), as portrayed in Figure 4.

3.4 Working environment

The mechanism known as FADE-ECC along with the FADE archetype were effectively implemented and executed on our Personal Computer as well as on the Firebase cloud. The process of execution entailed the utilization of a Python program in conjunction with a PyCharm editor on a Windows 10 Pro operating system. This particular OS is a 64-bit platform equipped with an Intel(R) Core (TM) i5-6300U processor that operates at a speed of 2.40GHz. Furthermore, the system was bolstered by a 16GB Random Access Memory (RAM) to support the execution of the tasks at hand. In order to gauge the performance of the application we proposed, a series of tests were conducted at various stages using files of different sizes, ranging from 1KB to 10 MB. The duration of the application's execution was carefully measured and then juxtaposed against that of the FADE mechanism at each stage. The primary objective of this evaluation was to assess the effectiveness of the implementation of our application. The interface of our implementation application is constructed based on the Tkinter module within the Python programming language. The functionality of our application commences with the input of credentials to grant access to the client for the use of the application. This process involves the design of a login system window, for entry. These fields require the

authorized customer's user name and password for progression to the subsequent window of the application, which encompasses the stages involved in the implementation of our proposed design. Individuals lacking authorization to access the system or those without the requisite login credentials are unable to advance beyond the initial window of the simulated application. Consequently, our application serves as a barrier to prevent hackers and unauthorized users from infiltrating the system and gaining access to client data.

4 Analysis and experiment

The FADE-ECC mechanism and the FADE archetype were effectively executed on our Personal Computer and on the Firebase cloud. The execution involved the utilization of a Python program and a PyCharm editor on a Windows 10 Pro operating system, which is a 64-bit OS equipped with an Intel(R) Core (TM) i5-6300U processor operating at a velocity of 2.40GHz. Additionally, the system was supported by a 16GB Random Access Memory (RAM). To evaluate the performance of our proposed application, we conducted tests at each of the three stages using files of varying sizes, ranging from 1KB to 10 MB. The execution duration of our application was measured and compared with that of the FADE mechanism at each stage. This assessment aimed to evaluate the efficacy of our application implementation.

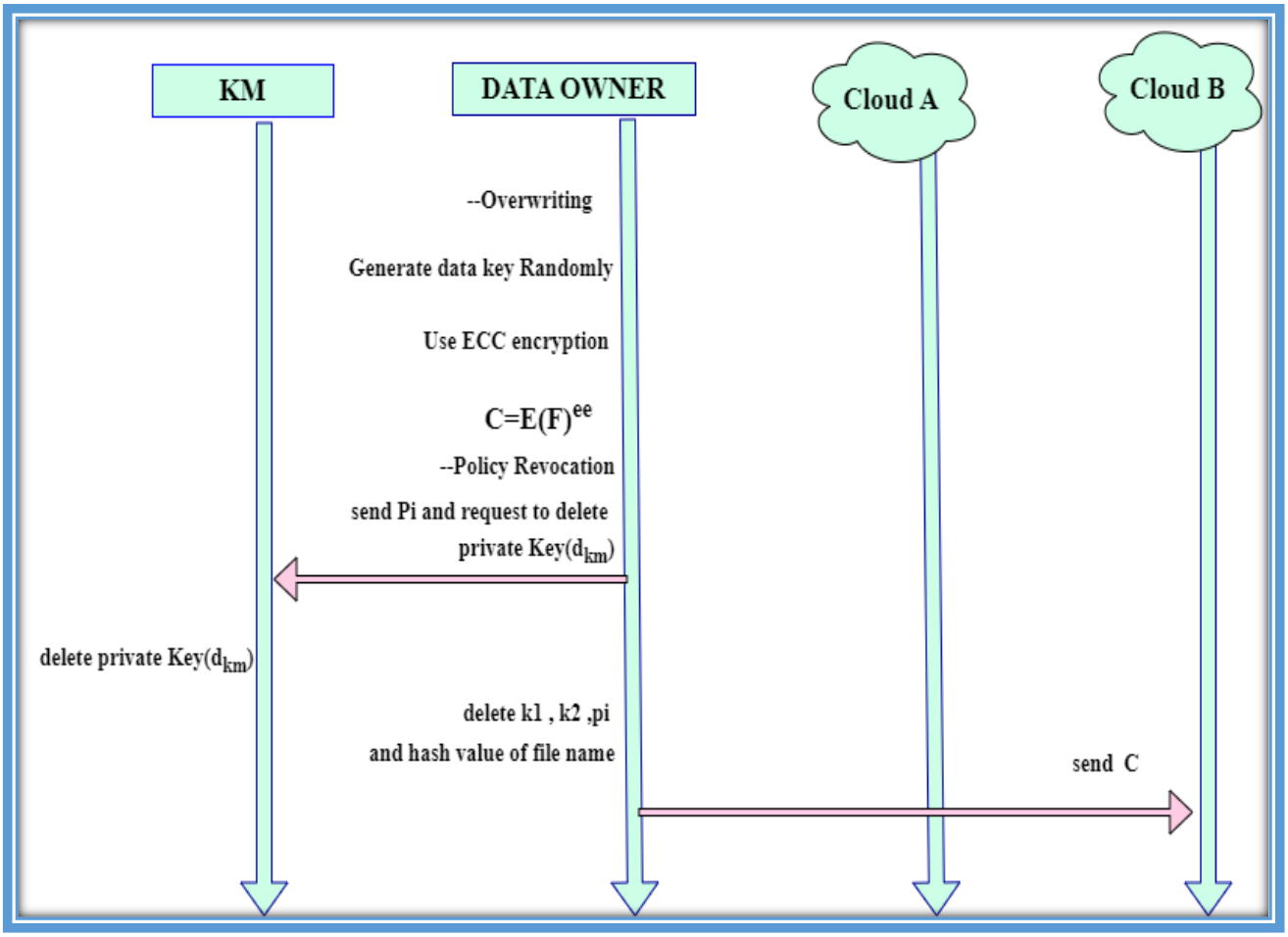

Figure 4: File deletion and verification process

4.1. File encryption and uploading stage

At this point, our system generates keys using ECC and choosing a policy, then encrypts the file, hashes the private key, and sends the hash with the file name to the key manager. The key manager also creates keys using ECC and sends the public key, then (encrypts) the keys and sends them to two clouds, while choosing the FADE model. At this stage, he chooses a policy associated with the file, then creates a key of type AES and sends the policy to the key manager to create two keys of type RSA and send the public key is sent to the file manager to encrypt the private key and send it to the cloud. The time has been calculated for the two

models as shown in Tables 1 and 2 and compared in Figure 5.

Table 2: FADE-ECC Execution Time (Seconds)

File size	File encryption	Key manger	Key encryption	File upload	Total
1KB	0.000958	0.02295	0.02318	3.48912	7.32628
10KB	0.001351	0.030574	0.02336	4.80971	11.00565
100KB	0.000997	0.028453	0.03497	4.09435	11.75391
1M	0.004986	0.030191	0.02353	5.46235	12.93462
10M	0.030574	0.025616	0.019398	18.53329	34.32785

Table 3: FADE Execution Time (Seconds)

File size	File encryption	Key manger	Key encryption	File upload	Total
1KB	0.011967	3.647901	0.003008	2.255421	10.6916
10KB	0.016942	5.212392	0.002023	2.947587	12.9932
100KB	0.021533	5.574451	0.002091	4.155981	13.4686
1M	0.024003	7.75404	0.002093	5.233113	18.3231
10M	0.08769	9.14106	0.0019814	25.28561	41.9516

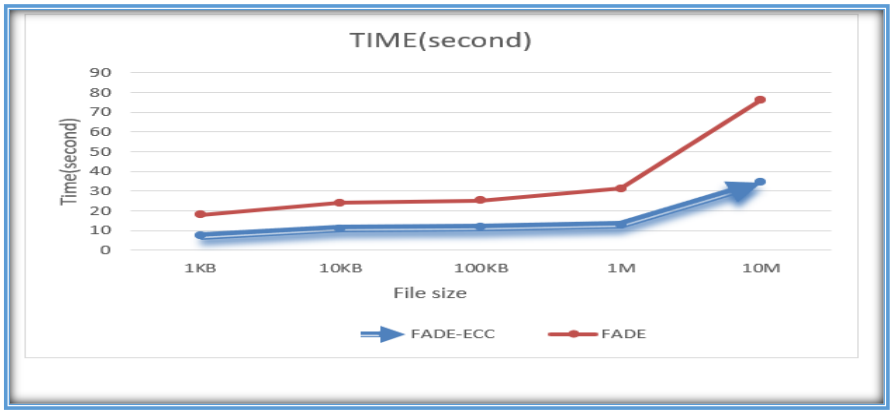

Figure 5: Comparison between the execution time of FADE and FADE-ECC

4.2 File downloading and decryption file

At this particular stage within our model, the individual who possesses the data proceeds to download both the file and the initial key from the first cloud. In addition to this, they also download the hash of the file and the second key from the second cloud. Following this, the individual in question selects a random value and combines it with the keys. Subsequently, they send this amalgamation, along with the hash of the file name, to the key manager. The key manager then proceeds to compare the hash with its own data and subsequently employs ECC to decrypt it using the provided keys. The key manager then transmits the decrypted keys back to the data owner, who then proceeds to gather the key and decrypt the file using ECC as well. In the FADE model, the data owner initially sends the key along with the random value to the key manager. The key manager then decrypts both the key and the random value using RSA, subsequently returning them to the data owner so that they may decrypt the file using AES. The time associated with both of these models is provided in Table 3 and 4, and is compared in Figure 6.

Table 4: FADE-ECC execution time (seconds)

File size	KM+ decryption keys	File decryption	Total
1KB	0.0132588	0.017235	0.0209423
10KB	0.0139636	0.017953	0.0329111
100KB	0.0150213	0.044862	0.0448623
1M	0.0130618	0.054103	0.0492492
10M	0.0120723	0.071856	0.0958753

Table 5: FADE execution time (seconds)

File size	KM+ decryption keys	File decryption	Total
1KB	0.019984	0.0301421	0.0321423
10KB	0.026381	0.0404043	0.0534001
100KB	0.029749	0.0571117	0.064261
1M	0.027643	0.0625601	0.077583
10M	0.023937	0.0940403	0.126302

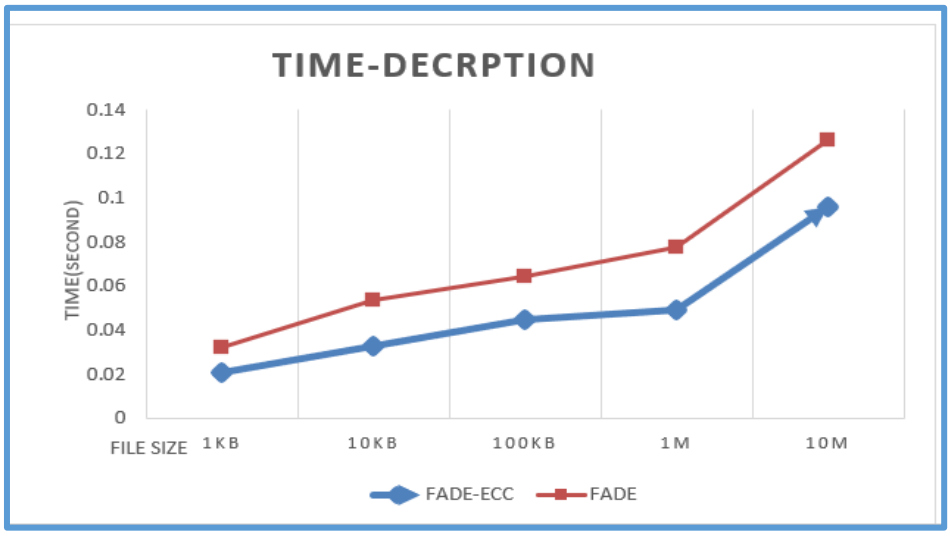

Figure 6: Comparison between the execution time of FADE and FADE-ECC

4.3 File deletion and verification phase

In our proposed model, this particular stage is characterized by the process of overwriting the file with a second layer of encryption using ECC. Additionally, this stage involves the removal of the initial encryption keys and the associated policy. In contrast, the FADE model entails the complete removal of the file from the cloud, deletion of the keys from the key manager, and elimination of the associated policy. This temporal progression of events is highlighted in table, specifically

in the time frame of the two models, as depicted in Figure 7.

Table 7: Time of assure deleting FADE-ECC

File size	Key manger	Total
1KB	0.159873	5.967233
10KB	0.159855	11.57982
100KB	0.169753	16.68525
1M	0.168642	22.76039
10M	0.292135	51.80828

Table 8: Time of assure deleting FADE

File size	Key manger	Total
1KB	0.001999	9.391458
10KB	0.290616	16.231414
100KB	0.290593	16.542681
1M	0.265314	16.563211
10M	0.385641	18.749572

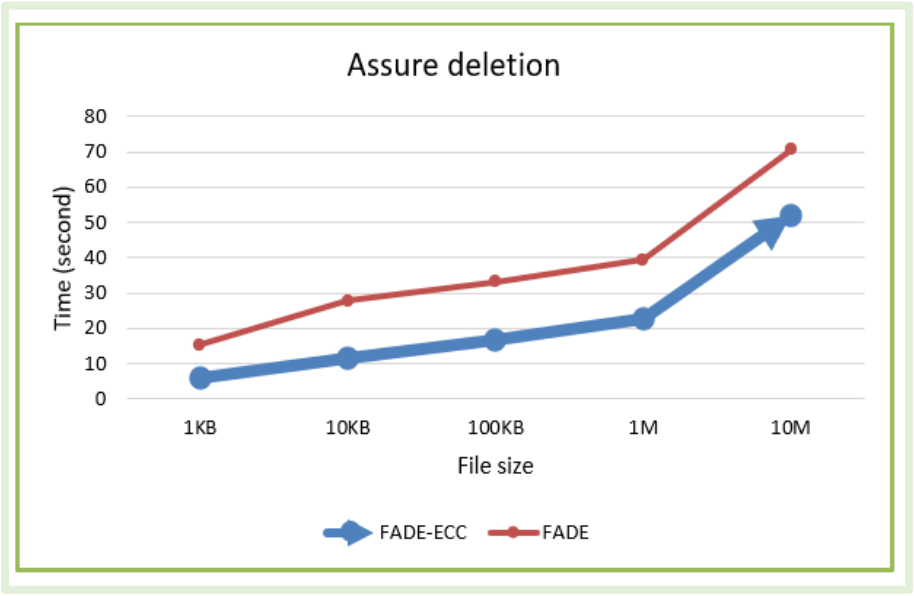

Figure 7: Comparison between the execution time of FADE and FADE-ECC

By implementing both our system and the FADE system on the same computer, the results we obtained showed the efficiency of our model in the encoding and decoding phases in terms of time. This is achieved through the use of ECC-type encryption, which features shorter keys and stronger security measures for both the data owner and the key manager. In contrast, FADE used AES encryption and RSA encryption for the key manager. From a security perspective, our model excelled in several aspects. we used two distinct clouds and used a file manager policy to split and distribute the keys. This ensures that sensitive files can only be accessed if all four parties cooperate, thus ensuring the integrity of our files even in the event of collusion between the service providers and the file manager, or in the event of an attack by an external party. Moreover, in the confirmed erasure phase, we used two confirmed erasure methods. The first method involved overwriting the file with random encryption and then uploading it to the cloud which would update all its copies. The second method involved all links

and keys being deleted from the key manager. Thus, our approach provides enhanced security for sensitive data, unlike FADE, which relies on deleting links and keys within the file manager (KM).

5 Discussion

All the systems that have been put forward in previous years have exhibited a myriad of robust features and characteristics within their mechanisms aimed at the preservation of confidential data. However, there are certain aspects that may be deemed as weaknesses or areas that require improvement, such as the necessity to avoid certain practices and to adopt more potent or cutting-edge techniques in the realm of system design and operation. An exemplary model that served as a point of comparison with our own proposed system is FADE, which comprises three primary components: the data user, the file manager, and the cloud service provider. This system navigates through a sequence of three distinct stages: the initial phase involves the encryption of files followed by their upload to the cloud, then comes the stage of downloading files and subsequent decryption, and finally, the phase of confirmed scanning. The encryption of files in this system is facilitated by the application of AES encryption technology, while the encryption of the key is accomplished using RSA technology, in addition to the utilization of policies in conjunction with files. Contrarily, our proposed system incorporates four essential elements: the data owner, the file manager, cloud provider A, and cloud provider B, all of which are intricately interconnected in the context of the research endeavor. The indispensability of these elements is underscored by the fact that the research work would not be deemed complete in their absence. Similarly, our system progresses through three key stages, namely the encryption phase and subsequent uploading of files, the phase of downloading files from the cloud, decryption, and the stage of confirmed scanning. In the encryption of files and keys within our system, we opted for the utilization of ECC encryption technology owing to its reputation as the most robust form of encryption characterized by having the smallest key size and posing a formidable challenge in terms of code-breaking compared to other encryption methods. Additionally, we incorporated hashing techniques to ensure data integrity and prevent unauthorized tampering with the data. Upon an exhaustive evaluation of several systems within this domain, the efficacy of a system is contingent upon how well the encryption keys are handled and stored within the system infrastructure or the cloud environment. Notably, our proposed system stands out as one of the most robust systems due to our distinctive approach to key management and the integration of state-of-the-art technologies in the handling of sensitive files

6 Conclusion

Assured deletion has become challenging topic in the recent years. This paper introduces a suggested security system to tackle concerns in cloud computing by utilizing two separate cloud platforms, dividing encryption keys, utilizing overwriting methods, and unlinking connections through verified deletion. Hence, the proposed method fulfils both secure and integral file upload/download and assured deletion utilizing the above-mentioned techniques.

References

[1] S. Kaushik and C. Gandhi, “Capability Based Outsourced Data Access Control with Assured File Deletion and Efficient Revocation with Trust Factor in Cloud Computing,” International Journal of Cloud Applications and Computing, vol. 10, no. 1, pp. 64–84, 2020, doi: 10.4018/IJCAC.2020010105.

[2] A. Lec Ghassan Sabeeh Mahmood, “Data Security Protection in Cloud Computing by using Encryption ,” 2017. [Online]. Available: www.kujss.com

[3] A. Hameed and Ogr. Uye. O. Karan, “Cloud Data Storage Confidentiality Using Steganography and Visual Cryptography: A Review,” Journal of Education and Science, vol. 32, no. 4, pp. 60–70, Dec. 2023, doi: 10.33899/edusj.2023.140697.1370.

[4] H. Kumar, P. J. Soh, and M. A. Ismail, “Big Data Streaming Platforms: A Review,” Iraqi Journal for Computer Science and Mathematics, pp. 95–100, Apr. 2022, doi: 10.52866/ijcsm.2022.02.01.010.

[5] B. Li, Y. Fu, and K. Wang, “A Review on Cloud Data Assured Deletion,” in 2022 Global Conference on Robotics, Artificial Intelligence and Information Technology (GCRAIT), IEEE, Jul. 2022, pp. 451–457. doi: 10.1109/GCRAIT55928.2022.00101.

[6] J. Tian and Z. Wang, “Cloud data assured deletion scheme based on dynamic sliding window,” Peer Peer Netw Appl, vol. 15, no. 4, pp. 1817–1833, Jul. 2022, doi: 10.1007/s12083-022-01318-3.

[7] Y. Miao et al., “Privacy-Preserving Attribute-Based Keyword Search in Shared Multi-owner Setting,” IEEE Trans Dependable Secure Comput, vol. 18, no. 3, pp. 1080–1094, May 2021, doi: 10.1109/TDSC.2019.2897675.

[8] J. Tian and T. Zhang, “Secure and effective assured deletion scheme with orderly overwriting for cloud data,” Journal of Supercomputing, vol. 78, no. 7,

pp. 9326–9354, May 2022, doi: 10.1007/s11227-021-04297-z.

[9] J. Ma, M. Wang, J. Xiong, and Y. Hu, "CP-ABE-based secure and verifiable data deletion in cloud," Security and Communication Networks, vol. 2021, 2021, doi: 10.1155/2021/8855341.

[10] H. Ritzdorf, N. Karapanos, and S. Čapkun, "Assited deletion of releted content," in ACM International Conference Proceeding Series, Association for Computing Machinery, Dec. 2014, pp. 206–215. doi: 10.1145/2664243.2664287.

[11] K. Q. Aziz and B. A. Mahmood, "Assured data deletion in cloud computing: security analysis and requirements," Indonesian Journal of Electrical Engineering and Computer Science, vol. 28, no. 2, pp. 1174–1183, Nov. 2022, doi: 10.11591/ijeecs.v28.i2.pp1174-1183.

[12] J. Hao, J. Liu, W. Wu, F. Tang, and M. Xian, "Secure and Fine-Grained Self-Controlled Outsourced Data Deletion in Cloud-Based IoT," IEEE Internet Things J, vol. 7, no. 2, pp. 1140–1153, Feb. 2020, doi: 10.1109/JIOT.2019.2953082.

[13] S. M. Khudaier and B. A. Mahmood, "A Review of Assured Data Deletion Security Techniques in Cloud Storage," Iraqi Journal of Science, vol. 64, no. 5, pp. 2492–2511, 2023, doi: 10.24996/ijs.2023.64.5.33.

[14] Y. Tang, P. P. C. Lee, J. C. S. Lui, and R. Perlman, "FADE: Secure Overlay Cloud Storage with File Assured Deletion."

[15] Y. Luo, M. Xu, S. Fu, and D. Wang, "Enabling assured deletion in the cloud storage by overwriting," in SCC 2016 - Proceedings of the 4th ACM International Workshop on Security in Cloud Computing, Co-located with Asia CCS 2016, Association for Computing Machinery, Inc, May 2016, pp. 17–23. doi: 10.1145/2898445.2898447.

[16] Z. Igarramen and M. Hedabou, "FADETPM: Novel Approach of File Assured Deletion Based on Trusted Platform Module," 2019, pp. 49–59. doi: 10.1007/978-3-319-97719-5_4.

[17] Z. Xie, W. Fu, J. Xu, and T. Zhu, "Assured Deletion: A Scheme Based on Strong Nonseparability," J Sens, vol. 2022, 2022, doi: 10.1155/2022/9691724.

[18] G. Wang and Y. Luo, "A Review on Assured Deletion of Cloud Data Based on Cryptography," Procedia Comput Sci, vol. 187, pp. 580–585, 2021, doi: 10.1016/j.procs.2021.04.111.

[19] S. B. Joshi and S. D. Panchal, "A Survey on Assured Data Deletion in Cloud Storage," International Journal of Computer Sciences and Engineering, vol. 7, no. 6, pp. 548–553, Jun. 2019, doi: 10.26438/ijcse/v7i6.548553.

[20] S. Kumar Verma and D. Ojha, "A Discussion on Elliptic Curve Cryptography and Its Applications," 2012. [Online]. Available: www.IJCSI.org

[21] M. Ma, "Comparison between RSA and ECC," in 2021 2nd International Seminar on Artificial Intelligence, Networking and Information Technology (AINIT), IEEE, Oct. 2021, pp. 642–645. doi: 10.1109/AINIT54228.2021.00129.

[22] R. Martino and A. Cilardo, "A Configurable Implementation of the SHA-256 Hash Function," 2020, pp. 558–567. doi: 10.1007/978-3-030-33509-0_52.

[23] S. R. Prasanna and B. S. Premananda, "Performance Analysis of MD5 and SHA-256 Algorithms to Maintain Data Integrity," in 2021 International Conference on Recent Trends on Electronics, Information, Communication & Technology (RTEICT), IEEE, Aug. 2021, pp. 246–250. doi: 10.1109/RTEICT52294.2021.9573660.

[24] R. Saha, G. Geetha, G. Kumar, and T. Kim, "RK-AES: An Improved Version of AES Using a New Key Generation Process with Random Keys," Security and Communication Networks, vol. 2018, pp. 1–11, Nov. 2018, doi: 10.1155/2018/9802475.

[25] A. H. Thiziers, H. Cisse, J. T., and B. Michel, "Enhanced, Modified and Secured RSA Cryptosystem based on n Prime Numbers and Offline Storage for Medical Data Transmission via Mobile Phone," International Journal of Advanced Computer Science and Applications, vol. 10, no. 10, 2019, doi: 10.14569/IJACSA.2019.0101050.

[26] Y. Tang, P. P. C. Lee, J. C. S. Lui, and R. Perlman, "Secure Overlay Cloud Storage with Access Control and Assured Deletion," IEEE Trans Dependable Secure Comput, vol. 9, no. 6, pp. 903–916, Nov. 2012, doi: 10.1109/TDSC.2012.49.

[27] Shahla Uthman, "An Improved RSA based on Double Even Magic Square of order 32," Kirkuk University Journal /Scientific Studies (KUJSS), vol. 12, no. 1992 – 0849, 2017.