DOI: 10.24996/ijs.2023.64.5.33

ISSN: 0067-2904

A Review of Assured Data Deletion Security Techniques in Cloud Storage

Sozan Mohammed Khudaier [1]*, Baban Ahmed Mahmood[2]

[1] *Department of Computer Science, College of Computer Science and Information Technology, Kirkuk University, Kirkuk, Iraq*

[2] *Department of Networks, College of Computer Science and Information Technology, Kirkuk University, Kirkuk, Iraq*

Received: 24/3/2022 Accepted: 27/8/2022 Published: 30/5/2023

Abstract

Cloud computing is an interesting technology that allows customers to have convenient, on-demand network connectivity based on their needs with minimal maintenance and contact between cloud providers. The issue of security has arisen as a serious concern, particularly in the case of cloud computing, where data is stored and accessible via the Internet from a third-party storage system. It is critical to ensure that data is only accessible to the appropriate individuals and that it is not stored in third-party locations. Because third-party services frequently make backup copies of uploaded data for security reasons, removing the data the owner submits does not guarantee the removal of the data from the cloud. Cloud data storage has grown in popularity, and the problem of ensured deletion has been solved. Several schemes to overcome the assured deletion problem have been proposed over the last few years. The proposed solutions have addressed the scaling overhead, trusted third parties, delays, single points of failure, and other inefficiencies. Customers had the option of receiving verifiable proof of deletion from cloud service providers. This article focuses on the issue of how cloud data storage clients may be confident that the deleted data from the cloud cannot be recovered. Furthermore, it discusses the practice of secure deletion. Moreover, the paper explores currently used methods to achieve the security of assuring the deletion of data faced by cloud entities such as cloud service providers, data owners, and cloud users. After that, the paper analyzes techniques to find the pros and cons of assured deletion and the problems that were solved by these techniques. Finally, the paper identifies some future directions for the development of assured deletion of cloud storage.

Keywords: Cloud Computing, Cloud Storage, Assured And Secure Data Deletion, Provable Data Deletion.

مراجعة لتقنيات الأمان المؤكدة لحذف البيانات في التخزين السحابي

سوزان محمد خضير [1]*, بابان احمد محمود [2]

[1] قسم علوم الحاسوب, كليه علوم الحاسوب وتكنلوجيا المعلومات, جامعة كركوك ,كركوك, العراق

[2] قسم الشبكات, كليه علوم الحاسوب وتكنلوجيا المعلومات, جامعة كركوك, كركوك, العراق

*Email: stcha009@uokirkuk.edu.iq

الخلاصة

الحوسبة السحابية هي تقنية مثيرة للاهتمام تتيح للعملاء الحصول على اتصال شبكي ملائم عند الطلب بناءً على احتياجاتهم مع الحد الأدنى من الصيانة والاتصال بين موفري السحابة. برزت قضية الأمن كمصدر قلق خطير ، لا سيما في حالة الحوسبة السحابية ، حيث يتم تخزين البيانات والوصول إليها عبر الإنترنت من نظام تخزين تابع لجهة خارجية. حيث ان من المهم التأكد من أن البيانات يمكن الوصول إليها وإزالتها من تخزين الطرف الثالث من قبل الشخص المناسب. نظرًا ان من خدمات الجهات الخارجية هي ان تقوم بشكل متكرر بعمل نسخ احتياطية من البيانات التي تم تحميلها لأسباب أمنية ،لذلك فإن إزالة البيانات التي أرسلها المالك لا تضمن إزالة البيانات في السحابة لذلك مع تطوير تخزين البيانات السحابية ، تم حل مشكلة ضمان الحذف .حيث تم اقتراح العديد من الخطط للتغلب على مشكلة الحذف المؤكد خلال السنوات القليلة الماضية. عالجت الحلول المقترحة النفقات العامة للتوسيع ، والأطراف الثالثة الموثوقة ، والتأخيرات ، ونقاط الفشل الفردية ، وأوجه عدم الكفاءة الأخرى. حيث كان لدى العملاء خيار تلقي إثبات يمكن التحقق منه على الحذف من مزودي الخدمات السحابية. تركز هذه المقالة على مسألة كيف يمكن لعملاء تخزين البيانات السحابية أن يكونوا واثقين من أنه لا يمكن استرداد البيانات المحذوفة من السحابة. علاوة على ذلك ، فإنه يناقش ممارسة الحذف الآمن. بالإضافة الى ذلك ، تستكشف الورقة الطرق المستعملة حاليًا لتحقيق الأمان لضمان حذف البيانات التي تواجهها الكيانات السحابية مثل موفري الخدمات السحابية ومالكي البيانات ومستخدمي السحابة. بعد ذلك ، تحلل الورقة التقنيات لمعرفة إيجابيات وسلبيات الحذف المؤكد والمشكلات التي تم حلها بواسط هذه التقنيات أخيرًا، تحدد الورقة بعض الاتجاهات المستقبلية لتطوير الحذف المؤكد للتخزين السحابي.

1..Introduction

Cloud computing is the Internet's backbone for hosting and delivering online services. The quantity of data saved and processed on the cloud has expanded rapidly due to cloud computing's potential to provide a full set of computer resources for a low cost, including applications, storage space, and processing capabilities. Therefore, computer resources have grown more affordable, powerful, and widely available. It was predicted that in 2020 there would be a quadrupling, drawing 2.3 billion users [1]. Without a doubt, 2020 was one of the most unpredictable years in human history. However, these events confirmed the importance of technology and its tangible impact on our daily lives. This technique enables us to work from home and helps us combat the spread of the Corona-virus (COVID-19) pandemic around the world. To further incentivize cloud migration, most traditional enterprises are motivated by the pay-as-you-go approach. In terms of global Gross Domestic Product (GDP) , the digital industry accounts for between 4.5% and 15.5% [2]. Cloud computing facilitates the deep integration of the internet, big data, artificial intelligence, and the current economy, and it is at the heart of quickening the creation of the current economic system [3].

1.1. The Deployment of the Cloud Computing.

Based on the underlying infrastructure, the National Institute of Standards and Technology in the United States (NIST) categorizes four deployment types [4].

Public cloud: Enterprises may outsource their data storage needs to public cloud providers like AWS and Alibaba Cloud without having to build or manage their own infrastructure. Authorized users can only see certain parts of the data. When it comes to small and medium-sized companies, the public cloud offers advantages such as cost savings and scalability that draw them to it.

Private cloud: Businesses must establish cloud storage infrastructures in the private cloud and use expert employees to administer and maintain servers. Accordingly, using a private cloud is more secure than using a public cloud because the organization retains ownership of its data. Despite this, the price continues to rise. A large company with a huge volume of valuable and sensitive data is better suited for this storage technique [3].

The advantages of both public and private clouds can be found in **a Hybrid Cloud,** which may store expensive and sensitive data, whereas public clouds can store other data. **The community cloud** is a novel cloud storage method. The medical and financial industries have benefited greatly from the community cloud. A community cloud offers cloud services to a group of enterprises in a specific area. Generally, many organizations have multiple concerns or a need to participate in certain operations. Members of the community cloud can either build and maintain their own infrastructure or contract it out to a third party [5].
Figure 1 depicts an overview of the NIST cloud computing architecture [6], including its primary actors, various cloud computing activities, and roles.

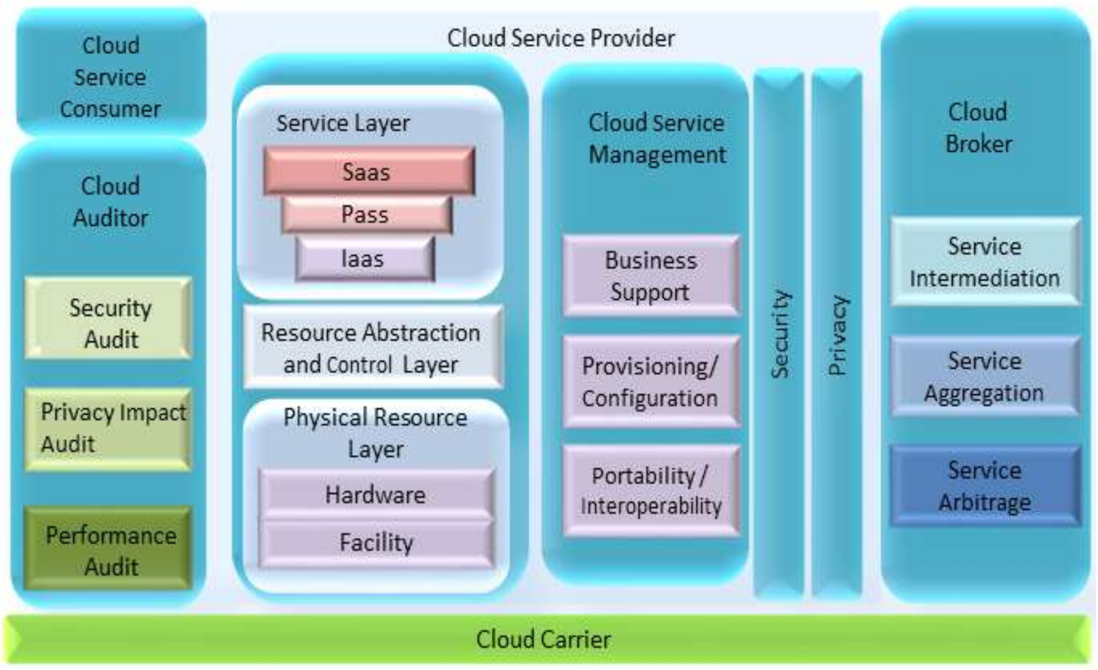

Figure 1: NIST cloud computing configuration [6].

1.2. Challenges of Security

Because third-party cloud service providers retain data, various security concerns about data privacy and accuracy may arise. Traditional and cloud storage systems are getting more trustworthy when retrieving data after a disaster or failure. Without the data owner's knowledge, cloud service providers (CSP) create numerous redundant copies of data that are made and disseminated over the cloud for reliability and availability. One particular problem is that it is difficult to remove all the copies of the data that have been made to assure reliability. Providing such assurances in the cloud remains a concern. The second part is because it stays in the Cloud Storage System CSS's physical storage device after users delete data, leading to data theft. Data that has been destroyed cannot be accessed, restored, or reconstructed by any role, including the data owner (DO), (CSS), or other users.

For certainty to delete data from the CSS, many researchers used a variety of techniques, such as schemes for data self-destruction-based DHT as presented by Geambasu et al. [7], Wang et al. [8], C. Li et al. [9] and J. Xiong et al. [10]. Data security is achieved using an overwriting technique after the deletion process, as demonstrated by Luo et al. [11], Bryan and M. Govindarasu [12], and Tian J, Zhang T. [13]. Tian et al. [14] guarantee data removal based on

Trusted Computing Protection. Several techniques use the FADE principle [15], as demonstrated by Habib et al. [16], who proposed the SFADE system. By Nusrat and others [17], considering the (SFADE) method, the (SFADE+) strategy was proposed. Zakaria. I et al. [18], developed a novel strategy, FADE-TPM. G. Wang et al. [19] believe that one of the most important types of schemes is those that rely on encryption techniques to ensure cloud storage deletion. This concept encrypts data with encryption keys before storing the encrypted data in CSS.

To achieve assured deletion, the corresponding key is deleted; consequently, the issue of data management has been transformed into one of key management. Unauthorized users will be unable to access the data once it has been encrypted. It is also dependent on the level of complexity of the encryption and the size of the key. There are two significant obstacles to overcome: guaranteeing and executing deletion in a useable manner. Guaranteed erasure aims to give customers peace of mind by guaranteeing that their data will be removed upon their request. There are no guarantees or controls over data destruction in current cloud systems. However, insufficient deletion might result in unintended disclosure [20]. When the consequences of data leakage are considerable, both cloud providers and consumers may suffer financial losses and brand damage.

1.3. Requirements for Assured Deletion

Assured deletion has certain qualities (criteria) which the technology used to ensure deletion must meet [21]:

- ➢ When removed data is no longer accessible, it is considered inaccessible.
- ➢ Response time: If possible, data should be removed from the system as soon as the owner or user requests it.
- ➢ When a user requests deletion, the system should erase all data in the system, including copies and their metadata completely.
- ➢ Users should be able to select which data should be destroyed and just that data should be erased without affecting other data kept in the system**.**
- ➢ To demonstrate that deleted data has been completely erased, you must be able to show that it has been removed from the system's storage**.**
- ➢ A service made available to consumers or to customers who have requested the deletion of their personal data should not be affected due to guaranteed deletion [22]. Any delay in service would be costly to the provider.

It is challenging to meet these objectives. Cloud companies meet these requirements through contractual agreements and privacy policies. The Cloud Service Agreement and contracts provide users with no technological proof except to assume that the supplier would delete data according to the terms of the agreement [5]. Furthermore, users cannot check the deletion because they do not have access to the infrastructure directly. It is also not obvious if current cloud deletion methods meet the needs of consumers.

1.4. Our Contributions

The research offered in this study tries to meet the following contributions:

- We present a comprehensive and detailed overview of the existing studies on techniques used to assure deletion in cloud computing.
- We analyze and evaluate the various ways of cloud data deletion. For example, our study outlines the benefits and drawbacks of previous cloud storage (guaranteed deletion) studies, which helps researchers devise new techniques.
- We present the evolution of solutions for assured deletion of cloud-stored data that have evolved over recent years, along with a discussion of the requirements, and we explain the

reasons for satisfying them. Additionally, we construct the chart indicating the technologies that can achieve some requirements of assured deletion mentioned above.

1.5. The Organization:

In the next section, we will discuss prior academic reviews on this issue. We will explain various types of assured deletion methods distinguished by some techniques. In Section 3, we analyzed techniques to determine the pros and cons of assured deletion and the problems solved by these techniques, and we investigated any of these techniques that met any of the confirmed deletion requirements. Finally, in section 4, we summarize and conclude this paper.

2.Literature review

2.1 The Prior Academic Reviews in Cloud Computing Security and Assured Data Deletion

P. Yang et al. [3] presented an in-depth look at cloud storage data security and privacy. They started by examining eight factors of data security: Data confidentiality, safeguarding the privacy of users while also ensuring data security, integrity, and availability are all important considerations while working in a large group. The authors discussed the following: identity-based encryption (IBE), attribute-based encryption (ABE), homomorphic encryption, searchable encryption, new encryption model research directions, and data encryption and protection mechanisms have been summarized. Several unanswered questions about cloud storage data security were raised in the article. In 2020, Hamed Tabrizchi et al. [6] studied cloud computing components and security and privacy issues. The authors thoroughly investigated the security risks of cloud parties, such as service providers, data owners, and cloud consumers

.

This study addressed several security issues facing cloud computing services, discussed open issues, and suggested future options. A comparative study of several strategies was proposed by Suchetha et al. [23]. The research aimed to provide data integrity as well as cloud data verification. The comparison was made according to a data integrity check, methodologies employed, performance metrics, security threats, and update modes. Angtai Li et al. [24] summarized the state of the art of different kinds of auditing schemes using ten evaluation criteria. The authors presented their findings in two parts: i) concerned with auditing a single copy of a file in a single cloud, which involves PDP-based and POR-based systems; and ii) auditing multiple copies across multiple clouds in a distributed cloud system. After reviewing the auditing literature, they discovered many flaws with existing schemes, and they finally proposed future research topics and applications.

If data is accidentally exposed, it can result in heavy financial penalties and reputational damage if it is not removed correctly. There has been no systematic examination of guaranteed deletion problems in public clouds. Kopo M. Ramokapane et al. [22] attempted to solve these issues by studying cloud-based ensured deletion needs, identifying cloud-based aspects that threaten assured deletion, and describing numerous assured deletion issues. Overall, based on this conversation, this study provided a systematization of the criteria and constraints of ensured deletion in the cloud and a solid foundation for future research into building novel solutions to assure deletion. R. Thames and A. Barsoum [25] analyzed the research articles on cloud storage deletion and how it offered an opportunity for CSPs. The paper aimed to offer guaranteed deletion as a service by introducing a risk that CSP consumers do not comprehend. Furthermore, the authors studied the progress of technologies to ensure the erasure of cloud-stored data during the previous ten years.

G. Wang et al. [19] classified specific deletion schemes-based cryptography into those with and without a third key management center (TKMC). An in-depth description of each scheme's implementation methods, access control, security, and future prospects was included in the article. TC's active defense has been exploited heavily to secure the platform's integrity and secrecy, allowing TKMC and DO to handle keys to enable assured deletion of cloud storage safely.

2.2 Assured Deletion Techniques for Cloud Storage

With rapid technological advancements, computer users have become increasingly reliant on web-based services. Personal data can be recorded, replicated, and retained without the permission or connivance of the original creator when stored and accessible on the Internet. All archival, copied, and cached data must be safe to protect the owner's privacy and integrity. If the user does not want their data saved in the cloud, delete it. Many ways of securely deleting data have been proposed in the literature. Each employs data encryption and overwriting to safely remove data (instead of erasing data from physical drives), because the volume and location of data stored in the cloud are uncertain.

2.2.1 Blockchain -Based Publicly Verifiable Data Deletion Scheme for Storage

C. Yang et al. [26] suggested a unique blockchain-based data deletion strategy that can make the deleting process more transparent. The authors utilized the Blockchain idea to ensure that, regardless of how a malicious cloud server (S) acts, everyone can check to see if the deletion was successful.

Their basic method of operation is as follows: In the first phase, the Data Owner *DO* processes the information file to generate some metadata to store it locally, and the key generation phase is performed at the *DO* side. To safeguard the privacy of the user, the file is first encrypted on the local computer before being sent to a cloud server *S*. After that, the ciphertext is retained for DO; then, the actual file is deleted by the *DO*. If the file is required by the *DO*, a query from the owner must be sent to *S* to download the ciphertext. When *DO* receives the ciphertext, it compares it to the mac generated during the encryption phase. The *DO* can decode the results to access the plaintext file if the results match. Finally, if *DO* no longer requires the file, a deletion request must be signed by the owner (DR) using his personal encryption key SK_O, where the DR is ("delete file", $\mathrm{Sig}_{\mathrm{Del}}$, TagF, td), and sent to *S* for deletion.

Upon receiving the deletion request, *S* first checks the signature's authenticity using its public key PK_O. After that, he looks at the time of deletion (td). If the td and the digital signature SigDel are correct, then this request is legitimate. Therefore, *S* will overwrite the physical disk with random data in order to erase the associated ciphertext to assure that the deleted data can never be recovered. Instead, *DO* receives an error message when trying to delete a file from the cloud server. After the file *f* has been deleted, to verify the destruction of data, a timestamp server and cloud server produce evidence for the Data Owner. *S* calculates proofi which is equal to (" delete file," $\mathrm{Sig}_{\mathrm{Del}}$, Sigs, td), where *Sigs* is $\mathrm{Sig}_{\mathrm{SKs}}$ ("delete file," Tagf, td), and they suppose $1 \le i \le m$ (Number of files in system (m) removed, when the period is over). *S* then delivers evidence to data owners so that each *DO* receives proof of the erased file. Furthermore, using the proofs' periodicity, S constructs a Merkle hash tree (MHT) and all the *MHT* is linked together by a hash chain. This tree is then announced, and the root node *rootj* is sent to the timestamp server *TS*. When *TS* receives the root node *rootj*, it first verifies that it is valid. If all the proofs proofi $(1 \le I \le m)$ are accurate, A reliable timestamp *tsj* is generated using *TS*, during this time *tj*, *S* and all data owners are informed that *tj* is the end of the current interval and *tsj* is declared to them. Otherwise, *TS* will notify *S* of an error. Using the previous hash value, the Merkle root, and *tsj* generated by the timestamp server, the cloud server S calculates a new hash value *hj* of the hash chain. *S* then declares the hash chain. Finally, Final proof τ can be provided to the data owner, equal to (proofi, rootj, hj) .If *S* does not erase the file in a truthful manner, the deletion outcome will be tracked

.

2.2.2 Assure Deletion Supporting Dynamic Insertion for Outsourced Data In Cloud Computing

Y. Liu et al. [27] used the Merkle sum hash tree (MSHT) concept to develop a cloud data erasing method that can be independently verified by the public, where *MSHT* is an extension of the *MHT* (see Table 1). The proposed scheme's primary processes are explained below.

- Set up a public/private key pair (pko, Sko) for the *DO* and one for the cloud server *S* using the Elliptic Curve Digital Signature Algorithm (ECDSA). A unique filename *nf* is then chosen by the *DO* to identify the file *F* that will be sent to the *S*. *H1 ()*, a one-way collision-resistant hash function, is selected by the *DO* and using the *Sign*, you can generate an *ECDSA* signature.
- Data Cryptography is used to safeguard personal data before uploading it to *S*. The *DO* begins by creating a secret key for the data, where *K* equivalent to *H1* (nf ‖ Sko). The key is required to encrypt the file *F* delivered to a third party, as *C* equalizes Enc_K (F), where *Enc* is the Advanced Encryption Standard (AES) method. The *DO* then breaks the ciphertext *C* into *n'* blocks and inserts (n-n') random blocks into them at random places, which are then recorded in a table *PF*. Additionally, the *DO* can get a data set *D* is ($C_{1, 1}$, ….,$C_{n, 1}$). Finally, it transmits the outsourced data set *D* and the file name *nf* to the *S*.
- The S stores the data in an (MSHT). So, the root node and its signature are returned by the data block *Ci* in the (i-th) leaf node of the MSHT Thus, the DO can assess the S's integrity.
- Data erasure: if some data blocks are no longer required, the *DO* obtains all the data blocks kept by the leaf node. Then, the *DO* creates a data erase request *DR* is (nf‖ j‖ q ‖Td ‖Sig_d), where Sig_d is signa

$$Sigd = Sign_{Sko} (nf \| j \| q \| Td) \qquad (1)$$

Where *Td* is the time stamp. Finally, the *DO* sends the *S* the data deletion request *DR*. When the *S* receives the *DR*. It verifies the signature's authenticity. If the *DR* is correct, the S deletes the data block *Cj,q* from leaf node *j* and updates the *MSHT*. If the leaf node *j* only has the data block *Cj,q* , the *MSHT* is updated. After that, a new root node *HR* is computed, and a new signature *Sig*r* equals $Sign_{Sks}$ is created. The *DO* receives the deletion evidence t equivalent to (HR, Sig* r) and the auxiliary verification information *j*.

- The *DO* can examine the data deletion outcome after getting *t* and *Øj*. In the beginning, the *DO* recalculates the root node H'_R and compares it to H_R. Then, *DO* checks the signature *Sig*r* for authenticity. If the verification fails, the *DO* rejects the data removal result; otherwise, the *DO* thinks the data deletion evidence is reliable. If the data block *Cj,q* is later found on the *S*, the *DO* may get compensated.

2.2.3 A Secure and Efficient Public Auditing System of Cloud Storage Based on BLS Signature and Automatic Blocker Protocol (AEA-Based BLS)

B. Abdulrahman et al. [28] proposed an efficient public cloud data auditing method that relies on the Boneh-Lynn-Shacham signature (BLS). This method's main idea begins with the initial phase and involves three algorithms: data protection, key generation (KeyGen), and signature generation (SigGen). The F data file can be split by the user into a collection of chunks of data *n*, (ch_1… ch_n) in the data protection algorithm. These chunks are encrypted with the Advanced Encryption Standard (AES) algorithm to achieve data confidentiality.

Consequently, they have the encrypted data file *F* is ei=(e_1….e_n). The Key Gen algorithm is also used to generate the public and secret keys on the user side. This approach uses the security parameter *k* (random value) as a secret key *sk* and generates a public key for each block. The user holds *sk* for each block and publishes the g^{sk} representing the public key. The user then uses the *SigGen* algorithm to generate a signature. This algorithm takes as inputs a public key *pk*, a secret key *SK*, a random value *a*, and encrypted data *E* that has been scrambled. The authentication identification for each chunk is represented by *Si* in the signature produced by this technique. Cloud storage is then used to store encrypted data chunks with their signatures and public keys.

Four algorithms are used in the integrity verification stage: Challenge Generation, Response, Check Proof, and Automatic Blocker Protocols (*ABP)*. The Transport Protection Protocol can be used if the user needs to check the data on the server. This is a Third Part Audit (TPA). The *TPA* must have a trusted account in the system for the user to use for auditing. This account is activated by the user in order to provide metadata information to the *TPA*. *TPA* and CSP work together to show the integrity of file F. Authentication from an untrusted *TPA* is made easier when the CSP requests that the user approve a challenge it gets from a *TPA*. The *CSP* accepts the challenge via the ABP protocol if the user agrees to the question. A response to the challenge is then provided by the *CSPs*. The algorithm's features allow them to combine many signatures for multiple blocks into a single signature (BLS).

At last, the auditor will have received the *CSP's* proof (p). It validates the returned proof and delivers success to the user. Otherwise, it gives a failure message. If there is a need to delete the block (ei), the user will submit a direct request to the server that contains the block name for the cloud to delete the specified block (ei) from the user file with its values being (Si; pki). Additionally, simultaneously, a similar request will be sent to the cloud. The block name is also included in the *TPA*. As a result, *TPA* will erase the block name (mi) and the value (vi), where vi is a random number of each block from the (ei) block.

2.2.4 Publicly Verifiable and Efficient Fine-Grained Data Deletion Scheme in Cloud Computing. (Verifiable deletion-IBF)

C. Yang et al. [29]: Depending on the Invertible Bloom Filter (IBF), a fine-grained outsourced data deletion approach will be proposed in 2020. To briefly explain the method, first the user begins to configure the keys. Here, the user must produce Elliptic Curve Digital Signature Algorithm (ECDSA) public-private key pairs (PKu, SKu) and (PKs, SKs) for both the cloud server and the final consumer. For file *F*, the user chooses a secure hash function and the tag *nf*. To safeguard sensitive data, the user performs the encryption operation according to equation 2:

$$C = Enc_K (F) \tag{2}$$

Where K equals to *H* (SKu||nf) and *Enc* is Indistinguishability under chosen-plaintext attack (IND-CPA) a secure classical symmetric encryption method. The ciphertext *C* is divided into *n'* data blocks, and random data blocks (n - n') are added to each of them by the end-user. For each data block Ci, the user randomly selects a single integer *ai*, which is used to represent the block's index in the database, where *i = 1, 2, n*. Also, the user sends *D* to the cloud server, where D is equivalent to (ai,Ci)i∈[1,n].

When *D* is received, the cloud server keeps the data blocks (C1,... Cn), adds all *ai* into the *IBF*. The cloud server then calculates the signature of server as seen in equation 3:

$$Sigs = Sign_{SKs} (IBF) \tag{3}$$

The *ECDSA* signature generating method is known as Sign. The cloud server then returns to the user's storage evidence λ is (Sigs, IBF). After that, the user will start storage checking. When a user needs to permanently remove unwanted data blocks from a physical medium, an index φ identifies the data blocks that must be deleted, and this index is created by the user. The user then generates a signature by equation 4:

$$\text{Sigd} = \text{Sign}_{\text{SKu}}\,(\text{nf}\,\|\varphi\|\text{Td}) \tag{4}$$

Td is a time stamp. Furthermore, the user produces and sends a data deletion request (*DR)* to the cloud server, where *DR* is equivalent to (nf, φ, Td, Sigd). When a *DR* is received, the cloud server validates it and verifies that the *Sigd* is legitimate. The cloud server can abort the operation if the signature Sigd is invalid; otherwise, the process moves on to the next stage of deletion.

The data blocks are removed from the cloud server $\{Ci\}i\in\varphi$ and the related indexes $\{ai\}i\in\varphi$ from the *IBF*, resulting in a new *IBF'*. Then, the server generates a new signature Sig'd is $Sign_{SKs}$ (DR||IBF') and returns to the user evidence of data deletion, τ is (IBF', DR, Sig'd). The user determines whether the *IBF'* is legitimate through signature verification and verifies *Sig'd* validity. The user aborts and reports a failure if the verification fails; otherwise, it succeeds. Next, the user checks the deletion with the help of *TPA*. The TPA receives the index from the user. The *TPA* lists all *IBF'* items and verifies whether the indexes (which have been deleted) are contained in *IBF'*; if they are, the TPA stops working and no output is produced. Alternatively, the TPA informs its users of its completion of the data deletion process.

2.2.5 Secure Overlay Cloud Storage with Access Control and Assured Deletion

Y. Tang et al. [30] suggested a secure overlay cloud storage system with fine-grained policy-based access control and file assured deletion. To achieve that, the authors proposed the *FADE* technique. In the system, each file is associated with a single policy or multiple policies. The file content is encrypted with a Data Key *DK*. The *DK* is further encrypted with the Control Key (*CK)* corresponding to the policy. The author started with the case that each file is associated with a single policy. The main operations of this method are as follows:

- File upload: the client requests the public control key (ni,ei) for policy *Pi* from the key manager (KM). If the same policy *Pi* relates to other files, it stores (ni,ei) for later use. The client then produces two random keys, *K* and *Si*, and sends them to the cloud as ($f\{K\}_{Si}$, S^{ei}_{i}), and $\{F\}K$. The client must then discard *K* and *Si*. To protect a file's integrity, the client generates a Hash-Based Message Authentication Codes (HMAC) signature for each encrypted file and stores it in the cloud with the encrypted file.
- File download: the $\{K\}_{Si}$,S^{ei}_{i}, and $\{F\}_K$ are retrieved from the cloud by the client. Before decrypting the file, the client will confirm that the *HMAC* signature is valid. The client then creates a secret random number *R*, calculates R^{ei}, and transmits S^{ei}_{i}. R^{ei} equal to $(S_iR)^{ei}$ to the key manager (KM) as a decryption request. After that, the *KM* computes and returns $((S_iR)^{ei})^{di}$ is equal to S_iR to the client. At this point, the client can remove *R* and obtain S_i, as well as decrypt $\{K\}_{Si}$ and hence $\{F\}_K$.
- Delete Files Upon Revocation of Policy: The *KM* deletes the private control key *di* and the secret prime numbers *Pi* and *qi* when a policy is canceled. As a result, they are unable to get *Si* from *Seii* and thus *K* and file *F*. They can confidently state that file *F*, which is associated with policy *Pi*, has been removed. In another case, when the file is related to multiple policies, the file upload is the same as earlier. But in file downloads, the client must be authenticated.

When a client asks the *KM* to decrypt S^{ei}_{i} R^{ei}, the *KM* encrypts its response with S_iR using attribute-based encryption (ABE) according to the file's policy. If the client meets the policy's requirements, it can decrypt the answer message and obtain S_i *R*, and policy revocation. For the *KM* to authenticate the client, they use a challenge-response system. The client informs the *K* that it needs to revoke policy *Pi*. The *KM* then produces a random number *r* as a challenge, encrypts it with *ABE*, and sends it to the client. The client can then decode *r* and transmit its hash to the *KM* as a response to the challenge if it is legitimate. Finally, the policy is revoked, and the customer is acknowledged.

2.2.6 Efficient Attribute-based Encryption with Attribute Revocation for Assured Data Deletion

AD-KP-ABE is (Key-Policy Attribute-Based Encryption) [31]. Assured deletion was presented by Xue. L. et al. in [32]. There are eight algorithms in an AD-KP-ABE scheme: The key manager is a Trusted Authority (TA) that completes the generation of Data Key DKs. When the DO needs to upload data, he encrypts it with (public-key PK, attribute set y, message M as input, outputs M's ciphertext CT), and to ensure data destruction, each DO creates a Merkle Hash Tree (MHT) from ciphertext components. Additionally, the DO constructs the root R of the MHT, signature Sigssk(R), and uploads the ciphertext and signature *Sigssk* (R) to the server. If the user needs to download files, a deterministic algorithm must be performed by taking as input the system's public key PK, an access structure that links the ciphertext CT of message M and the private key SK. The algorithm returns message M if the ciphertext's attributes match the private key's access structure. When it does not, it gives back a value of 0. If some files are no longer needed, the DO generates a delete request, *DelRequest* (γ) on the input attribute set γ, and outputs a data deletion request, DR. Then, given the master secret key MSK and a deletion request DR, TA performs (ReKeyGen). The re-encryption key rk is generated by taking the data deletion request DR and the master's secret key *MSK* as input. In this case, after executing the *ReEncrypt* (CT, rk) algorithm, new roots R' of MHT and the re-encrypted ciphertext CT' are generated by the cloud server. The Do runs the Verify (DR, AAI, R') algorithm to make sure that the data deletion process worked. The AAI is the Auxiliary Authentication Information of the node, and the algorithm outputs 1 or 0 to say whether the data deletion process worked or not.

2.2.7 Secure Data Transfer and Deletion from Counting Bloom Filter in Cloud Computing

Yang Changsong et al. [33] introduced a *CBF*-based secure data transport method capable of achieving verifiable data erasure. In the first stage, the method generates the Elliptic Curve Digital Signature Algorithm (ECDSA) public and private key pairs (PK_A, SK_A), (PK_B, SK_B), and (PK_O, SK_O) for cloud *A*, cloud *B*, and the Data Owner (DO). Then, the *DO* assigns a unique tag tag_f to the file sent to cloud *A*. Next, the *DO* selects *k* secure hash functions *g1*, *g2*, *gk* , each of which maps any integer in [1, N] to different cells in the Count Bloom Filter (CBF). To start, the *DO* encrypts the data as follows: The *DO* first generates the encryption key as shown in equation 5:

$$k = H(tag_f \,||SK_O) \tag{5}$$

And then it takes *k* to encrypt the file *C* is *Enck(F)*, where *Enc* is an *IND-CPA* safe encryption method. The DO then splits the ciphertext *C* into *n′* blocks while inserting (n -n′) random blocks into the *n′* blocks at random points, ensuring that the *CBF* is not null after data deletion and transfer. The *DO* then saves in Table *PF* these random points. The *DO* selects a distinct integer *ai* at random as the index of *Ci* and computes the hash values according to equation 6:

$$Hi = H(tagf \,||ai\, ||Ci) \tag{6}$$

Consequently, the outsourced data set can be denoted as *D* equal to $((a_1, C_1), ...,(an, Cn))$. Eventually, the *DO* transfers *D* and the file tag tag_f to cloud *A*. The *DO* then verifies the storage results and deletes the local backup. If the *DO* may switch cloud storage providers and move some data from cloud *A* to cloud *B*, following that, the *DO* may require cloud *A* to destroy some data blocks after they have been successfully transported to cloud *B* to explained in detail in the following steps:-

- The *DO* generates a signature, Sigd equal to $Sign_{SKA}$ (delete||tagf ||ϕ||Td), where the timestamp is represented by *Td* and ϕ is the index set of block indices, which will determine which data blocks to be transferred. The *DO* then generates and sends a data deletion request *Rd* which is equivalent to (delete, tagf,ϕ, Td, Sigd) to cloud *A*.
- Cloud *A* validates *Rd* after getting it. If *Rd* is invalid, cloud *A* exits and outputs fail; or else, cloud *A* overwrites the data blocks $\{(ai, Ci)\}i\in\phi$ to guarantee deleting them. Meanwhile, cloud *A* removes the $\{a_q\}\ q\in\phi$ indexes from the *CBFs* and acquires a new counting Bloom filter *CBFd*. Lastly, the cloud *A* computes a signature according to equation 7:

$$Sig_{da} = Sign\ (delete||Rd||CBFd) \tag{7}$$

And returns to the *DO* the data deletion evidence τ is equal to (Sigda, CBFd).

- The *DO* examines the Sigda after getting it. If Sigda is invalid, the *DO* exits and returns an error; else, to verify if *CBF* (aq) is 0, the *DO* selects half of the indexes from ϕ and determines if *aq* belongs to the *CBFd*. As long as the equations are valid, the *DO* knows they are correct.

2.2.8 An Efficient Scheme of Cloud Data Assured Deletion

For systems that cannot strike a balance between efficiency and fine-grained access, Zhen Li et al. [14] proposed an efficient cloud data assured deletion (ESAD) scheme. The system model of the *ESAD* scheme consists of the key management center, which is called the Attribute Key Management System (AKMS), and it is made up of a key generator and an attribute authorizer. The primary responsibility of the key generator is to maintain the system master key and the system public key. The attribute authorizer is primarily responsible for assigning a globally unique identification *ID* to each authorized user to prevent collusion attacks. The attribute authorizer keeps a list of the user's attributes, each of which is linked to the ID of the user who owns them as the private key. It uses Trusted Platform Module (TPM) security chips to safeguard the system master key and public key, as well as a list of approved users' attributes, and they communicate with other components separately. When implementing an attribute-based encryption algorithm, the simple scalar multiplication in an elliptic curve is used in this scheme instead of a complex bilinear pair, which simplifies the calculations during encryption and decryption, increases the efficiency of encryption and decryption of the data, and ensures the security of the encrypted data. The LSSS (linear secret sharing schema) [34] is used to split the encryption key, mix it with encrypted data, and upload the ciphertext to the cloud. As a result of altering an attribute, the cloud-encrypted data is no longer able to be decrypted in a smooth manner. After making an update request with a random number to the attribute authorizer in the AKMS, a data owner just needs to check the update result by utilizing *MHT* as indicated by table 1 when they want to delete data.

2.2.9 Enabling Assured Deletion in the Cloud Storage by Overwriting

Luo et al. [11] introduced a novel approach in 2016. The solution to the problem is to simultaneously shift user responsibility to the cloud while also preventing the cloud from cheating. A permutation-based hourglass function was developed, and an overwriting-based ensured deletion method called the Permutation-based Assured Deletion Scheme (PADS) was developed as a result of that work. In the straightforward scheme, file *F* is broken up into chunks by the client. (m1, m2,..., mn) and computes authenticators (1, 2,..., n) for remote integrity verification on each block. The user then uploads the data blocks *{mi}0≤i≤n*, in addition to their authenticators *{σi}0≤i≤n*, to the cloud. When the user no longer needs the file, first the user uploads a single randomly generated block, The data file *F* is split into *n* blocks (m1, m2, ..,mn). Each block is then broken down into a number of symbols *s*, (mi,1, mi,2, . ., mi,s), where *1 ≤ i ≤ n*. and *num* is the number of blocks involved in each authenticator, and a permutation key. The cloud then permutes the block's symbols to generate several other permuted blocks. Each permutation process will place a time constraint on the cloud for it to deliver a legitimate result. Secondly, a Deletion Request is generated. The user generates the public parameters, *mk, r1, and r2,* where is a randomly produced data block with *s* symbols (*1, 2,...,s), mk* is the master key for permutation operations, and *r1, r2* are two random numbers. The user then produces a key as well as two random numbers, *r3* and *r4*, to be used in the computation of authenticators. Following that, the user computes *t* authenticators (*σ1, σ2, ... , σt),* Finally, the user transmits, *mk* to the cloud as a deletion request and keeps the authenticators *{σi}1≤i≤s* local storage for deletion auditing. When the cloud receives the user's deletion request *mk*, to overwrite the original data file that is being deleted, it builds n blocks from the deletion request. An Audit of Deletion During this phase, the user uses a challenge-response protocol to audit the results of the ensured deletion in the cloud. If the cloud has assured deletion at rest, it can respond rapidly to the user's query. Performing permutation operations in the cloud will slow response time. Because of this, the user can check the results of the ensured deletion by checking the cloud's response time.

2.2.10 RITS-MHT: Relative Indexed and Time-Stamped Merkle Hash Tree-Based Data Auditing Protocol for Cloud Computing

In [35], Neenu Garg et al. described an approach based on the Relative Index and the Time-Stamped Merkle Hash Tree (RITS-MHT). There are n data blocks in a file named *"F"* that will be sent out for outsourcing (d[1], d[2],......, d[n]). Let us assume that the bilinear map (e: G × G→ Gr) represents a group *G* bilinear map, where g is the generator and p is the prime order of the group. A cryptographic hash function with the letter *H.* Numerous algorithms make up *RITS-MHT.* First Data Proprietor (DP) executes some of these algorithms. Key generation takes security parameters to output a key pair (pubkey, seckey). File tag: A number of parameters are passed into this process, including a number of block partitions (n) and the date and time the file was prepared (td), along with the file name to be outsourced. The symbol *τ* indicates this file tag. BlockSigGen takes as input *k*, a hash of file blocks *H(d[i]), dt*, and a random element *u∈G.* It produces *θ* an ordered collection of the Boneh-Lynn-Shacham signatures (*BLS)* as its output.

Another algorithm performed by *TPA* and *CSP*. *TPA* executes this algorithm to send a challenge message to Data Proprietor *DP* upon delegation of auditing. *GenProof* (F,θ,C), *CSP* runs the procedure in response to the challenge message *C* received from *TPA* (CSP creates a proof *Pf* and sends it to *TPA* for verification). File *F,* signature set *θ*, and challenge message *C* are the three inputs for this algorithm. In Verify Proof, this algorithm was conducted by *TPA*. Its inputs are *C, Pf*, and the public key g^k, and its result is either True or False depending on whether the verification succeeds or fails. Lastly, *RITS-MHT* provides support for data deletion procedure (DDP), data insertion, and data modification as dynamic procedures. Concerning our topic is the Data Deletion Procedure (DDP), a data deletion message (D, V, i, τ’). Where the field *V* defines the position of the block to be deleted. The node with hash value *H* (d[i+1]) is removed from the original tree if the V field has been set to A. A node with the hash value *H* (d[i1]) is removed from the original tree if the V field is set to B. The parent node's relative index value and hash value are also changed. Because of the regenerated *MHT, CSP* now generates a new root *R*. Finally, *CSP* provides *DP* with evidence of the deletion method as *Pf*. When *DP* receives evidence of deletion, *Pf*, the *DO*, first authenticates. After the successful authentication, it generates root by utilizing *AI(i)*, *H*(d[i]) obtained from *CSP*. The previous tree's auxiliary information is defined by AI(i). The newly produced root is then authenticated by comparing *R* to *R'* obtained from *CSP*. If TRUE, *DP* may finally assure that the desired block is deleted.

3. Discussion

This section is dedicated to analyzing and comparing the file-assured deletion mechanisms presented in Section 3. C. Yang et al. [26] investigated the security of the proposed secure data deletion procedure. Instead of using a trust third party (TTP), the authors utilized blockchain technology to solve the trust issue between *Do* and *S* (listed in Table 1). The experiments demonstrated that a data deletion scheme could satisfy correctness, completeness, and accountable traceability. Some computational effort is required in the *KeyGen* phase, which leads to costly processing. The suggested technique is more efficient in both the encrypting and decrypting phases. One signature operation and one signature verification action are required when a file is removed. The time cost is reasonable, so it is less overhead. This method is more efficient for practical applications. The proof's generation phase is very efficient because the cloud server *S* is in charge of the proof's generating phase. The *DO* does not have to suffer any costs. In the verification phase, the Merkle Hash Tree must be verified using hash function computations. Because generating a hash function is far faster than checking the authenticity of a signature, the scheme is more efficient than alternative schemas that need signature validity verification. As indicated in Table 2, this strategy might be capable of achieving some of the requirements of assuring deletion, timelessness, fine grain deletion, and availability of services. When the cloud server receives a deletion request, it will process it immediately, and the user can indicate which files he wants to delete. As long as a user requests deletion, the cloud server S will not affect any of the other user's data or other cloud services.

C. Yang et al. [27] addressed a trust issue between the data owner *DO* and the cloud server *S*. They used the *MSHT* primitive in the construction to create publicly verifiable outsourced data erasure procedures, so that data storage and dynamic insertion can both be demonstrated. In theory, this new strategy is compared to three other solutions [36, 37] to see which is the most effective. Only one system [37] did not consider data confidentiality, whereas the other three systems may have done so. All four strategies accomplished publicly verifiable deletion of outsourced data. While the proposed scheme (in [27]) adds the ability to add new data blocks to the pool of outsourced information dynamically, the other three systems are not able to do so. The suggested system prevents the *TTP* since it employs the appealing benefits of *MSHT*. Additionally, the approach outperformed the other three solutions in getting private and public verifiability. Furthermore, the authors illustrated the theoretical computational complexity. This technique requires less time than the schemes [36] and [38] in data encryption. Additional encryption is required because of the scheme [36] to generate the MAC message authentication code. The scheme [38] to generate the encryption keys requires additional hash computations. Furthermore, it is better than the other two schemes [38] and [37] in data storage. In the data deletion process, the authors discovered that the time cost of the scheme [36] is constant and that the suggested scheme is less expensive than the schemes presented in [38, 37]. There is also a limit to the number of data blocks that can be wiped. The suggested approach has a lower time overhead than other schemes. As a result, the suggested strategy for deleting outsourced data blocks is efficient. As shown in Table 2, the proposed scheme meets all the requirements for the same reasons mentioned above.

For data secrecy and integrity, B. Abdulrahman and co-workers [28] came up with a solution. The system encrypts the data before sending it to the cloud and supports batch auditing. Accordingly, with the use of this system, a *TPA* is secure, efficiently, and safely handles the needs of several potential users at once. As a result, even if users add, delete, or modify data in the cloud, the system's storage accuracy guarantee remains intact. Table 1 illustrates this point; the suggested solution uses *TPA* for auditing and improves the validation of security using an Automatic Blocker Protocol (ABP) in order to prevent illegal use of *TPA*. The authors demonstrated the system's superiority compared with its counterparts in terms of data integrity in computations by both the server and auditor. Furthermore, this approach is much more effective and safer than the previous one. Moreover, the authors formally demonstrated that data integrity may be checked for audits at a low cost due to an authentication signature. It has just 1 component, reducing the signature's storing and transfer expenses. They compared the suggested system's performance to that of related solutions. The findings demonstrated this system's minimal computational and communication overhead. We conclude that the suggested system meets most of the requirements indicated in Table 2. Because it challenges and responds to protocol, it meets proof of deletion. It achieves fine grain, timeliness, and service availability since the deletion process is performed immediately based on a user's request without being affected by other files or services of other users in cloud computing.

C. Yang et al. [29] proposed an approach that can be realized with (public and private) data deletion results reliability. Any verifier who has access to proof of the deletion of data can validate the data deletion outcome. If the data is not sent by the cloud server, then delete the order honestly. The system's performance evaluation demonstrated its efficiency compared to the existing systems. Furthermore, the computational complexity of this data deletion approach is shown to be effective no matter how many data pieces are outsourced. Moreover, the system is still very efficient in the cost of outsourcing data in terms of time and data capacity checking process. When they increase the number of data blocks to be removed, the time it takes to complete the process increases. In the proposed scheme, the computing overhead includes a signature verification procedure and (k + 3) l hash calculations are required to generate two signatures. On the other hand, the authors in [36] performed l processes for creating signatures. The scheme [39] can do n modular exponentiation computations, $l+2$ signature generation computations, and one signature verification operation. Consequently, the proposed system's time cost is the smallest. Additionally, the proposed method is more efficient in the data deletion procedure than the proposed techniques in [36, 39]. Table 1 shows that this approach uses *TTP* for auditing after deletion. Also, it uses time-stamped *TS* to give legal proof that the processes have not been modified. The main goal of this approach is to offer better security to the client and supplier. The proposed system satisfies service availability since users can delete unnecessary data blocks flexibly. In contrast, the physical medium retains all of the usable data blocks. and achieve proof of deletion by asking the *TPA* for auditing (see Table 2).

Y. Tang et al. [30] proposed the *FADE* to control who can see and delete files that are stored in today's cloud storage services. Furthermore, *FADE* uses established cryptographic approaches, such as attribute-based encryption (ABE) and a key management quorum based on threshold secret sharing. It is based on a set of cryptographic key operations selfmaintained by a quorum of key managers (KM) who are not dependent on third-party clouds to meet such security goals (see Table 1). The *KM* is responsible for all control keys (CKs). Before gaining access to the *KM*, the *DO* must first supply user credentials to the *KM*. It is important to mention that the design is based on blinded *RSA* [40]. A blind decryption strategy is deployed to keep The actual content of the data is important to keep secret to both the key manager and any attacker who intercepts the client-key management conversation. When the attacker attempts to access the original file, the *KM* is requested to decode the data key. The *ABE*-based access key protects the *KM* response. When a policy linked with a file is revoked, the file is erased. A deleted file is still encrypted with a data key; however, the control key for the revoked policy has been destroyed. Accordingly, the attacker cannot recover the original data. Even if the attacker is powerful enough to obtain the access key, it cannot decrypt the data key. It's important to note that the policy revocation processes (that were pointed to previously) don't need any cloud interactions. Through experiments, the authors found that the FADE ensures the security of outsourced data while incurring low performance and monetary costs and may be exposed to SPOF, brute-force attacks, and collusion attacks. As shown in Table 2, FADE satisfies some of the requirements of assured deletion.

The AD-KP-ABE proposed by Xue L. et al. [32] offers fine-grained access and verifiability properties. Furthermore, they proposed using Merkle Hash Trees (MHT) [41] to obtain proof of deletion from cloud storage. Moreover, the critical manager (Trusted Authority (TA)) was utilized to complete the operations of keys (see Table 1). However, as a trusted authority, *TA* cannot guarantee its credibility and security. Also, the system suffers from a collision attack (user's private key leakage issues). It is demonstrated from Table 2 that the proposed protocol achieves the design goals (requirements). We present how the authors examined the protocol's computing cost in two parts. The first part discovered that the time cost of encryption and decryption increases as the number of attributes increases. The authors put the deletion process to the test in the second part. Only one *Zp* multiplication computation is required, which reduces the total amount of time used by the user to generate a re-encryption key using *TA*. An e-cloud server is on the cloud server's side. Also, the time spent was minimal because of only exponentiation on (G_1) (G1 is the cyclic multiplicative group of prime order p). It is required to re-encrypt the previously encrypted ciphertext. To validate a guaranteed removal, the user must execute a bilinear pairing, comparing the root of the *MHT* once it is rebuilt to the one received from the cloud server. The user's time cost is unaffected by the number of attributes in the ciphertext.

Yang Changsong et al. [33] devised a novel counting Bloom filter-based strategy. The strategy addresses the challenge of securely migrating data from one cloud to another while permanently deleting the transferred data from the original cloud (see Table 1). They proposed a comprehensive comparison of this method to two prior schemes. [39], *SSE* [36]. The authors discovered that their scheme and the proposed one in [39] do not contain any *TTP*. By contrast, the scheme [36] necessitates the introduction of a *TTP*. Nevertheless, all three approaches are capable of achieving verifiable data erasure. The proposed approach outperformed the schemes in [39] and [36] for the following reasons: 1) the method [39] requires many hash calculations to produce encryption keys, and 2) the scheme [36] needs many encryption operations to generate the Message Authentication Code (*MAC)*. Accordingly, the approach is more efficient for encrypting the file. In other words, this approach just requires simple hash computations and a signature verification operation. Therefore, the time cost of generating storage-proof and the overhead are substantially lower than those of the Yang et al. scheme [39]. However, the Yang et al. technique [39] necessitates many bilinear pairing computations. The time cost of the proposed system and Yang et al.'s scheme of [39] grows in proportion to the amount of erased information, but when the deleted data blocks exceed 20, the scheme of [39] takes much longer. As a result, the authors believe their technique is more beneficial with the purpose of erasing the transferred information, than the time overhead of Hao et al. [36]'s approach is nearly constant. From the last details in section 3, it can be concluded that it provides most of the requirements for confirmed deletion (see Table 2).

Tian et al. [14] suggested The *ESAD* method is capable of effectively resisting various attacks, such as *VLAN* hopping, sniffer attacks, and preventing system failures caused by failures at a single location. Although the *AD-KP-ABE* technique [32] lets you revoke a user's access to their attributes, when compared to the ESAD, it is more susceptible to collusion and sniffer attacks. By comparing the communication costs of the *ESAD* and the *AD-KP-ABE,* they found that the communication overhead is lower than the *AD-KP-ABE* and the *ESAD* scheme's. Key generation is merely tied to the number of user attributes, whereas the AD-KP-ABE strategy is tied to the number of characteristics in the access structure. So the *ESAD* scheme has the lowest key generation time. The encryption time of the *ESAD* scheme is lower than AD-KP-ABE because *ESAD's* encryption process is simply a scalar multiplication operation on the access control matrix. The *ESAD* technique considerably reduces user decryption time when compared to *AD-KP-ABE*. The *ESAD* system takes less time to delete data than the *AD-KP-ABE* scheme. However, they detect an increase in time overhead during the verification phase in both the *ESAD* and the *AD-KP-ABE* methods. This method satisfies all the requirements as indicated in table 2, such as fine grain, inaccessibility, and service availability, since only the property that meets the access policy and individuals who haven't been removed can decrypt the ciphertext correctly. Also, it achieves timeliness and proof of deletion. Despite all of these advantages, there are some drawbacks. When the number of *ESAD* scheme users grows, so does the attributes list maintained by the attribute authorizer, which increases the time overhead of data deletion and verification, and fine-grained access will have a negative impact on performance.

The proposed method was developed by Luo et al. [11]. The PADS technique, as shown in Table 1, includes overwriting cloud data in a predictable and proven manner. According to them, they proposed this approach to solve three major flaws in these versions of FADE [30], and RERK [42]: (i) Data encryption occurs prior to data outsourcing; (ii) encryption makes cloud computation on outsourced data more difficult; and (iii) encrypted data remains on the cloud server after deletion functions. However, although the approach is creative, it requires conditions that cannot be guaranteed, such as a cloud service provider (CSP) that only keeps the most recent version of the data and data consistency across all copies when updating. And we concluded that this method often leads to malicious duplication and malicious preservation. Furthermore, in this approach, when block size and permutation time were tested, they began comparing the time and results of permuting blocks of various sizes. It is clear that the longer the permutation, the larger the block size operation required. This is because the permutation must access all of the symbols in the block at random. They assessed the time constraint for the challenge-response process. In this experiment, they consider an honest cloud and a harmful cloud. It is possible to demonstrate that the malicious cloud has a far higher time cost than the honest cloud since the malicious cloud chooses to permute on the fly. As a result, the malicious cloud must retrieve all of the symbols spread over the whole data file containing the disputed blocks. As a result, based on the cloud's response time, users can evaluate the accuracy of ensured deletion in the cloud. As indicated in Table 2, this method fulfills most of the requirements.

N.Garg et al. presented the *RITS-MHT* scheme in [35]. As illustrated in Table 1, it integrates *MHT* with a node's relative index. The cost of obtaining a data block went from O(n) in Wang's protocol to O(log n) and the time of the last change to the data also went down, which kept the data fresh. *RITS-MHT* ensured that the data that was sent to a third party was not corrupted and that the most current copy of the data was restored. This protocol allows for public data audits and easily enables dynamic data activities such as adding, removing, and altering of outsourced data at a low computational cost compared with Wang et al. [43] and Tan and Jia [44] protocols. The authors used BLS signature-based homomorphic tags in *RITS-MHT* because they are shorter in length than RSA signatures and there is no change in the metadata storage overhead at the Data Proprietor DP and TPA. At *CSP*, however, the variation is linear. We concluded that this technique satisfies most of the requirements as illustrated in Table 2 because of the following reasons: 1) it performs the deletion process based on the request of DP; 2) it deletes the target block; 3) it is not affected by another block; and 4) after deletion, *CSP* provides proof of deletion.

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

No	Scheme	Blockchain	TTP	overwriting	KM	Policy	TS	MHT	PBF
1	Blockchain-based-Publicly Verifiable Data Deletion [26]	✓	✗	✓	✗	✗	✓	✓	✗
2	MSHT [27]	✗	✗	✗	✗	✗	✓	✓	✗
3	FADE [30]	✗	✗	✗	✓	✓	✗	✗	✗
4	AD-KP-ABE [32]	✗	✗	✗	✓	✓	✗	✓	✗
5	AEA based BLS [28]	✗	✓	✗	✗	✗	✗	✗	✗
6	verifiable deletion-IBF [29]	✗	✓	✗	✗	✗	✓	✗	✓
7	CBF [33]	✗	✗	✓	✗	✗	✓	✗	✓
8	ESAD [14]	✗	✗	✗	✓	✓	✗	✓	✗
9	PADS [11]	✗	✗	✓	✗	✗	✗	✗	✗
10	RITS-MHT [35]	✗	✓	✗	✗	✗	✓	✓	✗

Table 2: Comparisons between Assured deletion mechanisms.

No	Scheme	Fine-grain	Complete deletion (Inaccessible)	Timeliness	Proof of deletion	Service availability
1	blockchain-based publicly verifiable data deletion [26]	✓	✓	✓	✓	✓
2	MSHT [27]	✓	✓	✓	✓	✓
3	AEA based BLS [28]	✓	✓	✓	✓	✓
4	verifiable deletion-IBF [29]	✓	✓	✓	✓	✓
5	FADE [30]	✓	✓	✓	✗	✓
6	AD-KP-ABE [32]	✓	✓	✓	✓	✓
7	CBF [33]	✓	✓	✓	✓	✓
8	ESAD [14]	✓	✓	✓	✓	✓
9	PADS [11]	✓	✓	✓	✓	✓
10	RITS-MHT [35]	✓	✓	✓	✓	✓

Note: ✓ Indicates that the mechanism satisfy the requirements of assured deletion.
✗ Indicates that the mechanism does not satisfy the requirements of assured deletion.

4. Conclusion

Nowadays, cloud computing is the most widely utilized computation model, and everyone is using it to provide better marketing services. Many companies have benefited from using cloud storage by decreasing their data management overhead to a considerable degree. Nevertheless, because users would lose control of their data if they outsourced it to a *CSP* for cloud storage, several strategies have emerged to prevent the theft of their personal information by other malicious users and CSP management. Apart from data security threats, removing data from cloud storage is a significant concern these days. When cloud users need to delete their data, they must ensure that the data is destroyed from all cloud storage sources and that no replica of the data exists anywhere in cloud storage. The reviewed scientific literature has indicated that assured deletion strategies have improved throughout time. Most of the techniques use cryptography before uploading the data to the cloud to deal with the issue of limited control over outsourced data. This paper selects the schemes based on some techniques to summarize each scheme's implementation methodologies. Furthermore, we have identified the recent hot topics and key publications about developing new approaches in this field. Moreover, the article has addressed the main concerns: Firstly, a thorough understanding of the most important techniques used in assured deletion, Secondly, by presenting a discussion and analyzing the previous studies (published in well-reputed journals and conferences) to determine the pros and cons of each technology in terms of time, cost, and quality. Additionally, this paper presented some critical issues faced by the previous studies. Lastly, understand the verified delete requirements and the strategies used to achieve these objectives.

References

[1] C. G. C. Index, "Forecast and methodology, 2016–2021 white paper," Updated: February, vol. 1, 2018. https://virtualization.network/Resources/Whitepapers/0b75cf2e-0c53-4891-918e-b542a5d364c5_white-paper-c11-738085.pdf

[2] "Digital Economy Report, document UN Symbol": UNCTAD/DER/2019, United Nations Conference on Trade and Development, Geneva, Switzerland, 2019.

[3] P. Yang, N. Xiong, and J. Ren, "Data security and privacy protection for cloud storage: A survey," *IEEE Access*, vol. 8, pp. 131723-131740, 2020.

[4] P. Mell and T. Grance, "The NIST definition of cloud computing," National Institute of Standards and Technology, US Dept. of Commerce, 2011.

[5] R. H. Campbell, M. Montanari, and R. Farivar, "A middleware for assured clouds," *Journal of Internet Services and Applications*, vol. 3, no. 1, pp. 87-94, 2012.

[6] 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.

[7] R. Geambasu, T. Kohno, A. A. Levy, and H. M. Levy, "Vanish: Increasing Data Privacy with Self-Destructing Data," in *USENIX security symposium*, 2009, vol. 316, p. 10.5555.

[8] G. Wang, F. Yue, and Q. Liu, "A secure self-destructing scheme for electronic data," *Journal of Computer and System Sciences*, vol. 79, no. 2, pp. 279-290, 2013.

[9] C. Li, Y. Chen, and Y. Zhou, "A data assured deletion scheme in cloud storage," *China Communications*, vol. 11, no. 4, pp. 98-110, 2014.

[10] J. Xiong, F. Li, J. Ma, X. Liu, Z. Yao, and P. S. Chen, "A full lifecycle privacy protection scheme for sensitive data in cloud computing," *Peer-to-peer Networking and Applications*, vol. 8, no. 6, pp. 1025-1037, 2015.

[11] Y. Luo, M. Xu, S. Fu, and D. Wang, "Enabling assured deletion in the cloud storage by overwriting," in Proceedings of the 4th ACM international workshop on security in cloud computing, 2016, pp. 17-23.

[12] B. Hall and M. Govindarasu, "An assured deletion technique for cloud-based IoT," in *2018 27th International Conference on Computer Communication and Networks* (ICCCN), 2018: IEEE, pp. 1-9.

[13] J. Tian and T. Zhang, "Secure and effective assured deletion scheme with orderly overwriting for cloud data," *The Journal of Supercomputing*, pp. 1-29, 2022.

[14] Y. Tian, T. Shao, and Z. Li, "An efficient scheme of cloud data assured deletion," *Mobile Networks and Applications*, vol. 26, no. 4, pp. 1597-1608, 2021.

[15] Y. Tang, P. P. Lee, J. Lui, and R. Perlman, "FADE: Secure overlay cloud storage with file assured deletion," in *International Conference on Security and Privacy in Communication Systems*, 2010: Springer, pp. 380-397.

[16] A. B. Habib, T. Khanam, and R. Palit, "Simplified file assured deletion (sfade)-a user friendly overlay approach for data security in cloud storage system," in 2013 *International Conference on Advances in Computing, Communications and Informatics* (ICACCI), 2013: IEEE, pp. 1640-1644.

[17] R. Nusrat and R. Palit, "Simplified FADE with sharing feature (SFADE+): A overlay approach for cloud storage system," in 2017 IEEE 7th Annual Computing and Communication Workshop and Conference (CCWC), 2017: IEEE, pp. 1-6.

[18] Z. Igarramen and M. Hedabou, "FADETPM: Novel approach of file assured deletion based on trusted platform module," in *International Conference of Cloud Computing Technologies and Applications*, 2017: Springer, pp. 49-59.

[19] G. Wang and Y. Luo, "A Review on Assured Deletion of Cloud Data Based on Cryptography," *Procedia Computer Science*, vol. 187, pp. 580-585, 2021.

[20] A. Rahumed, H. C. Chen, Y. Tang, P. P. Lee, and J. C. Lui, "A secure cloud backup system with assured deletion and version control," in 2011 40th *International Conference on Parallel Processing Workshops*, 2011: IEEE, pp. 160-167.

[21] K. M. Ramokapane, Usable assured deletion in the cloud, A PhD dissertation, Lancaster University (United Kingdom), 2019.

[22] K. M. Ramokapane, A. Rashid, and J. M. Such, "Assured deletion in the cloud: requirements, challenges and future directions," in *Proceedings of the 2016 ACM on Cloud Computing Security Workshop*, 2016, pp. 97-108.

[23] R. Pujar, S. S. Chaudhari, and R. Aparna, "Survey on data integrity and verification for cloud storage," in *2020 11th International Conference on Computing, Communication and Networking Technologies* (ICCCNT), 2020: IEEE, pp. 1-7.

[24] A. Li, Y. Chen, Z. Yan, X. Zhou, and S. Shimizu, "A survey on integrity auditing for data storage in the cloud: from single copy to multiple replicas," *IEEE Transactions on Big Data*, 2020.

[25] R. Thames and A. Barsoum, "Cloud Storage Assured Deletion: Considerations and Schemes," *Journal of Network and Information Security*, vol. 6, no. 2, pp. 01-04, 2018.

[26] C. Yang, X. Chen, and Y. Xiang, "Blockchain-based publicly verifiable data deletion scheme for cloud storage," *Journal of Network and Computer Applications*, vol. 103, pp. 185-193, 2018.

[27] C. Yang, Y. Liu, and X. Tao, "Assure deletion supporting dynamic insertion for outsourced data in cloud computing," *International Journal of Distributed Sensor Networks*, vol. 16, no. 9, p. 1550147720958294, 2020.

[28] B. A. Jalil, T. M. Hasan, G. S. Mahmood, and H. N. Abed, "A secure and efficient public auditing system of cloud storage based on BLS signature and automatic blocker protocol," *Journal of King Saud University-Computer and Information Sciences*, 2021.

[29] C. Yang, Y. Liu, X. Tao, and F. Zhao, "Publicly verifiable and efficient fine-grained data deletion scheme in cloud computing," *IEEE Access*, vol. 8, pp. 99393-99403, 2020.

[30] Y. Tang, P. P. Lee, J. C. Lui, and R. Perlman, "Secure overlay cloud storage with access control and assured deletion," *IEEE Transactions on dependable and secure computing*, vol. 9, no. 6, pp. 903-916, 2012.

[31] V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-based encryption for fine-grained access control of encrypted data," in *Proceedings of the 13th ACM conference on Computer and communications security*, 2006, pp. 89-98.

[32] L. Xue, Y. Yu, Y. Li, M. H. Au, X. Du, and B. Yang, "Efficient attribute-based encryption with attribute revocation for assured data deletion," *Information Sciences*, vol. 479, pp. 640-650, 2019.

[33] C. Yang, X. Tao, F. Zhao, and Y. Wang, "Secure data transfer and deletion from counting bloom filter in cloud computing," *Chinese Journal of Electronics*, vol. 29, no. 2, pp. 273-280, 2020.

[34] Q. PENG and Y.-l. TIAN, "A secret sharing scheme based on multilinear Diffie-Hellman problem," *Acta Electonica Sinica*, vol. 45, no. 1, p. 200, 2017.

[35] N. Garg and S. Bawa, "RITS-MHT: relative indexed and time stamped Merkle hash tree based data auditing protocol for cloud computing," *Journal of Network and Computer Applications*, vol. 84, pp. 1-13, 2017.

[36] F. Hao, D. Clarke, and A. F. Zorzo, "Deleting secret data with public verifiability," *IEEE Transactions on Dependable and Secure Computing*, vol. 13, no. 6, pp. 617-629, 2015.

[37] L. Xue, J. Ni, Y. Li, and J. Shen, "Provable data transfer from provable data possession and deletion in cloud storage," *Computer Standards & Interfaces*, vol. 54, pp. 46-54, 2017.

[38] C. Yang, X. Tao, F. Zhao, and Y. Wang, "A new outsourced data deletion scheme with public verifiability," in *International Conference on Wireless Algorithms, Systems, and Applications, 2019: Springer*, pp. 631-638 .

[39] C. Yang, X. Tao, and F. Zhao, "Publicly verifiable data transfer and deletion scheme for cloud storage," *International Journal of Distributed Sensor Networks*, vol. 15, no. 10, p. 1550147719878999, 2019.

[40] W. Starlings, "*Cryptography and Network Security," ed: Prentice Hall. New York*, 2006.

[41] R. C. Merkle, "A digital signature based on a conventional encryption function," in *Conference on the theory and application of cryptographic techniques, 1987: Springer*, pp. 369- 378.

[42] Z. Mo, Q. Xiao, Y. Zhou, and S. Chen, "On deletion of outsourced data in cloud computing," in *2014 IEEE 7th International Conference on Cloud Computing, 2014: IEEE*, pp. 344-351.

[43] Q. Wang, C. Wang, K. Ren, W. Lou, and J. Li, "Enabling public auditability and data dynamics for storage security in cloud computing," *IEEE transactions on parallel and distributed systems*, vol. 22, no. 5, pp. 847-859, 2010.

[44] S. Tan and Y. Jia, "NaEPASC: a novel and efficient public auditing scheme for cloud data," *Journal of Zhejiang University SCIENCE C*, vol. 15, no. 9, pp. 794-804, 2014.