

# Assured data deletion in cloud computing: security analysis and requirements

Kawa Qambar Aziz<sup>1</sup>, Baban Ahmed Mahmood<sup>2</sup>

<sup>1</sup>Department of Computer Sciences, University of Kirkuk, College of Computer Sciences and Information Technology, Kirkuk, Iraq

<sup>2</sup>Network Department, University of Kirkuk, College of Computer Sciences and Information Technology, Kirkuk, Iraq

## Article Info

### Article history:

Received May 12, 2022

Revised Jul 28, 2022

Accepted Aug 30, 2022

### Keywords:

Assured deletion

Cloud security

Cryptography

Deletion verification

Overwriting

## ABSTRACT

With the rapid development of cloud storage, more data owners store their data on the remote cloud to reduce the heavy local storage overhead. Cloud storage provides clients with a storage space that they may outsource and use on a pay-as-you-go basis. Due to data ownership separation and management, local data owners lose control over their data. Hence, all the operations over the outsourced data such as data transfer, update, and deletion, will be executed by the remote cloud server. As a result of that, various security challenges appear in terms of data privacy and integrity. In addition to data deletion that becomes an important security challenge, once a cloud user intends to delete his data, it must be sure that data is deleted from all cloud storage sources and prevent the cloud server from reserving the data maliciously for economic interests. In this paper, we present and discuss several types of research that use different technologies to solve assured deletion problems and verification the deletion result. The paper also presents a thorough analysis of the surveyed protocols in terms of fine-grained, security, performance, and requirements of remote cloud storage design.

*This is an open access article under the [CC BY-SA](https://creativecommons.org/licenses/by-sa/4.0/) license.*



## Corresponding Author:

Baban Ahmed Mahmood

Network Department, College of Computer Sciences and Information Technology, University of Kirkuk

Kirkuk, Iraq

Email: baban.mahmoodjaf@gmail.com

## 1. INTRODUCTION

Cloud computing is an online service model that provides efficient storage, programming tools, computing services, and so on, in a pay-as-you-go manner [1], [2]. Clients and organizations utilize cloud computing programs without having to install or maintain them and store their data remotely to access them from any computer with internet connectivity at any time to lower the cost of data storage and administration [3]–[5]. With the rapid growth of data due to the development of modern technology, the demand for secure and efficient cloud storage is growing [6], [7]. In spite of the wide acceptance of cloud computing as a promising service model, the inability to control remotely stored data directly caused large concerns before migrating towards the cloud in various data security issues [8], [9].

These security issues can be classified as follows: i) Data confidentiality which is about protecting data against unintentional, unlawful, or unauthorized access, disclosure, or theft [10]–[12]. ii) Data integrity which verifies that data remains unaltered during transmission and storing, and the provider doesn't lose user data due to hardware failure, human mistakes or external intrusion. Integrity is a measure of the validity of stored data [13]–[15]. iii) Access control is a security technique that regulates what resources can be viewed or used in computing environments and specifies the authorized user to access the stored data [16]–[18].

iv) Assured data deletion which means in the event of a data deletion request, the outsourced data must be permanently unreachable from anybody even the data owner [19], [20]. v) Data privacy which used to indicate the ability to control the hiding of stored data, especially when it comes to stored personal data to protect information from unauthorized parties [21]–[23].

The most important challenges that data owners face when storing their data in the cloud are provable data possession (PDP) which allows a client which has stored data at an untrusted server to verify that the server possesses the original data that is stored without retrieving it [24]–[26]. In addition to verification of assured deletion of data, the researchers used a range of different methods to verify deletion using hash functions and trees' techniques [27]–[29]. Assured deletion is very critical for the data owners and cloud service providers as the owner wants to protect his data from leakage and unauthorized access after deletion. Regarding the cloud service providers, the revelation of users' sensitive information can result in a loss of reputation and significant financial fines. Cloud service providers create several redundant copies of data and disseminate them over the cloud without the data owner's knowledge to assure reliability. It's difficult to remove all the copies, and some copies may be forgotten or refused deleted by the cloud service provider despite the data owner's request for deletion [30]. Data deletion is one of the most critical parts of managing outsourced data and a critical element of maintaining confidentiality. Users' sensitive data may be exposed inadvertently as a result of incomplete data erasure. The cloud service provider must destroy data completely and provide the owner with an assurance that the data has been safely removed with a proof of verification [31], [32].

Most popular assured data deletion methods can be described as follows: i) Deletion by overwriting which generates data randomly with the same name, type and size of the data needed to delete to overwrite the original data in the physical medium in the form of updating the original data [33], ii) Deletion by cryptography in which the data are encrypted by an encryption algorithm then the decryption key of the encrypted outsourced data file is destroyed [34], [35]. In this work, we present and discuss several types of research that uses different technologies to solve assured data deletion problems and verification of the deletion. Also, we present a crucial evaluation, enhanced by summarizing tables, of the performance, fine-grained, and safety of the discussed protocols.

The remainder of this paper is organized as follows. Section 2 introduces some of the related works. A thorough presentation of assured data deletion of different papers is presented in section 3. In section 4, we discuss and analyze the protocols presented in Section 3. Finally, a conclusion is given in section 5.

## 2. LITERATURE REVIEW

Wang and Luo [36] presented a taxonomy of assured cloud data deletion based on cryptography from the standpoint of having a third-party key management center and not having it. Joshi and Panchal [37] studied the requirements to be considered for assured data deletion when the client does not trust the cloud server provider, which includes (fine-grained deletion, cloud computation, availability of services, timeliness, complete deletion, deletion acknowledgement). In addition to presenting existing techniques to guarantee assured deletion in the cloud and its limitations. Tabrizchi and Rafsanjani [38] analyzed several components of cloud computing, as well as the current security and privacy issues that these systems confront, and provide a new categorization of contemporary security solutions in this field. Diesburg and Wang [39], described and compared current approaches for storing and deleting sensitive data in personal computing platforms. Hua *et al.* [40] presented an overview of existing secure data deletion solutions, as well as comparisons between private and public cloud storage. Fernandes *et al.* [41] surveyed the works on cloud security issues addressing several key topics, namely vulnerabilities, threats, and attacks. They also gave a thorough review of the main concepts concerning the security state of cloud environments.

## 3. PROTOCOLS

In this section, we present several techniques for assured data deletion and deletion verification used in different papers. The reviewed protocols are of several types, namely, overwriting, cryptography, and hash-based protocols. Then, based on thorough analysis, pros and cons of each protocol is provided.

### 3.1. Secure overlay storage with assured deletion and access control [42]

Yang Tang *et al.* [42] design a file assured deletion (FADE), a secure overlay cloud storage system that achieves fine-grained access control and assured deletion of data based on policy. The system is composed of three entities: First, the client which is an interface that connects the data source with the cloud. It performs an encryption and decryption operation on the cloud storage data. Second, the key managers which are responsible for key and policy management whose enforce trustworthiness through a quorum scheme, each one is an independent entity that keeps policy-based control keys for access control and assured

deletion. Last entity is the cloud server. The outsourced files are associated with a single or a Boolean combination of file access policies defined by the user for example (type of permissions of authorized users, time expiration). To encrypt file content the data owner uses a data key and 128-bit advanced encryption standard algorithm (AES) with the cipher block chaining mode. Then uses a secret key to encrypt the data key, after that, the client splits the secret key to  $N$  shares and encrypts them by using a blinded rivest-shamir-adleman (RSA) algorithm and public control key that corresponds to the policy.

All encrypted data files, the encrypted data key, and the metadata will be stored on the cloud, while key managers keep all control keys. Each policy is associated with a control key based on a 1024-bit blinded RSA algorithm. For data integrity checking, files are signed using hash-based message authentication code (HMAC). To access files, client authentication is required in the download operation. When a client requests that the key managers decrypt secret key shares, the key managers encrypt their response using attribute-based encryption (ABE) in accordance with the file policy. Therefore, the response message can be decrypted by the client if he satisfies the policy through which he can get the secret key shares which then are assembled to get the secret key. Only the users who own the data access policies accessible data files. When the policies for file access have been revoked, and files become outdated, the key manager removes the corresponding control key to the revoked policy and makes them permanently unrecoverable to everyone, including the owner. As a result, the data key and the file's encrypted content cannot be retrieved using the revoked policy's control key. Here, to divide the control key and distribute them among autonomous key managers, Shamir's ( $M, N$ ) threshold secret sharing scheme is applied. To recreate the original control key, we need to get at least from  $M$  right shares out of  $N$  key managers. This protocol is designed to separate the administration of the cryptographic keys and encrypted data, allowing encrypted data to remain on cloud storage, while cryptographic keys are preserved and controlled by a quorum of trusted key managers independently.

### 3.2. Enabling assured deletion by overwriting [33]

Luo *et al.* [33] propose a new scheme for assured deletion permutation-based assured deletion scheme (PADS) to delete data in the cloud storage. In this system, the cloud creates data blocks randomly to overwrite the original data and make it unrecoverable forever. And give the data owner ability to check the overwriting results to ensure the data deletion in the cloud. Furthermore, to confirm that the cloud has completed the deletion process based on a request for deletion at rest instead of based on a challenging demand. This paper considers that the cloud merely keeps the most recent version of the user's file, which includes many copies depending on the service level agreement (SLA), and upon updating data, all the copies will be consistent. Unlike existing solutions, users are not required to encrypt the file before outsourcing under this system. As a result, clients may leverage the powerful compute capacity to manage their data, allowing them to use outsource both computing and storage simultaneously.

At the data deletion phase, the user creates the deletion request, then sends it to the cloud, and stores the authenticator's information in local storage for checking data deletion. Users are only required to generate a random single data block and a permutation key and upload them to the cloud. When the cloud receives a user's deletion request, it permutes the block's symbols to create several new blocks to overwrite the data block to be deleted. In this system, the owners hide the overwriting procedures behind many data updating processes. For deletion audit, the system first extends the permutation-based hourglass work to impose a time range on the deletion process in the cloud to generate a valid response and to ensure that the cloud executes the overwriting process at rest. Whereas the user uses a challenge-response time protocol to check the results of the assured data deletion in the cloud. If the cloud executed the deletion operation accurately at rest, it will be able to reply to the user's challenge on time. Otherwise, it would have to execute the permutation operations on the fly, causing a considerable delay in time. As a result, the user may check the outcomes of the ensured deletion using the cloud's response time. In addition, the approach of provable data possession (PDP) is used to audit the results of overwriting. Where it allows the owner to publicly verify the integrity of cloud storage data through using sampling strategies and homomorphic authenticators and hash functions without retrieving the whole data.

### 3.3. Assured deletion with fine-grained access control for industrial applications of fog [43]

Yu *et al.* [43] propose a ciphertext-policy based assured deletion system (CPAD) that fulfils verifiable data deletion and flexible data access control by making use of ciphertext policy-attribute based encryption (CP-ABE). In this schema, the attribute authority (AA) is responsible for attributes management and creating private keys for users according to their attributes. The protocol uses the secure hash algorithm (SHA-1) and a signing algorithm. The smart object generates the data and chooses random a symmetric key, and encrypts the data using AES, a symmetric encryption algorithm. It uses the CP-ABE scheme to encrypt the data key. Then the encrypted data, as well as the ciphertext of the data key, are maintained in the nearest

fog device. The data keys ciphertext is stored on the fog device and transmits encrypted data to the cloud. For data access, a series of attributes are assigned to users, and an access structure is built by linear secret sharing scheme (LSSS). When the user's attributes fulfil the ciphertext's access requirements, he can obtain the correct plaintext. There is an attribute called (dummy attribute) that is included in each user's attributes and is required in the data access. Upon the data deletion process, a deletion request is sent from the smart object to the fog device. After the request is received, the fog device verifies the signature in the deletion request, then transmits the request to the cloud. The cloud validates deletion requests. If it is valid, the cloud deletes the ciphertext file. The fog device then modifies the encrypted data related to the dummy attribute. This indicates that the ciphertext's access structure has changed, and the ciphertext's attribute revocation ensures data deletion. As a result, the data is removed forever and users who could previously decrypt the data are no longer able to do so. The fog device then sends the response message to the smart object. When the smart object receives a response from the fog device, it checks the signature and confirms the deletion by comparing the ciphertext to ensure that the data has been modified and is no longer accessible.

### 3.4. CP-ABE-based secure and verifiable data deletion in cloud [44]

Ma *et al.* [44] design (SDVC) scheme a novel secure data deletion based on ciphertext policy-attribute based encryption (CP-ABE) and deletion verification for cloud storage data. A data deletion based on the CP-ABE applies fine-grained deletion control. Attribute association tree (AAT) is constructed in the SDVC scheme to realize quick attribute revocation depending on the concept of CP-ABE. Through node attribute updating in the AAT, new access policies can be created and data re-encrypted, therefore, encrypted data cannot be recovered. Then a rule transposition algorithm (RTA) is constructed by a cloud server provider, that creates data blocks randomly to replace the cloud data that have expired. To verify the process of data deletion results, the hash function of every data block is computed, and the value of the root node is created via merkle hash tree (MHT). A trusted authority (TA) generates the public key and the private key of the CP-ABE and distributes the public key to the data owner, and securely maintains the private key. For data upload, the owner creates a symmetric data key, constructs the access policy from a set of attributes, then splits the original file into data blocks with a size of 64MB, encrypts them by using an AES-256 algorithm and data key, then uploads them to the cloud service provider.

Next, the owner encrypts the symmetric data key by the public key of the CP-ABE and access strategy and uploads the encrypted data key to a trusted authority for secure storage. To access data, if the attributes owned by the data user satisfy the access policy, the trusted authority uses the private key to decrypt the data key and sends it to the user, who decrypts the ciphertext to get the original data. Data deletion operation starts with a deleting request sent by the owner to the TA to delete the encrypted data key, from his side the owner updates the attribute by the attribute association tree, the data related data key is re-encrypted and sent to the trusted authority by using a new access policy. Simultaneously the data owner also transmits deletion parameters and random data blocks to the cloud service provider. The RTA produces data randomly with size as the deleted data and overwrites expire out data, then it computes a validation value during the overwriting for the generation of a verifier by the MHT. The cloud service provider provides the validator to the owner within a suitable period after overwriting the deleted data, and the owner compares it to the local result to end the verification.

### 3.5. Assure deletion supporting dynamic insertion for outsourced data in cloud computing [45]

Yang *et al.* [45] designed a novel publicly verifiable outsource data deletion system, at the same time, allowing for dynamic data insertion and achieving provable data storage based on merkle sum hash tree (MSHT) is a derivative of the merkle hash tree (MHT). MSHT, unlike MHT, allows for dynamic data input and deletion. The inputs of all leaf nodes of MSHT are the data items as well as the number of these data items that are included in the same leaf node, while the inputs to the internal node are the concatenation of its two children and the number of data blocks included in its children node. The major procedures of the suggested scheme are that the data encryption key is first computed by the data owner, then an AES algorithm is used to encrypt the file, and to split the ciphertext into blocks, then a set of random blocks are inserted between these blocks at different random positions, and these random positions are recorded in a table. These random blocks cannot be erased, ensuring the MSHT doesn't be empty. The owner selects the SHA-1 hash function as the secure collision-resistant hash function and elliptic curve digital signature algorithm. The server keeps the data in an MSHT on the cloud and returns the signature of the root nodes.

In the deletion phase, the owner first downloads all of the data blocks kept by the leaf node. Then, he creates a signature and a data deletion command and sends the command of data deletion to the cloud server. The correctness of the delete command is verified by the cloud server via signature verification. If the delete command is correct, a data block is removed from the leaf node by the cloud server, and the MSHT is updated. Then, calculate a new root node, as well as a new signature. Lastly, the owner receives the deletion evidence (root node and signature) as well as the auxiliary verification information. Upon obtaining deletion

evidence and auxiliary verification information, the owner can verify the authenticity of the signature by recalculating the root node and comparing it to the received root node.

### 3.6. An efficient scheme of cloud data assured deletion [46]

Tian *et al.* [46] propose an assured deletion scheme for data in cloud storage (ESAD) for schemes that are not able to balance efficiency and fine-grained access. In this protocol, simple scalar multiplication is used in an elliptic curve to execute attribute-based encryption instead of a complex linear pair and guarantees encrypted data security. It also achieves fine-grained access to encrypted data and improves the efficiency of encryption and decryption. The system uses the LSSS to divide the key's ciphertext and merge it with the ciphertext of data to form ciphertext then send it to the cloud storage. The schema includes the Attribute Key Management System which is consist of an attribute authorizer and a key generator. The system master key and the public key are generated and managed by the key generator. While a unique identifying ID to each authorized user is assigned and helps him in partly decrypting the encrypted data, and creating the user's private key, by the attribute authorizer.

In the data deleting phase, the data owner must create a random value to replace the attribute value that has to be erased from the original attribute list. The data owner sends an attribute authorizer the key update request and then delivers the user's original attribute value and the changed attribute value. The attribute authorizer identifies the person who owns the original attribute value and replaces it with the new one. The user's private key is replaced when the attribute value is altered, but the associated attribute value's public key is not updated. As a result, the ciphertext cannot be decoded by the user, and the encrypted data is deleted. The attribute authorizer hashes each user's attribute list and then utilizes the hash value of each field in the attribute list as the leaf node of the hash tree to create a hash tree and determine the root value. The owner also keeps an authorized user's attributes list, computes the hash values for each of the user's attribute lists using the same hash function that the attributes authorizer use, and creates a tree. When the owner wants to check if the attribute authorizer is performing an update or deletion process, the new attribute list hash trees root value is sent to the owner. Then the hash value of the user set is updated by the data owner and computes the local hash tree root value using the same hash function and the auxiliary authentication information to match it with the received root value.

### 3.7. Publicly verifiable and efficient fine-grained data deletion scheme in cloud computing [47]

Yang *et al.* [47] put forward a fine-grained scheme for outsourced data deletion depended on an invertible bloom filter (IBF), that also achieves verifiability from private and public data deletion and storage results. One of the components of the system is a third party auditor (TPA) a trusted party that solves the dispute that occurs between both the user and the cloud server. First, the user generates Elliptic Curve Digital Signature Algorithm ECDSA private and public key pairs for both the cloud server and the user. After that, the user selects a secure hash function and selects a tag for the file. The user uses indistinguishability under a chosen-plaintext attack (IND-CPA) secure traditional symmetric encryption algorithm to encrypt the data and keep sensitive information secure. Then splits the ciphertext into blocks and insertion some random blocks and uploaded them to cloud computing. Using the ECDSA signature generation algorithm, the cloud server computes the outsourced data set signature. After that, returns storage evidence to the user. For checking the honesty of the cloud server in saving the outsourced data blocks. The user checks IBF validity through signature verification. Then reconstruct a new IBF using the indexes and compares it with IBF that returns by the cloud server. To permanently delete data blocks from the cloud storage, the user creates a block index that specifies the data blocks must be erased and computes the signature of the data blocks.

In addition, the user generates and sends a deletion command to the cloud server. When the cloud server receives a deletion command from the user, it verifies the integrity of the command by checking the signature. If the signature is true, the cloud server overwrites the specific data blocks for deletion, erases the relevant indexes from IBF, and creates a new IBF. Then the cloud server generates a new signature and sends the evidence of data erasure to the user. The user verifies the cloud storage data deletion outcome by evaluating the data deletion proof and verifies the validity of the new IBF by validating the new signature. Then the user checks whether the connected indexes are part of the new IBF. If the indexes belong to the new IBF which means the server failed to erase the cloud data, the user transmits the contentious index to the TPA. The TPA validates whether the index belongs to the new IBF by listing all of the elements of the new IBF. If it contains the index, the TPA reports a failure, else, the TPA reports data deletion success to the user.

### 3.8. A secure cloud backup system with assured deletion and version control [48]

Rahumed *et al.* [48] propose fade version, a system that achieves file version control and assured deletion. It makes them compatible with each other in the same design for a cloud backup system. The system achieves fine-grained deletion, allowing clients to define which versions of files in the cloud should

be removed assuredly, while the other version that has the same data as the removed versions of files would not be impacted. While, in the version control process deduplication is performed, which prevents the storing of duplicate data parts. When two objects have similar data, it is needed to only store one of them and build pointers to refer to the saved object. To see if two objects have similar content, we check each object's content through a hash function like SHA-1 and see if both objects return a similar hash result. Upon uploading data, the system adds a layered method of cryptography, and the key escrow system generates a control key based on policies that determine how every file is accessed for each version then the data is encrypted using the data keys as the first layer, and the data keys are further encrypted using the related policies' control keys as the second layer of keys.

The protocol uses symmetric algorithms like AES to encrypt the files and the data key. The encrypted data keys are kept in the version's metadata object and will be transferred to the cloud later. While control keys are stored by a key escrow system. To recover a version file object, we must first obtain the version control key through the key escrow system, followed by the decryption of the file's associated data key, then the encrypted file object. When a specific policy is revoked for file deletion, the control key associated with it is removed. If the object is only related to the cancelled policy, it will be assuredly removed; however, if the object is related to both the cancelled policy and another, not revoked policy, it will still be accessible via the active policy. For a file that appears in more than one version, the system encrypts the file using the data key first, then uses various control keys related to various versions to encrypt the data key separately. As a result, even if a control key from one version is removed, the data key and the file in another version still be recovered.

### 3.9. An assured deletion technique for cloud-based IoT [49]

Hall and Manimaran [49] suggested a layered encryption system for a hybrid approach of assured deletion that mixed secure overwriting with cryptographic security. This is to safeguard internet of things (IoT) data on the cloud from malicious preservation, malicious data duplication, and cryptanalysis attacks. The architecture of this schema consists of an IoT device user network (personal computers and owners of IoT devices) and a cloud public network with a data server. Cryptographic safeguard techniques encrypt cloud storage data and secure it from unauthorized access (both original versions and any copies produced by the cloud storage provider). Overwriting techniques, on the other hand, replace data to be deleted with random bytes, rendering it unrecoverable. In this scheme, the procedures start with the creation of files from the IoT device and their initial transmission to the private host. Next, a passphrase (Pi) is sent from the user to the private cloud host to use as file policy, the Pi uses from the cloud host side to generate asymmetric keys (KPi) which work as the control keys. After that, a symmetric key (K2) is randomly generated by the cloud host which works as the data key. Before file uploads to the cloud, the files are encrypted using (AES) algorithm and K2, and the data keys are encrypted using (RSA) algorithm and KPi.

SHA256 cryptographic hash method is used by the private cloud host to compute encrypted file hash value and store the result to verify each file integrity upon download, then keep the encrypted K2 and files ciphertext on the cloud. To access files, the user downloads the encrypted file and the data key from the cloud to the trusted private host. The private host then checks the hash of the downloaded file and data key against the previously kept hash values. If they are the same, the trustworthy host uses KPi to decrypt K2 and then uses K2 to decrypt the file. For file deletion, the user queries the metadata information (like the file type, name, and size) from the cloud server. This is a form of checking that ensures the file is kept in the cloud servers. Then a dummy file is generated by the user using the same size, name, and type as the target file and fill it with random bytes. The user sends the random false file to the cloud as an update, overwriting the cloud's most recent file copy. After overwriting is completed, the user deletes all keys linked with the removed files, where any file copies on the cloud are either ciphertext or randomized bytes.

### 3.10. A secure data self-destructing scheme in cloud computing [50]

Xiong *et al.* [50] propose a novel secure data self-destructing scheme in cloud computing called a key policy attribute-based encryption with time-specified attributes (KP-TSABE). Which mainly focuses on supporting user-defined authorized periods, providing fine-grained access control on these periods, and implementing data self-destruction once the authorization period has expired. The KP-TSABE structure includes two main entities, first is authority, which is in charge of creating, distributing and managing the whole private keys. The second is the time reference server which is responsible for accurate release time specification and providing the current time. In addition to generating system private key, master key, and public parameters, and master key are kept secretly to itself. Each data item in this system can be related to a list of attributes, and each attribute is linked to a decryption attribute time interval specification, while the private key is related to a time instant.

The sensitive data will be securely self-destructed after a user-specified expiration time and no one can decrypt it. For the time-constrained encryption, the data owner selects an attribute set for the shared

message and assigns a time interval set to it, then uses the encryption algorithm to encrypt the shared message to its ciphertext, which is associated with the time interval and attributes set. Lastly, sends the ciphertext to cloud servers. When the user wants access to the shared data through the authorization period, he must satisfy the identity authentication process, and the current time instant must be provided by the time server. If the current time instant is in the range of the time interval and the user's attribute set satisfy the access tree, then the private key is generated by the authority and sends it to the user. After receiving the private key, the user obtains the ciphertext from the cloud servers and decrypt the ciphertext to access the shared data message. Due to data self-destruction, the user will be unable to get the private key because of secure key expiration after the current time instant passes the time interval (expiration time) of the allowed period. Hence, the ciphertext cannot be decrypted in different range times, which facilitates the shared data self-destruction after expiration.

#### 4. DISSCUSSION OF THE SERVED PROTOCOLS

In FADE [42], the author uses the blinded decryption method to prevent the key manager and attacker from getting the original content of the data key if he sniffs the client's meeting with the key manager. This makes the system secure even with a semi-trusted key manager. The solution drawback is that the deletion policy only supports a limited number of Boolean expression layers, and complex decryption operations performed by key managers for data files, which makes it unsuitable for big dynamic user scenarios due to the key manager's functions in all operations of data deletion, insertion and access. Therefore, if only one key manager is used, this may cause a one point failure, because a non-trusted key manager may fail to erase the keys when the client wanted or remove them before it is requested to do so.

The PADS Scheme [33] reduces the user-side cost of communications as compared with the traditional method which is costly because the traditional method requires the client to upload the randomly created file that has the same size as the removed file and then perform several update processes to overwrite the original data in the cloud. The accuracy of this scheme is proved by the hourglass procedure, which imposes a time restriction on the cloud to the permutation-based overwriting operation. That prevents the dishonest cloud to create the challenge-response if it has not completed the data removal at rest. As the experimental results and theoretical analysis show that the scheme is efficient and secure.

The suggested scheme CPAD [43] has been proven secure, and it is resistant to collusion attacks by unauthorized users. That is, unauthorized users' collusion cannot lead to data disclosure. Because only data owners and fog devices are participating in removing cloud data and checking the deletion of this data, the protocol is practical due to the relatively low latency and real-time fog interaction.

In SDVC scheme [44], the random overwriting that is used has better efficiency than the all-zero overwriting. Low time cost and high efficiency for the encryption/decryption operation in this system because of employing AES-256 algorithm. As experimental results prove that during the encryption and decryption process, increasing the number of attributes increases the time cost.

The suggested approach in Yang *et al.* [45] is to ensure only the data owner can accurately decrypt the encrypted text, that means, ensuring the confidentiality of data when it is outsourced. The cloud server will not be able to create deletion proof effectively to persuade the owner that the data have been erased, if he does not remove the data honestly. The advantage of this protocol is its ability to achieve public verifiability without the need for any trusted third party. This system is more efficient for data insertion, data storage, encrypting and deleting the outsourced file with fewer time costs overhead.

In ESAD [46] the system prevents collusion attacks by assigning a unique ID to each user, associating it with the user's attribute and introducing the unique ID into the user's private key. When a user's attribute is cancelled, the user's usual access to other authorized attributes does not affect the due to the user revocation implemented at the specific level of the attribute. This schema has a lower communication overhead because fewer public keys are used, and the deletion process just sends attributes to be removed when the ciphertext is decrypted by the authorized user. As the number of users of the ESAD system grows, so does the attributes list kept by the attribute authorizer. This will raise the time overhead of data deletion and verification, resulting in the efficiency of deletion and verification decreasing as the number of users grows. The suggested approach in [47] is ideal for huge data deletion instances because the computational complexity of data deletion and verification operations is not based on the number of data blocks outsourced. Through experiments, it turns out that the proposed scheme is efficient in the time cost of data outsourcing, data storage checking, data deletion, and checking the data deletion result process.

In the FadeVersion system [48], the control keys are stored safely by a key escrow scheme, which increases the attacker's attack resources needs. Due to that, it requires users to manage their keys via a key escrow scheme which may cause a heavy burden on data users because the size of the keys can be very large

if fine-grained deletion is needed. This happens because fine-grained deletion assigns various control keys with any version of the file.

The cryptographic protection in [49] prevent malicious preservation of the file but lets the data to exposure cryptanalysis attacks, while overwriting methods prevent cryptanalysis attacks but do not protect data while it is in use. By using a hybrid method of the two types, data can be protected from all three above threats. The disadvantage of the hybrid method in this paper is that it uses the algorithm AES, RSA, and SHA256 together which are too expensive for IoT devices. On the user side, the communication and computation overhead is high since the user must build a data file of the same size as the data to be removed and upload it to the cloud to perform secure overwriting.

The KP-TSABE schema [50] has several properties and is superior as compared with other existing self-destructing techniques. It supports the user-defined authorization period feature, ensuring that sensitive data cannot be accessed both before and after the set release time. It doesn't get Sybil attacked because it does not use distributed hash table (DHT) network. It can apply fine-grained access control during the permission period and perform self-destruction of sensitive data once the authorization period has expired without the need for human interaction. The authorization period of the shared data is predefined flexibly by the data owner, therefore, the system limitation does not affect the authorization time or the expiry time. In Table 1 we present the Properties and methods used in deletion and verification algorithms by several of these protocols.

Table 1. Properties and methods used in deletion and verification algorithms

| Protocol     | Data encryption algorithm | Data key encryption algorithm | Verification method and function | Data deletion method            | Protocol properties                                |
|--------------|---------------------------|-------------------------------|----------------------------------|---------------------------------|--|
| FADE         | AES                       | AES                           | HMAC                             | Delete control key              | Assured data deletion, fine-grained access control |
| PADS         | Not encrypt               | /                             | approach of (PDP)                | overwriting                     | Assured data deletion                              |
| CPAD         | AES                       | CP-ABE                        | SHA-1                            | ciphertext attribute revocation | Assured data deletion, flexible access control     |
| SDVC         | AES                       | CP-ABE                        | MHT                              | delete data key and overwriting | Assured data deletion, access control              |
| Fade version | AES                       | AES                           | SHA-1                            | delete control key              | Assured data deletion, version control             |
| REF          | AES                       | RSA                           | SHA 256                          | Overwriting and delete keys     | Assured data deletion                              |

## 5. CONCLUSION

Assured deletion schemes have evolved over the last ten years as evidenced by the reviewed academic literature. Assured data deletion is a significant obstacle to adopting public cloud services. In this paper, we have stated the importance of assured data deletion in cloud storage. We have surveyed the existing solutions for data deletion and outlined the advantage and limitations of different schema models. We conclude that the security of the cloud storage file is based on the complexity of the encryption algorithm applied and the size key, and the use of hybrid deletion methods gives more assurance of data deletion. We note many researchers have proposed new solutions for assured data deletion in cloud storage providing an essential path to a wider community of researchers to extend this work to provide assured data deletion in cloud storage.

## REFERENCES

- [1] C. Wu, A. N. Toosi, R. Buyya, and K. Ramamohanarao, "Hedonic pricing of cloud computing services," *IEEE Trans. Cloud Comput.*, vol. 9, no. 1, pp. 182-196, Jan. 2021, doi: 10.1109/TCC.2018.2858266.
- [2] W. Hassan, T.-S. Chou, O. Tamer, J. Pickard, P. Appiah-Kubi, and L. Pagliari, "Cloud computing survey on services, enhancements and challenges in the era of machine learning and data science," *Int. J. Informatics Commun. Technol.*, vol. 9, no. 2, p. 117, Aug. 2020, doi: 10.11591/ijict.v9i2.pp117-139.
- [3] I. Odun-Ayo, O. Ajayi, B. Akanle, and R. Ahuja, "An overview of data storage in cloud computing," in *Proceedings - 2017 International Conference on Next Generation Computing and Information Systems, ICNGCIS 2017*, Nov. 2018, pp. 38-42, doi: 10.1109/ICNGCIS.2017.9.
- [4] J. Surbiryala and C. Rong, "Cloud computing: History and overview," in *Proceedings - 2019 3rd IEEE International Conference on Cloud and Fog Computing Technologies and Applications, Cloud Summit 2019*, Aug. 2019, pp. 1-7, doi: 10.1109/CloudSummit47114.2019.00007.
- [5] A. Elmorshidy, "Cloud computing: A new success model," in *Proceedings - 2019 International Conference on Future Internet of Things and Cloud Workshops, FiCloudW 2019*, Aug. 2019, pp. 7-12, doi: 10.1109/FiCloudW.2019.00015.
- [6] Y. Mansouri, A. N. Toosi, and R. Buyya, "Data storage management in cloud environments: Taxonomy, survey, and future directions," *ACM Computing Surveys*, vol. 50, no. 6. Association for Computing Machinery, Dec. 01, 2017, doi: 10.1145/3136623.
- [7] R. Mendes, T. Oliveira, V. Cogo, N. Neves, and A. Bessani, "CHARON: A secure cloud-of-clouds system for storing and sharing big data," *IEEE Transactions on Cloud Computing*, 2019.







- [8] W. Hassan, T.-S. Chou, X. Li, P. Appiah-Kubi, and T. Omar, "Latest trends, challenges and solutions in security in the era of cloud computing and software defined networks," *Int. J. Informatics Commun. Technol.*, vol. 8, no. 3, p. 162, Dec. 2019, doi: 10.11591/ijict.v8i3.pp162-183.
- [9] L. Zhang, H. Xiong, Q. Huang, J. Li, K. K. R. Choo, and J. Li, "Cryptographic solutions for cloud storage: challenges and research opportunities," *IEEE Trans. Serv. Comput.*, vol. 15, no. 1, pp. 567–587, 2022, doi: 10.1109/TSC.2019.2937764.
- [10] A. Hatamian, M. B. Tavakoli, and M. Moradkhani, "Improving the security and confidentiality in the internet of medical things based on edge computing using clustering," *Comput. Intell. Neurosci.*, vol. 2021, 2021, doi: 10.1155/2021/6509982.
- [11] Y. K. Kumar and R. M. Shafi, "An efficient and secure data storage in cloud computing using modified RSA public key cryptosystem," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 10, no. 1, pp. 530–537, 2020, doi: 10.11591/ijece.v10i1.pp530-537.
- [12] S. D. Capitani, D. Vimercati, S. Foresti, S. Paraboschi, G. Pelosi, and P. Samarati, "Three-server swapping for access confidentiality," *IEEE Trans. Cloud Comput.*, vol. 6, no. 2, pp. 492–505, Apr. 2018, doi: 10.1109/TCC.2015.2449993.
- [13] W. Shen, J. Qin, J. Yu, R. Hao, J. Hu, and J. Ma, "Data integrity auditing without private key storage for secure cloud storage," *IEEE Trans. Cloud Comput.*, vol. 9, no. 4, pp. 1408–1421, 2021, doi: 10.1109/TCC.2019.2921553.
- [14] B. Li, Q. He, F. Chen, H. Jin, Y. Xiang, and Y. Yang, "Auditing cache data integrity in the edge computing environment," *IEEE Trans. Parallel Distrib. Syst.*, vol. 32, no. 5, pp. 1210–1223, May 2021, doi: 10.1109/TPDS.2020.3043755.
- [15] M. M. S. Altaee and M. Alanezi, "Enhancing cloud computing security by paillier homomorphic encryption," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 11, no. 2, pp. 1771–1779, Apr. 2021, doi: 10.11591/ijece.v11i2.pp1771-1779.
- [16] S. Amghar, Y. Tabaa, and A. Medouri, "Secure confidential big data sharing in cloud computing using KP-ABE," in *ACM International Conference Proceeding Series*, Mar. 2017, vol. Part F129474, doi: 10.1145/3090354.3090388.
- [17] R. F. Abdel-Kader, S. H. El-Sherif, and R. Y. Rizk, "Efficient two-stage cryptography scheme for secure distributed data storage in cloud computing," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 10, no. 3, pp. 3295–3306, 2020, doi: 10.11591/ijece.v10i3.pp3295-3306.
- [18] Y. Xie, H. Wen, B. Wu, Y. Jiang, and J. Meng, "A modified hierarchical attribute-based encryption access control method for mobile cloud computing," *IEEE Trans. Cloud Comput.*, vol. 7, no. 2, pp. 383–391, Apr. 2019, doi: 10.1109/TCC.2015.2513388.
- [19] Z. Xie, W. Fu, J. Xu, and T. Zhu, "Assured deletion: a scheme based on strong nonseparability," *J. Sensors*, vol. 2022, pp. 1–8, Mar. 2022, doi: 10.1155/2022/9691724.
- [20] D. Zheng, L. Xue, C. Yu, Y. Li, and Y. Yu, "Toward assured data deletion in cloud storage," *IEEE Netw.*, vol. 34, no. 3, pp. 101–107, May 2020, doi: 10.1109/MNET.011.1900165.
- [21] Z. Maohong, Y. Aihua, and L. Hui, "Research on security and privacy of big data under cloud computing environment," in *ACM International Conference Proceeding Series*, Oct. 2018, pp. 52–55, doi: 10.1145/3291801.3291820.
- [22] J. Li, J. Wei, W. Liu, and X. Hu, "PMDP: A framework for preserving multiparty data privacy in cloud computing," *Secur. Commun. Networks*, vol. 2017, 2017, doi: 10.1155/2017/6097253.
- [23] L. Huang and H. H. Lee, "A medical data privacy protection scheme based on blockchain and cloud computing," *Wirel. Commun. Mob. Comput.*, vol. 2020, 2020, doi: 10.1155/2020/8859961.
- [24] K. He, J. Chen, Q. Yuan, S. Ji, D. He, and R. Du, "Dynamic group-oriented provable data possession in the cloud," *IEEE Trans. Dependable Secur. Comput.*, vol. 18, no. 3, pp. 1394–1408, May 2021, doi: 10.1109/TDSC.2019.2925800.
- [25] H. Wang, D. He, A. Fu, Q. Li, and Q. Wang, "Provable data possession with outsourced data transfer," *IEEE Trans. Serv. Comput.*, vol. 14, no. 6, pp. 2000–2010, 2021, doi: 10.1109/TSC.2019.2892095.
- [26] H. Yang *et al.*, "Improved outsourced provable data possession for secure cloud storage," *Secur. Commun. Networks*, vol. 2021, 2021, doi: 10.1155/2021/1805615.
- [27] C. Yang, X. Tao, and Q. Chen, "New publicly verifiable data deletion supporting efficient tracking for cloud storage," *Int. J. Netw. Secur.*, vol. 22, no. 5, pp. 885–896, 2020, doi: 10.6633/IJNS.202009.
- [28] C. Yang, X. Tao, and F. Zhao, "Publicly verifiable data transfer and deletion scheme for cloud storage," *Int. J. Distrib. Sens. Networks*, vol. 15, no. 10, Oct. 2019, doi: 10.1177/1550147719878999.
- [29] Y. Liu, X. A. Wang, Y. Cao, D. Tang, and X. Yang, "Improved provable data transfer from provable data possession and deletion in cloud storage," in *Lecture Notes on Data Engineering and Communications Technologies*, vol. 23, Springer Science and Business Media Deutschland GmbH, 2019, pp. 445–452.
- [30] S. Tezuka, R. Uda, and K. Okada, "ADEC: Assured deletion and verifiable version control for cloud storage," in *Proceedings - International Conference on Advanced Information Networking and Applications*, 2012, pp. 23–30, doi: 10.1109/AINA.2012.116.
- [31] K. M. Ramokapane, A. Rashid, and J. M. Such, "Assured deletion in the cloud: Requirements, challenges and future directions," in *CCSW 2016 - Proceedings of the 2016 ACM Cloud Computing Security Workshop, co-located with CCS 2016*, Oct. 2016, pp. 97–108, doi: 10.1145/2996429.2996434.
- [32] Z. Yan, L. Zhang, W. Ding, and Q. Zheng, "Heterogeneous data storage management with deduplication in cloud computing," *IEEE Trans. Big Data*, vol. 5, no. 3, pp. 393–407, Jul. 2019, doi: 10.1109/TBDATA.2017.2701352.
- [33] 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*, May 2016, pp. 17–23, doi: 10.1145/2898445.2898447.
- [34] 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.
- [35] H. Mu and Y. Li, "An assured deletion scheme for encrypted data in internet of things," *Adv. Mech. Eng.*, vol. 11, no. 2, Feb. 2019, doi: 10.1177/1687814019827147.
- [36] G. Wang and Y. Luo, "A review on assured deletion of cloud data based on cryptography," in *Procedia Computer Science*, 2021, vol. 187, pp. 580–585, doi: 10.1016/j.procs.2021.04.111.
- [37] S. B. Joshi and S. D. Panchal, "A survey on assured data deletion in cloud storage," *Int. J. Comput. Sci. Eng.*, vol. 7, no. 6, pp. 548–553, Jun. 2019, doi: 10.26438/ijcse/v7i6.548553.
- [38] H. Tabrizchi and M. K. Rafsanjani, "A survey on security challenges in cloud computing: issues, threats, and solutions," *J. Supercomput.*, vol. 76, no. 12, pp. 9493–9532, Dec. 2020, doi: 10.1007/s11227-020-03213-1.
- [39] S. M. Diesburg and A. I. A. Wang, "A survey of confidential data storage and deletion methods," *ACM Comput. Surv.*, vol. 43, no. 1, Nov. 2010, doi: 10.1145/1824795.1824797.
- [40] M. Hua, Y. Zhao, and T. Jiang, "Secure data deletion in cloud storage: a survey," *International Journal of Embedded Systems*, vol. 12, no. 2, pp. 253–265, 2020.





- [41] D. A. B. Fernandes, L. F. B. Soares, J. V. Gomes, M. M. Freire, and P. R. M. Inácio, "Security issues in cloud environments: A survey," *Int. J. Inf. Secur.*, vol. 13, no. 2, pp. 113–170, 2014, doi: 10.1007/s10207-013-0208-7.
- [42] 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 Secur. Comput.*, vol. 9, no. 6, pp. 903–916, 2012, doi: 10.1109/TDSC.2012.49.
- [43] Y. Yu, L. Xue, Y. Li, X. Du, M. Guizani, and B. Yang, "Assured data deletion with fine-grained access control for fog-based industrial applications," *IEEE Trans. Ind. Informatics*, vol. 14, no. 10, pp. 4538–4547, Oct. 2018, doi: 10.1109/TII.2018.2841047.
- [44] J. Ma, M. Wang, J. Xiong, and Y. Hu, "CP-ABE-based secure and verifiable data deletion in cloud," *Secur. Commun. Networks*, vol. 2021, 2021, doi: 10.1155/2021/8855341.
- [45] C. Yang, Y. Liu, and X. Tao, "Assure deletion supporting dynamic insertion for outsourced data in cloud computing," *Int. J. Distrib. Sens. Networks*, vol. 16, no. 9, Sep. 2020, doi: 10.1177/1550147720958294.
- [46] Y. Tian, T. Shao, and Z. Li, "An efficient scheme of cloud data assured deletion," *Mob. Networks Appl.*, vol. 26, no. 4, pp. 1597–1608, Aug. 2021, doi: 10.1007/s11036-019-01497-z.
- [47] 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, doi: 10.1109/ACCESS.2020.2997351.
- [48] A. Rahumed, H. C. H. Chen, Y. Tang, P. P. C. Lee, and J. C. S. Lui, "A secure cloud backup system with assured deletion and version control," in *Proceedings of the International Conference on Parallel Processing Workshops*, 2011, pp. 160–167, doi: 10.1109/ICPPW.2011.17.
- [49] B. Hall and G. Manimaran, "An assured deletion technique for cloud-based IoT," in *2018 27th International Conference on Computer Communication and Networks (ICCCN) IEEE*, pp. 1-9, 2018.
- [50] J. Xiong *et al.*, "A secure data self-destructing scheme in cloud computing," *IEEE Trans. Cloud Comput.*, vol. 2, no. 4, pp. 448–458, Oct. 2014, doi: 10.1109/TCC.2014.2372758.

## BIOGRAPHIES OF AUTHORS



**Kawa Qambar Aziz**     received the B.Sc. degree in Computer Sciences from the University of Kirkuk, in 2007. He is currently pursuing to receive an M.Sc. degree in Computer Sciences from the College of Computer Sciences and Information Technology at the University of Kirkuk. He can be contacted at email: kawaqambar@gmail.com.



**Baban Ahmed Mahmood**     is currently the chairman of Networks Department at University of Kirkuk, Kirkuk, Iraq. He received a B.Sc., degree in Computer and Software engineering from University of Al-Mustansryah, Iraq, in 2003 and a M.Sc., degree in Computer Science from University of Sulaimaniya, Iraq, in 2009. He received a PhD degree in Computer Science from University of Kentucky, Lexington, Kentucky, USA 2016. He worked in the program of some international conferences. He reviewed many papers for several prestigious journals and conferences. He published his research work in the following areas: routing in ad hoc networks, security of source routing protocols in MANET, and security of health records via cloud. He can be contacted at email: baban.mahmoodjaf@gmail.com.