Assure deletion supporting dynamic insertion for outsourced data in cloud computing

Changsong Yang¹,²,³, Yueling Liu¹ and Xiaoling Tao¹,²

Abstract
With the rapid development of cloud computing, an increasing number of data owners are willing to employ cloud storage service. In cloud storage, the resource-constraint data owners can outsource their large-scale data to the remote cloud server, by which they can greatly reduce local storage overhead and computation cost. Despite plenty of attractive advantages, cloud storage inevitably suffers from some new security challenges due to the separation of outsourced data ownership and its management, such as secure data insertion and deletion. The cloud server may maliciously reserve some data copies and return a wrong deletion result to cheat the data owner. Moreover, it is very difficult for the data owner to securely insert some new data blocks into the outsourced data set. To solve the above two problems, we adopt the primitive of Merkle sum hash tree to design a novel publicly verifiable cloud data deletion scheme, which can also simultaneously achieve provable data storage and dynamic data insertion. Moreover, an interesting property of our proposed scheme is that it can satisfy private and public verifiability without requiring any trusted third party. Furthermore, we formally prove that our proposed scheme not only can achieve the desired security properties, but also can realize the high efficiency and practicality.

Keywords
Cloud storage, secure data deletion, dynamic data insertion, public verifiability, Merkle sum hash tree

Date received: 28 March 2020; accepted: 20 August 2020

Handling Editor: Jerzy Balicki

Introduction
Cloud computing is the fusion, development, and application of the concepts of parallel computing, grid computing, and distributed computing.¹,² It can connect large-scale distributed resources through network and form a pool of computing resources to provide tenants with a series of on-demand services, such as data sharing service,³,⁴ data migration service,⁵ and data storage service (i.e. cloud storage service).⁶,⁷ Due to a lot of attractive advantages, these services have been widely applied in the daily life and work, especially for the cloud storage service. By employing cloud storage service, the resource-constraint data owners, including the individuals and enterprises, can outsource their large-scale data to the remote cloud server, which can help them to greatly reduce local software/hardware overhead and human resource investments.

Although employing cloud storage service is economically attractive, it inevitably suffers from some new security problems. The data owners lose the direct control over their outsourced data due to the separation of data ownership and management.⁸,⁹ Therefore,
outsourced data security is the major concern in cloud storage. Previous work mainly studied the existence of outsourced data, that is, data integrity, which has been already well solved. However, we focus on a complementary issue, that is, secure data deletion which received much less attentions than the data integrity. In cloud storage, outsourced data deletion operation will be executed by the remote cloud server. However, the selfish cloud server might maliciously reserve some data backups for the following two factors. On the one hand, the cloud server can dig some implicit benefits from the data reservation. On the other hand, outsourced data deletion operation may cost some expensive computational overhead, which is unexpected from the cloud server’s point of view. Hence, the cloud server may maliciously reserve some data and return a wrong data deletion result to cheat the data owner.

Some solutions have been proposed to solve the problem of secure data deletion, however, there are still some inherent limitations in the existing data deletion schemes. On the one hand, most of the existing verifiable outsourced data deletion schemes need to introduce a trusted third party (TTP), which would become a bottleneck because it may refuse to serve due to the heavy computational overhead. Moreover, the security of the TTP is worrying since it will more easily attract the adversary’s attentions. Meanwhile, the TTP cannot resist the commands of the court and government. On the other hand, the efficiency of the existing data deletion schemes is unsatisfactory, especially for these schemes that achieve deletion by overwriting.

In the overwriting model, the outsourced data must be overwritten by some random data, whose size is the same with the outsourced data. However, the outsourced data might be large-scale, resulting in heavy computational cost and communication overhead. Therefore, how to efficiently achieve publicly verifiable outsourced data deletion without requiring any TTP is a problem that needs to be solved solidly.

Besides secure data deletion, dynamic data insertion is another concern. In cloud storage, data insertion has become a fundamental requirement of the data owner. The data insertion operation will be executed by the remote cloud server because the data owner loses the direct control over the outsourced data. However, the cloud server might not honestly perform the data insertion operation since it may cost some computing resources and storage spaces. Therefore, how to conveniently insert some new data blocks into the outsourced data set and efficiently verify the data insertion result is another severe security challenge. To the best of our knowledge, it seems that there is no research work on publicly verifiable outsourced data deletion scheme that simultaneously supports dynamic data insertion without requiring any TTP. Therefore, it is very meaningful to design a publicly verifiable data deletion scheme with dynamic data insertion for cloud storage.

Our contributions
In this article, we mainly study the problem of designing publicly verifiable outsourced data deletion scheme that simultaneously supports dynamic data insertion in cloud storage. The main contributions of this article are listed into the following two aspects.

- We adopt Merkle sum hash tree (MSHT) to propose a new publicly verifiable data deletion scheme for cloud storage. The proposed scheme can also simultaneously achieve provable data storage and dynamic data insertion. Moreover, through using the advantages of MSHT, the proposed scheme can achieve public verifiability without requiring any TTP, which is better than most of the existing solutions.
- We formally prove that the proposed scheme can satisfy the desired security requirements through detailed security analysis. Moreover, we empirically evaluate the performance overhead of the proposed scheme through simulation experiments, which can prove that the proposed scheme is efficient for the resource-constraint data owners. Meanwhile, it also can demonstrate the practicality of the proposal in real-world applications.

Related work
The problem of secure data deletion has been studied for several decades, resulting in a great number of solutions. All of the existing data deletion schemes can be categorized into three types through different deletion approaches. The first approach to reach outsourced data deletion is unlinking. When the data owner wants to delete a file, the storage system removes the file link from the underlying file system and returns a one-bit reply (success/failure) to the data owner. Although unlinking can very efficiently delete file link, the content of the file still remains on the storage medium. The adversaries could adopt tools to scan the disk to recover the deleted file easily.

The second approach to realize outsourced data deletion is overwriting. The main idea is that the storage system utilizes some random data to overwrite the disk that maintains the data. Paul and Saxena designed a new cloud data deletion scheme called “Proof of erasability” (PoE), which aims to provide data owner with the ability to verify the data deletion result. Their scheme uses random patterns to overwrite...
the disk and returns the same patterns as the data deletion evidence. Luo et al. adopted hourglass function to design a new assured data deletion scheme. In their proposed scheme, they assume that the cloud server only maintains the latest version of the data owner’s file, and all the file backups will be consistent when they are updated. Therefore, they can reach data deletion by updating all the file backups with random data. That is, they disguise the overwriting operation as the data updating operation. Finally, the data owner can verify the returned data deletion result through a challenge-response protocol.

The third approach to achieve outsourced data deletion is destroying data decryption key. The main idea is that the data owner encrypts the file before outsourcing. Therefore, when the data will not be needed anymore, the data owner can destroy the corresponding decryption key to reach data deletion. Boneh and Lipton proposed the first cryptography-based solution to process the secure data deletion problem, with a series of follow-up schemes. The TTP would become a bottleneck that impedes the development of the verifiable outsourced data deletion system. To solve this problem, Yang et al. presented a new publicly verifiable data deletion scheme for cloud storage, which can also simultaneously achieve fine-grained access control. Hao et al. used trusted platform model (TPM) to design a publicly verifiable deletion scheme for secret data. Yang et al. adopted invertible Bloom filter (IBF) to propose a fine-grained data deletion scheme. However, the above schemes need to introduce a TTP (i.e., attribute authority), which is similar to schemes. The TTP would become a bottleneck that impedes the development of the verifiable outsourced data deletion system. To solve this problem, Yang et al. presented a new publicly verifiable data deletion scheme for cloud storage, which is characterized by confidential data storage, data integrity verification, and verifiable data deletion. Their scheme makes use of blockchain to achieve publicly verifiable deletion without requiring any TTP.

Moreover, Yu et al. focused on how to simultaneously achieve verifiable data transfer and deletion in cloud computing. Their scheme can satisfy the properties of data integrity, data confidentiality, secure data transfer, and verifiable data deletion. To the best of our knowledge, their scheme is the first solution to the problem of efficient data migration and provable transferred data deletion. Later, Xue et al. proposed a similar scheme, which can improve the efficiency in the data deletion process compared to scheme. However, Liu et al. pointed out that there is a security flaw in scheme and then proposed an improved verifiable data transfer protocol, which can also achieve verifiable transferred data deletion. However, the above data transfer and deletion schemes need a third party auditor (TPA). To remove the TTA, Yang et al. designed a publicly verifiable data transfer and deletion scheme based on vector commitment (VC). Their scheme allows the data owner to migrate outsourced data from the original cloud server to the target cloud server and then delete the transferred data from the original cloud server.

**Organization**

The rest of this article is organized as follows: In section “Preliminaries,” we describe the preliminary of MSHT, which can be viewed as an extension of the traditional Merkle hash tree (MHT). In section “Models and requirements,” we formalize the system model, present the main security challenges, and identify the design goals. In section “Assured deletion with dynamic insertion,” we propose a concrete publicly verifiable data deletion scheme that also supports dynamic data insertion. The detailed security analysis and the performance evaluation are provided in sections “Security analysis” and “Performance evaluation,” respectively. Finally, we simply conclude this article in section “Conclusion.”

**Preliminaries**

Merkle sum hash tree (MSHT) was first proposed by Miao et al., which can be viewed as an extension of the MHT. Different from the MHT, MSHT can support dynamic data insertion and deletion. Therefore, MSHT can be utilized to design dynamic data insertion and verifiable data deletion scheme. Moreover, the inputs of the MSHT are different from the MHT, as shown in Figure 1. For every leaf node, the inputs are the data items and the number of the data items that this leaf node contains, for example, $h_{11} = H(m_{11} \| 1)$, where 1 represents the i-th leaf node only maintains a data block $m_{11}$. In the internal node, the inputs are the concatenation of its two children, and the number of the data blocks that this internal node contains, for example, $h_{21} = H(h_{11} || h_{12} || 2)$. Similarly, we can compute the values for all the internal nodes. As a result, the hash value of the root node is $h_R = H(h_{21} || h_{22} || 4)$. Then, the cloud server

![Figure 1. An example of MSHT.](image-url)
generates a signature $\text{Sig}_R$ on $h_R$ with the traditional public key signature technology.

The MSHT can support dynamic data insertion and deletion, which are very attractive. Once a data insertion operation is executed, the MSHT should be also updated simultaneously. Without loss of generality, we assume that the data owner wants to insert a new data block $m_{3,2}$ into the third leaf node of the MSHT. Then, the hash value of the third leaf node should be updated as $h_{13} = H(m_{3,1}, h_{1,2})$. Simultaneously, the hash value of every node on the path from the third leaf node to the root node will be updated, as shown in Figure 2. Finally, a new signature on the updated root node will be regenerated through the public key signature technology.

Similarly, after executing a data deletion operation, the MSHT will be also updated. If the leaf node only maintains the data block that needs to be deleted, the leaf node will be removed and the MSHT will be updated, as shown in Figure 3. Otherwise, the cloud server only removes the data block that needs to be deleted, while the other data blocks still remain in the leaf node, as shown in Figure 4.

**Remark 1.** When the data owner wants to insert a new data block into the $i$-th leaf node of the MSHT, we assume that the new data block will be always inserted into the end of all the data blocks that are maintained by the leaf node $i$. That is, the new inserted data block is the last data block of the leaf node $i$.

**Remark 2.** To check that whether the MSHT maintains the data blocks honestly, the verifier also needs to utilize the auxiliary authentication information $\Phi_i$ to re-compute the root node of the MSHT and then verifies the validity of the signature. The auxiliary authentication information $\Phi_i$ is a data set that contains all of the sibling nodes on the path from the desired leaf node $i$ to the root node, such as $\Phi_i = (h_{1,2}, h_{2,2})$. Then, the verifier uses $\Phi_i$ to re-compute the root and check the storage result. Specifically, the verifier computes $h'_R = H(H(m_{1,1})||h_{1,2})||h_{2,2})$ and compares it with $h_R$. If equation $h_R = h'_R$ holds and signature $\text{Sig}_R$ is valid, the verifier believes that the data block $m_{1,1}$ is intact.
Models and requirements

In this section, we first formalize the system model of our novel scheme. Then, we describe the main security threats and design goals.

System model

In the system model of our scheme, there are two entities: a data owner and a cloud server, as shown in Figure 5. The data owner is a resource-constraint entity, who wants to outsource his large-scale personal data to the remote cloud server for greatly saving local storage overhead. Later, the data owner may insert some new data blocks into the outsourced data set. Meanwhile, the data owner wants to permanently delete some data blocks when he will not need them anymore. The cloud server has powerful computing ability and almost unlimited storage resources, and provides data storage service for data owners. Meanwhile, it will execute the data insertion and deletion operations according to the data owner’s commands and returns corresponding evidences. Therefore, the data owner can efficiently check the data insertion and deletion results by verifying the returned evidences.

Security challenges

In our scheme, we mainly consider the following three security challenges.

Data privacy leakage. The adversaries, such as hackers and curious manager of the cloud, want to learn some sensitive information from the outsourced data, which should be kept secret from the data owner’s point of view. Moreover, the selfish cloud server may share the data with other corporators, which may cause data leakage. Furthermore, the software/hardware failures might also cause data leakage. Hence, data privacy leakage is a severe challenge.

Dishonest data reservation. The selfish cloud server might not honestly delete the outsourced data for the following two factors. Firstly, the outsourced data may contain some valuable information. Therefore, the cloud server may maliciously reserve some data backups for digging some implicit benefits. Secondly, the data deletion operation may cost some expensive computational overhead, which is unexpected from the cloud server’s point of view. As a result, dishonest data reservation is another security challenge.

Malicious slander. Both the data owner and the cloud server may deny their behaviors and slander the other maliciously. On the one hand, the malicious cloud server might arbitrary delete some data but claims that it deleted the data according to the data owner’s command. On the other hand, the dis honest data owner had...
required the cloud server to delete the data. However, when he needs the data later, he might slander that the cloud server arbitrary deleted the outsourced data without his permission.

**Design goals**

In the proposed scheme, we should achieve the following three design goals.

- **Data confidentiality.** The data confidentiality means that the sensitive information is kept so secret that the attackers cannot obtain them maliciously. That is, the data owner should utilize secure encryption algorithm to encrypt the outsourced data before uploading and maintain the data decryption key secretly.

- **Verifiable data deletion.** Verifiable data deletion is defined that the data owner can efficiently verify the data deletion result. If the cloud server does not honestly execute the data deletion operation, it cannot generate an effective evidence to persuade the data owner that the data blocks have been deleted.

- **Accountable traceability.** The accountable traceability means that if the dishonest participant behaves maliciously, his malicious acts will be caught and proven. That is, both the data owner and the cloud server cannot deny their behaviors and slander the other successfully.

**Assured deletion with dynamic insertion**

**Overview**

We consider the verifiable data deletion model in cloud storage, which is very similar to schemes. In our scenario, there is a trust problem between the data owner and the cloud server. That is, the data owner does not believe that the cloud server might honestly execute data insertion and deletion operations. To solve this trust problem, plenty of solutions have been proposed, and most of them require a TTP. However, the TTP would become a bottleneck that impedes the development of verifiable data deletion system. To remove this bottleneck, we design a MSHT-based data deletion scheme, which can also achieve dynamic data insertion without requiring any TTP.

The main processes of the proposed scheme are described in Figure 6. In the first step, the data owner encrypts the outsourced file. After that, the data owner divides the ciphertext into \( n_0 \) blocks, then inserts \( n - n' \) random blocks into them at random positions, and outsources the final data set to the cloud server. The cloud server maintains the received data set in a MSHT and returns the root node and its signature. Thus, the data owner can check whether the cloud server maintains the data set honestly. Later, when the data owner wants to insert some new data blocks into the outsourced data set or delete some unnecessary data blocks, the cloud server executes the data insertion/deletion operations and returns corresponding evidences. Then, the data owner can check the data insertion/deletion results by verifying the returned evidences.

**The concrete construction**

In the following part, we propose a novel scheme, which can simultaneously achieve data insertion and deletion in cloud storage. Note that the cloud service provider must authenticate the identity of the data owner before providing him with data storage service. For simplicity, we assume that the data owner has already passed the verification and become a legal user. Hence, the data owner can directly enjoy the cloud storage service. Then, the concrete construction is described as follows:

- **Initialization.** Firstly, generate the elliptic curve digital signature algorithm (ECDSA) public/
Yang et al.

Data storage.

(1) The cloud server maintains the received data set $D$ that will be outsourced to the cloud server. Meanwhile, the data owner chooses a secure one-way collision resistant hash function $H_i(\cdot)$, and $\text{Sign}$ is a ECDSA signature generation scheme.

- **Data encryption.** To protect the sensitive information that contained in the outsourced file, the data owner should encrypt the file before uploading.

  (1) The data owner first computes a data encryption key $K = H_1(n_f||sk_o)$. Then, the data owner uses this key to encrypt the outsourced file $F$ as $C = \text{Enc}_K(F)$, where $\text{Enc}$ is an indistinguishability under chosen-plaintext attack (IND-CPA) advanced encryption standard (AES) algorithm.

  (2) Secondly, the data owner divides the ciphertext $C$ into $n'$ blocks. After that, the data owner inserts $n-n'$ random blocks into them at random positions and records these random positions in a table $PF$. Furthermore, the data owner can obtain a data set $D = (C_1,1,\ldots,C_{n,1})$. The random data blocks will not be deleted, which can guarantee that the MSHT will not be empty.

  (3) Finally, the data owner sends the outsourced data set $D$ to the cloud server, along with the file name $n_f$, which can be viewed as the index of the outsourced data set $D$.

- **Data storage.** The cloud server maintains the outsourced data set $D$ for the data owner and returns a corresponding storage proof. Thus, the data owner can check the storage result by verifying the returned storage proof.

  (1) The cloud server maintains the received data set $D$ in a MSHT. Specifically, the data block $C_{i,1}$ will be maintained in the $i$-th leaf node of the MSHT. Then, the cloud server generates a signature on the root node of the MSHT as $\text{Sig}_R = \text{Sign}_{sk_o}(h_R)$, where $h_R$ is the hash value of root node. Finally, the cloud server returns the data storage proof $(\text{Sig}_R, h_R)$ to the data owner.

  (2) Upon receiving the storage evidence $(\text{Sig}_R, h_R)$, the data owner randomly chooses a data block $C_{i,1}$ and obtains the corresponding auxiliary verification information $\Phi_i$ from the cloud server. Then, the data owner re-computes the root node $h'_R$ and checks the signature $\text{Sig}_R$. If the equation $h_R = h'_R$ holds and $\text{Sig}_R$ is a valid signature on the root node $h_R$, the data owner deletes the local backups.

- **Data insertion.** The data owner may want to insert some new data blocks into the outsourced data set $D$ and verify the corresponding data insertion result. The detailed processes are as follows:

  (1) The data owner first retrieves all the data blocks that are maintained by the leaf node $i$. After that, the data owner computes a signature $\text{Sig}_r = \text{Sign}_{sk_o}(n_f||i||T_i)$ and then generates a data insertion command $IR = (n_f,i,\text{Sig}_r,T_i)$, which means that the data owner wants to insert the new data block into the leaf node $i$, where $T_i$ is the timestamp. Finally, the data owner sends the data insertion command $IR$ to the cloud server, along with the new data block $C_i$, which is the ciphertext of file block $m_y$.

  (2) On receipt of the data insertion command $IR$ and the new data block $C_i$, the data owner, the cloud server checks the validity of $IR$ through signature verification. If $IR$ is valid, the cloud server inserts $C_i$ into the end of the data blocks that maintained by the leaf node $i$. Meanwhile, the cloud server updates the MSHT, and sends the new root node $h_r$ and the new signature $\text{Sig}_r$ to the data owner, along with the auxiliary verification information $\Phi_i$, where $\text{Sig}_r = \text{Sign}_{sk_o}(h_r)$.

  (3) Finally, the data owner re-computes the root node $h'_r$ and compares it with $h_r$ that is received from the cloud server. If the equation $h_r = h'_r$ holds, the data owner will further verify the validity of the signature $\text{Sig}_r$. If $\text{Sig}_r$ is a valid signature on the root node $h_r$, the data owner believes that the data insertion is successful.

- **Data deletion.** When some data blocks will not be needed anymore, the data owner may require the cloud server to permanently delete them and return a corresponding data deletion evidence.

  (1) The data owner first retrieves all the data blocks that are maintained by the leaf node $j$. Then, the data owner computes a signature $\text{Sig}_d = \text{Sign}_{sk_o}(n_f||j||q||T_d)$ and generates a data deletion command $DR = (n_f,j,q,T_d,\text{Sig}_d)$, which means that the data owner wants to delete the data block $C_{j,q}$ from the leaf node $j$. Finally, the data owner sends the data deletion command $DR$ to the cloud server.

  (2) On receiving the data deletion command $DR$, the cloud server verifies the validity of $DR$
through signature verification. If $DR$ is valid, the cloud server deletes the data block $C_{i,q}$ from the leaf node $j$ and updates the MSHT. If the leaf node $j$ only maintains the data block $C_{i,q}$, the MSHT is updated by deleting the leaf node $j$; otherwise, the MSHT is updated by deleting data block $C_{i,q}$. After that, a new root node $HR$ is re-computed and a new signature $Sig^r$ is generated, where $Sig^r = \text{Sign}_{sk}(HR)$. Finally, the deletion evidence $\tau = (HR, Sig^r)$ is returned to the data owner, along with the auxiliary verification information $\Phi_j$.

(3) Upon receiving $\tau$ and $\Phi_j$, the data owner can check the data deletion result. Firstly, the data owner re-computes the root node $HR'$ and compares it with $HR$. Meanwhile, the data owner verifies the validity of the signature $Sig^r$. If the verification is fail, the data owner refuses the data deletion result; otherwise, the data owner believes that the returned data deletion evidence is trustworthy. If the data block $C_{i,q}$ is later discovered on the cloud server, the data owner could be entitled to compensation.

Security analysis

Lemma 1. Assume that $(K, E, D)$ is an IND-CPA secure symmetric encryption scheme, $F$ is a pseudo-random function, where $K = \{0, 1\}^m$, $E : K \times K \rightarrow \mathcal{Y}$, and $K^F = \{0, 1\}^n$. After that, $(K^F, E)$ is IND-CPA secure, where $E^F_k(x) = E_{F^k_0}(x)$.

Proof. $(K^F, E)$ is IND-CPA secure means for any messages $x$, $y$, and any probability polynomial time (PPT) attacker $A$, there always exists a polynomial $p(n)$ and an integer $N$, if $n > N$, the following equation holds

$$\text{Adv}_{\text{CPA}}^{E^F} = \left| Pr \left[ A^{E^F} \left( E_{F^k_0}(x) \right) = 1 \right] - Pr \left[ A^{E^F} \left( E_{F^k_0}(y) \right) = 1 \right] \right| < \frac{1}{p(n)}$$

Then, we first define some games.

Game 0:
1. Challenger $C$ runs $\text{Gen}(1^n)$ to generate key $k$.
2. Input $n$, attacker $A$ asks random predictor $Enc_{F^k_0}(\cdot)$ outputs messages $m_0$ and $m_1$ with the same length.
3. Challenger $C$ chooses $b \in \{0, 1\}$ randomly, computes $c = Enc_{F^k_0}(m_b)$, and sends it to attacker $A$.
4. Attacker $A$ asks random predictor $Enc_{F^k_0}(\cdot)$, outputs $b'$ as a guess at $b$.
5. If $b = b'$, output 1; otherwise, output 0.

Game 1:
The Game 1 is very similar to the Game 0. The only difference between the Game 0 and the Game 1 is that attacker $A$ asks random predictor $Enc_{k}(\cdot)$. Hence, we do not describe Game 1 repeatedly.

Game 2:
Game 2 is similar to Game 1. The difference is that the encrypted message is independent of the key. Based on the indistinguishability of pseudo-random function and the IND-CPA security of $(K^F, E)$, we can prove that the Game 0 and Game 1 are indistinguishable. That is, for any message $x$ and any PPT attacker $A$, there exists a polynomial time algorithm $p(n)$ and an integer $N$, when $n > N$, the following equation holds

$$\text{Adv}_{\text{CPA}}^{E} = \left| Pr \left[ A^{E^2} \left( E_{F^k_0}(x) \right) = 1 \right] - Pr \left[ A^{E^2} \left( E_{F^k_0}(y) \right) = 1 \right] \right| < \frac{1}{p(n)}$$

Similarly, for any message $y$ and PPT attacker $A$, there exists a polynomial $p(n)$ and an integer $N$, when $n > N$, the following equation holds

$$\text{Adv}_{\text{CPA}}^{E} = \left| Pr \left[ A^{E^2} \left( E_{F^k_0}(y) \right) = 1 \right] - Pr \left[ A^{E^2} \left( E_{F^k_0}(x) \right) = 1 \right] \right| < \frac{1}{p(n)}$$

As a result, we can say that the Game 0 and Game 1 are indistinguishable. Moreover, we can prove that the Game 1 and the Game 2 are indistinguishable. Since the proof is similar, so we do not repeat it here in detail.

Theorem 1. The proposed scheme can satisfy outsourced data confidentiality.

Proof. Data confidentiality can ensure that only the data owner can decrypt the outsourced ciphertext correctly. In the proposed scheme, the outsourced file is encrypted with an IND-CPA secure symmetric encryption algorithm before uploading. Therefore, based on Lemma 1, we can find that our proposed scheme can guarantee only the data owner can decrypt the ciphertext correctly. That is, the proposed scheme can satisfy outsourced data confidentiality.

Theorem 2. The proposed scheme can satisfy verifiable outsourced data deletion.

Proof. Verifiable data deletion can guarantee that if the cloud server does not honestly delete the data blocks, it cannot generate an effective deletion evidence to persuade the data owner that the data blocks have been deleted. In the proposed scheme, the data owner inserts $n - n'$ random blocks into the data set at random positions. When the data owner wants to delete some unnecessary data blocks or even the whole file $F$, the
inserted random data blocks will be remained to generate the data deletion evidence. The data owner can also use the auxiliary authentication information to re-compute a new root node and then compares it with the one received from cloud server. The data insertion and deletion operations may change the path from one block to the root. As a result, there are plenty of paths that can reach the root node from the remaining leaf nodes. Without the loss of generality, we can assume that there are \( k \) data blocks left on the cloud server. Thus, there are \( (2k - 2)!/k!(k - 1)! \) ways to generate the root node of MSHT. That is, the probability of the cloud server can guess the right path to the root node is \( P_f = k!(k - 1)!/(2k - 2)! \). When the number \( k \) is large enough (e.g. when \( k > 8192 \)), the probability \( P_f \) is so small that it can be negligible. Note that the outsourced file is large-scale. Hence, the number \( k \) is large enough. Furthermore, if the new root node that is re-computed by the data owner can successfully pass the verification, it means that the cloud server has indeed performed the data deletion command honestly.

Meanwhile, note that in the data deletion result verification process, the verifier uses the specific leaf node, the corresponding auxiliary authentication information, and the public key of the cloud server to verify the data deletion result. The leaf node maintains the ciphertext blocks of the outsourced file. That is, the verification process does not need any private information of the data owner and the cloud server. Furthermore, any verifier who owns the data deletion evidence can verify the data deletion result. Therefore, we can think that the proposed scheme can achieve publicly verifiable data deletion.

**Theorem 3.** The proposed scheme can satisfy accountable traceability.

**Proof.** Accountable traceability can guarantee that the malicious participant cannot successfully slander the other. The detailed analyses are presented as follows.

**Dishonest data owner**

If the data owner is dishonest, he may maliciously deny his behavior and slander the cloud server. Specifically, the dishonest data owner had already required the cloud server to delete some data blocks. However, when the data owner needs the deleted data blocks later, he may deny his deletion command and slander that the cloud server arbitrarily deleted those data blocks. In this case, the cloud server can show the deletion command \( DR \) that is received from the data owner. Note that \( DR \) contains a signature \( \text{Sig}_d \), which is computed by the data owner with his private key \( sk_o \). That is, only the data owner can generate \( DR \) and no one else can forge it. Hence, \( DR \) can be viewed as an evidence, which can prove that the data owner had required the cloud server to delete the data blocks. Furthermore, the dishonest data owner cannot deny his data deletion command and slander the cloud server successfully.

**Malicious cloud server**

Similarly, the malicious cloud server may also slander the data owner dishonestly. Specifically, the malicious cloud server may arbitrarily delete some data blocks that are rarely accessed for saving overhead. And if the dishonest data deletion is discovered, the malicious cloud server may slander that he deleted the data blocks according to the data owner’s command. In this case, the data owner can require the cloud server to demonstrate the deletion command \( DR \), which contains a signature \( \text{Sig}_d \) that only can be generated by the data owner with his secret key \( sk_o \). Hence, the malicious cloud server cannot successfully forge \( DR \) with a non-negligible probability. Moreover, note that the data owner had never generated such a \( DR \). Hence, the malicious cloud server cannot prove that the data owner had required to delete the data blocks. That is, the malicious cloud server cannot successfully deny his dishonest data deletion operation with a non-negligible probability.

**Performance evaluation**

In this section, we first compare the functionality of the proposed scheme and some existing solutions. Then, we provide the numeric analysis. Finally, we provide the efficiency evaluation through simulation experiments.

**Functionality comparison**

In this part, we compare the functionality of the proposed scheme with three existing solutions\(^{16,24,40}\) in theory, and the comparison results are shown in Table 1. From Table 1, we can easily obtain the following four findings. Firstly, only scheme\(^{40}\) does not consider data confidentiality, while the other three scheme all can guarantee outsourced data confidentiality. Secondly, all these four schemes are able to achieve publicly verifiable deletion for outsourced data, but only scheme\(^{16}\) cannot satisfy the property of accountable traceability, which is very different from the other three solutions. Thirdly, dynamic and efficient data insertion has become one of the most fundamental requirements from the data owner’s point of view. Therefore, the proposed scheme provides the data owner with the ability to dynamically insert some new data blocks into the outsourced data set. However, the other three schemes cannot achieve dynamic data insertion. Last but not least, the TTP could become a bottleneck that impedes the development of verifiable data
deletion system. Therefore, our proposed scheme removes the TTP, which is much better than the other three schemes. Overall, we can think that our proposed scheme is much more attractive.

**Numeric analysis**

In the following, we describe the theoretical computational complexity in each process. For simplicity, we utilize the symbols $S$ and $V$ to represent the ECDSA signature generation calculation and ECDSA signature verification operation, respectively. Meanwhile, symbol $E$ represents a traditional symmetrical encryption operation (e.g. AES), $E$ represents a modular exponentiation calculation, and symbol $H$ represents an operation of computing a hash value. Moreover, we denote by $n$ as the total number of outsourced data blocks, $k$ by the number of data blocks that will be inserted, and $l$ by the number of data blocks that will be deleted. We can assume that every data block in scheme$^{40}$ is further split into $s$ sectors. For simplicity, we do not consider other operations, such as the addition calculation and the communication overhead between the data owner and the cloud server. After that, we compare the theoretical computational complexity, as shown in Table 2.

From Table 2, we can find that our proposed scheme needs a hash calculation and an encryption operation to encrypt the outsourced file. However, scheme$^{16}$ requires four hash computations and two encryption operations, and scheme$^{24}$ needs $n$ hash calculations and an encryption operation. In data storage process, our proposed scheme needs $2n + \log_2 n$ hash computations, a signature generation, and a signature verification calculation. However, scheme$^{24}$ requires $43n$ hash computations, a signature generation, and a signature checking operation, and scheme$^{40}$ needs $2n - 1$ hash calculations, two signature generation operations, and $ns$ modular exponentiation calculations. To delete $l$ data blocks, our proposed scheme costs more computational overhead than scheme$^{16}$ but it costs less overhead than scheme$^{24}$. Meanwhile, the growth rate of our proposed scheme is relatively lower than that of scheme$^{40}$. As a result, we can think that the overall efficiency of our proposed scheme is much more attractive.

**Simulation results**

In this part, we simulate the proposed scheme based on the open secure socket layer (OpenSSL) library and the pairing-based cryptography (PBC) library. In our experiments, we choose SHA-1 as the secure hash function. Moreover, all the experiments are executed on a Linux machine with Intel(R) Core(TM) i5-6200U processors running at 8 GB main memory and 2.4 GHz. For simplicity, we ignore some other computations, such as addition calculation, multiplication calculation, and communication cost.

**Computation of data encryption.** Note that the dominated computational overhead in this phase is encrypting the outsourced file. Hence, we increase the encrypted file from 1 to 10 MB with a step for 1 MB in this experiment. Meanwhile, we fix the number $n = 8192$ in our proposed scheme and scheme$^{24}$ for simplicity, and Figure 7 shows the approximate time cost comparison. Although the time cost increases with the size of encrypted file, the data encryption is one-time. Meanwhile, note that our proposed scheme needs least.

| Scheme | Scheme$^{16}$ | Scheme$^{24}$ | Scheme$^{40}$ | Our scheme |
|--------|--------------|--------------|--------------|------------|
| Encrypt | $4H + 2E$    | $nH + E$     | –            | $1H + E$   |
| Store  | –            | $43nH + 1S + 1V$ | $(2n - 1)H + 2S + nE$ | $(2n + \log_2 n)H + 1S + 1V$ |
| Insert | –            | –            | $(2S + 2V + 2k)H$ | $(2S + 2V + 2l)H$ |
| Delete | $1S + 1V$   | $2S + 2V + 43H$ | $2S + 1V + 3l\log_2 nH$ | $2S + 2V + 2l(\log_2 n + 1)H$ |

| Table 1. Functionality comparison. |
|-----------------------------------|
| Scheme | Scheme$^{16}$ | Scheme$^{24}$ | Scheme$^{40}$ | Our scheme |
|--------|--------------|--------------|--------------|------------|
| System model | Amortized | Amortized | Amortized | Amortized |
| Trusted third party | Yes | Yes | Yes | No |
| Data confidentiality | Yes | Yes | No | Yes |
| Dynamic data insertion | No | No | No | Yes |
| Verifiable data deletion | Yes | Yes | Yes | Yes |
| Accountable traceability | No | Yes | Yes | Yes |
time cost because scheme\textsuperscript{16} needs to execute additional encryption to generate the message authentication code (MAC), and scheme\textsuperscript{24} needs more hash calculations to generate the data encryption keys. Moreover, the growth rate of our proposed scheme is relatively lower than these of the other two schemes. Hence, we can think that our proposed scheme is more efficient to encrypt the outsourced file than the other two schemes.

Computational of data storage. In this process, the main computational overhead comes from data storage evidence generation and verification, which increases with the number of outsourced data blocks. Therefore, we increase the number of outsourced data blocks from $2^3$ to $2^{12}$ in this simulation experiment. Moreover, for simplicity, the number $s$ in scheme\textsuperscript{40} is fixed in $s = 100$. After that, we can test the approximate time overhead, as shown in Figure 8. We can find that our proposed scheme costs least time overhead. Meanwhile, although the time cost increases with the number of outsourced data blocks, the growth rate of our proposed scheme is relatively lower than these of the other two schemes.\textsuperscript{24,40} Moreover, note that the data storage operation not only is one-time but also is accomplished off-line. Therefore, we can think that our proposed scheme is much more efficient than the other two schemes\textsuperscript{24,40} in data storage process.

Computational of data insertion. For simplicity, we assume that the new blocks will not be inserted into the leaf nodes that have the same parent node. Meanwhile, we increase the height of the MSHT from 10 to 50 with a step for 20. Meanwhile, we increase the number of inserted data blocks from 5 to 50 with a step for 5. After that, we can test the approximate time overhead, as shown in Figure 9, where $d$ is the height of the MSHT. We can easily find from Figure 9 that the time overhead increases with both the height of the MSHT and the number of inserted data blocks. Fortunately, the growth rates are not very high and the time overhead is quite small. Therefore, we can think that the proposed scheme is efficient in data insertion step. Furthermore, it will not greatly affect the overall efficiency of our proposed scheme.

Moreover, we test the data owner’s time overhead in data insertion process, as shown in Figure 10. We can easily find that the growth rate is very low. Moreover, compared with the total time shown in Figure 9, the time cost shown in Figure 10 is much less. It means that plenty of computations are executed by the cloud server. Meanwhile, note that the data insertion result

![Figure 7. The time cost of encrypting.](image1)

![Figure 8. The time cost of storage verification.](image2)

![Figure 9. Total time cost in data insertion.](image3)

![Figure 10. The data owner's time overhead.](image4)
verification operation is one-time, and it can be completed off-line by the data owner. Furthermore, we can think that our proposed scheme is efficient for the data owner in data insertion phase.

**Computation of data deletion.** In this process, the dominated computational overhead comes from data deletion command, data deletion evidence generations and data deletion result verification. In this experiment, we fix the number of outsourced data blocks in \( n = 8192 \) for simplicity. Then, we increase the number of deleted data blocks from 10 to 100 with a step for 10, and then Figure 11 shows the time overhead comparison among our proposed scheme and the other three schemes.\(^{16,24,40}\) From Figure 11, we can easily find that the time cost of scheme\(^{16}\) is constant, which is different from the other three schemes. Fortunately, the growth rate of our proposed scheme is lower than these of schemes.\(^{24,40}\) Meanwhile, when the number of deleted data blocks is larger than 90, our proposed scheme costs less time overhead than schemes.\(^{24,40}\) Therefore, we still can think that our proposed scheme is efficient to delete outsourced data blocks.

**Conclusion**

In cloud storage, there is a trust problem between the data owner and the cloud server. That is, the data owner does not believe that the cloud server may honestly execute outsourced data insertion and deletion operations. In order to solve this trust problem, we adopt the primitive of MSHT as a building block to design a publicly verifiable outsourced data deletion scheme, which can also achieve provable data storage and dynamic data insertion. If the cloud server does not behaves honestly, the data owner can detect the dishonest acts with an overwhelming probability. By utilizing the attractive advantages of MSHT, the proposed scheme can both achieve private and public verifiability without requiring any TTP. Moreover, we prove that the proposed scheme can satisfy the desired design goals through formal security analysis. Finally, we simulate the proposed scheme and provide the efficiency evaluation, which is able to intuitively demonstrate the efficiency and practicability of the proposed scheme.

**Declaration of conflicting interests**

The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

**Funding**

The author(s) disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: This work was supported by the Opening Project of Shanghai Key Laboratory of Integrated Administration Technologies for Information Security, the National Natural Science Foundation of China (No. 61962015), the Natural Science Foundation of Guangxi (No. 2016GXNSFAA380098), and the Science and Technology Program of Guangxi (No. AB17195045).

**ORCID iDs**

Changsong Yang https://orcid.org/0000-0002-5273-8678  
Xiaoling Tao https://orcid.org/0000-0002-6573-2291

**References**

1. Buyya R, Yeo CS, Venugopal S, et al. Cloud computing and emerging IT platforms: vision, hype, and reality for delivering computing as the 5th utility. *Future Gener Comp Syst* 2009; 25(6): 599–616.
2. Yang C and Ye J. Secure and efficient fine-grained data access control scheme in cloud computing. *J High Speed Netw* 2015; 21: 259–271.

3. Shen J, Zhou T, Chen X, et al. Anonymous and traceable group data sharing in cloud computing. *IEEE T Inf Foren Sec* 2017; 13(4): 912–925.

4. Han S, Han K and Zhang S. A data sharing protocol to minimize security and privacy risks of cloud storage in big data era. *IEEE Access* 2019; 7: 60290–60298.

5. Wang Y, Tao X, Ni J, et al. Data integrity checking with reliable data transfer for secure cloud storage. *Int J Web Grid Serv* 2018; 14: 106–121.

6. Dhal K, Pattnaik PK and Rai SC. RACC: an efficient and revocable fine grained access control model for cloud storage. *Int J Knowl Intell Eng Syst* 2019; 23(1): 21–32.

7. Zhang J, Wang B, He D, et al. Improved secure fuzzy auditing protocol for cloud data storage. *Soft Comput* 2019; 23(10): 3411–3422.

8. Takabi H, Joshi JB and Ahn GJ. Security and privacy challenges in cloud computing environments. *IEEE Secur Prat* 2010; 8(6): 24–31.

9. Stojmenovic I, Wen S, Huang X, et al. An overview of fog computing and its security issues. *Concur Comp: Pract E* 2016; 28(10): 2991–3005.

10. Jiang T, Chen X and Ma J. Public integrity auditing for shared dynamic cloud data with group user revocation. *IEEE Trans Comput* 2015; 65(8): 2363–2373.

11. Zhu Y, Wang H, Hu Z, et al. Dynamic audit services for integrity verification of outsourced storages in clouds. In: *Proceedings of the 2011 ACM symposium on applied computing*, Taichung, Taiwan, 21–24 March 2011, pp.1550–1557. New York: ACM.

12. Liu D, Shen J, Chen Y, et al. Privacy-preserving data outsourcing with integrity auditing for lightweight devices in cloud computing. In: *Proceedings of the international conference on information security and cryptology*, Fuzhou, China, 14–17 December 2018, pp.223–239. Cham: Springer.

13. Yang C and Tao X. New publicly verifiable cloud data deletion scheme with efficient tracking. In: *Proceedings of the 21th international conference on security with intelligent computing and big-data services*, Guilin, China, 14–16 December 2018, vol. 895, pp.359–372. Cham: Springer.

14. Ni J, Lin X, Zhang K, et al. Secure outsourced data transfer with integrity verification in cloud storage. In: *Proceedings of 2016 IEEE/CIC international conference on communications in China*, Chengdu, China, 27–29 July 2016, pp.1–6. New York: IEEE.

15. Yang C, Liu Y, Tao X, et al. Publicly verifiable and efficient fine-grained data deletion scheme in cloud computing. *IEEE Access* 2020; 8(1): 99393–99403.

16. Hao F, Clarke D and Zorzo AF. Deleting secret data with public verifiability. *IEEE T Depend Secure* 2016; 13: 617–629.

17. Wright C, Kleiman D and Sundhar RSS. Overwriting hard drive data: the great wiping controversy. In: *Proceedings of the 4th international conference on information systems security*, Hyderabad, India, 16–20 December 2008, pp.243–257. Berlin; Heidelberg: Springer.

18. Reardon J, Basin S and Capkun S. SoK: secure data deletion. In: *Proceedings of the 2013 IEEE symposium on security and privacy*, Berkeley, CA, 19–22 May 2013, pp.301–315. New York: IEEE.

19. Miao M, Wang J and Ma J. New publicly verifiable databases supporting insertion operation. In: *Proceedings of the 2015 18th international conference on network-based information systems*, Taipei, Taiwan, 2–4 September 2015, pp.640–642. New York: IEEE.

20. Yang C, Chen X and Xiang Y. Blockchain-based publicly verifiable data deletion scheme for cloud storage. *J New Comput Appl* 2018; 103: 185–193.

21. Zhang Y, Xiong J, Li X, et al. A multi-replica associated deleting scheme in cloud. In: *Proceedings of the 10th international conference on complex, intelligent, and software intensive systems*, Fukuoka, Japan, 6–8 July 2016, pp.444–448. New York: IEEE.

22. Lai J, Xiong J, Wang C, et al. A secure cloud backup system with deduplication and assured deletion. In: *Proceedings of the international conference on provable security*, Xi’an, China, 23–25 October 2017, pp.74–83. Cham: Springer.

23. Wang M, Xiong J, Ma R, et al. A novel data secure deletion scheme for mobile devices. In: *Proceedings of the 27th international conference on computer communication and networks*, Hangzhou, China, 30 July–2 August 2018, pp.1–8. New York: IEEE.

24. Yang C, Tao X, Zhao F, et al. A new outsourced data deletion scheme with public verifiability. In: *Proceedings of the 14th international conference on wireless algorithms, systems, and applications*, Honolulu, HI, 24–26 June 2019, pp.631–638. Cham: Springer.

25. Garfinkel SL and Shelat A. Remembrance of data passed: a study of disk sanitization practices. *IEEE Secur Prat* 2003; 99: 17–27.

26. Gutmann P. Secure deletion of data from magnetic and solid-state memory. In: *Proceedings of the 6th conference on USENIX security symposium*, San Jose, CA, 22–25 July 1996, pp.77–89. USENIX Association.

27. Bauer S and Priyantha NB. Secure data deletion for Linux file systems. In: *Proceedings of the 10th conference on USENIX security symposium*, Washington, DC, 13–17 August 2001. USENIX Association.

28. Wei M, Grupp LM, Spada FE, et al. Reliably erasing data from flash-based solid state drives. In: *Proceedings of the 9th USENIX conference on file and storage technologies*, San Jose, CA, 15–17 February 2011, pp.105–117. USENIX Association.

29. Diesburg S, Meyers C, Stanovich M, et al. Trueerase: leveraging an auxiliary data path for per-file secure deletion. *ACM T Storage* 2016; 12(4): 181–1837.

30. Jia S, Xia L, Chen B, et al. NFPS: adding undetectable secure deletion to flash translation layer. In: *Proceedings of the 11th ACM on Asia conference on computer and communications security*, Xi’an, China, 30 May–2 June 2016, pp.305–315. New York: ACM.

31. Paul M and Saxena A. Proof of erasability for ensuring comprehensive data deletion in cloud computing. *Comm Com Inf Sci* 2010; 89: 340–348.

32. Luo Y, Xu M, Fu S, et al. Enabling assured deletion in the cloud storage by overwriting. In: *Proceedings of the 4th ACM international workshop on security in cloud computing*, Xi’an, China, 30 May–2 June 2016, pp.17–23. New York: ACM.
33. Boneh D and Lipton RJ. A revocable backup system. In: Proceedings of the 6th conference on USENIX security symposium, San Jose, CA, 22–25 July 1996, pp.91–96. USENIX Association.

34. Mo Z, Qiao Y and Chen S. Two-party fine-grained assured deletion of outsourced data in cloud system. In: Proceedings of the 2014 IEEE 34th international conference on distributed computing systems, Madrid, Spain, 30 June–3 July 2014, pp.308–331. New York: IEEE.

35. Du L, Zhang Z, Tan S, et al. An associated deletion scheme for multi-copy in cloud storage. In: Proceedings of the 18th international conference on algorithms and architectures for parallel processing (ICA3PP 2018), Guangzhou, China, 15–17 November 2018. Cham: Springer.

36. Yang C, Tao X and Zhao F. Publicly verifiable data transfer and deletion scheme for cloud storage. Int J Distrib Sens Netw 2019; 15(10): 1–12.

37. Yu Y, Xue L, Li Y, et al. Assured data deletion with fine-grained access control for fog-based industrial applications. IEEE T Ind Inform 2018; 14(10): 4538–4547.

38. Xue L, Yu Y, Li Y, et al. Efficient attribute-based encryption with attribute revocation for assured data deletion. Inform Sci 2019; 479: 640–650.

39. Yu Y, Ni J, Wu W, et al. Provable data possession supporting secure data transfer for cloud storage. In: Proceedings of the 10th international conference on broadband and wireless computing, communication and applications, Krakow, 4–6 November 2015, pp.38–42. New York: IEEE.

40. Xue L, Ni J, Li Y, et al. Provable data transfer from provable data possession and deletion in cloud storage. Comput Stand Inter 2017; 54: 46–54.

41. Liu Y, Xiao S, Wang H, et al. New provable data transfer from provable data possession and deletion for secure cloud storage. Int J Distrib Sens N 2019; 15(4): 1–12.

42. Yang C, Wang J, Tao X, et al. Publicly verifiable data transfer and deletion scheme for cloud storage. In: Proceedings of the 20th international conference on information and communications security, Lillé, 29–31 October 2018, pp.445–458. Cham: Springer.

43. Miao M, Ma J, Huang X, et al. Efficient verifiable databases with insertion/deletion operations from delegating polynomial functions. IEEE T Inf Foren Sec 2017; 13(2): 511–520.

44. Liu C, Ranjan R, Yang C, et al. MuR-DPA: top-down levelled multi-replica Merkle hash tree based secure public auditing for dynamic big data storage on cloud. IEEE T Comput 2014; 64(9): 2609–2622.

45. Evdokimov S and Günther O. Encryption techniques for secure database outsourcing. In: Proceedings of the 2007 European symposium on research in computer security, Dresden, 24–26 September 2007, pp.327–342. Berlin; Heidelberg: Springer.