IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received March 30, 2022, accepted April 20, 2022, date of publication April 22, 2022, date of current version April 29, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3169793

Privacy-Preserving Public Auditing for Shared Cloud Data With Secure Group Management

DONGMIN KIM[1] **AND KEE SUNG KIM**[2]
[1]Affiliated Institute of ETRI, Daejeon 34129, South Korea
[2]Daegu Catholic University, Daegu 38430, South Korea

Corresponding author: Kee Sung Kim (kee21@cu.ac.kr)

This work was supported by a grant from the National Research Foundation of Korea (NRF), funded by the Korean Government (MSIT), under Grant NRF-2019R1G1A1097540.

ABSTRACT With cloud storage services, users can store their data in the cloud and efficiently access the data at any time and any location. However, when data are stored in the cloud, there is a risk of data loss because users lose direct control over their data. To solve this problem, many cloud storage auditing techniques have been studied. In 2019, Tian *et al.* proposed a public auditing scheme for shared data that supports data privacy, identity traceability, and group dynamics. In this paper, we point out that their scheme is insecure against tag forgery or proof forgery attacks, which means that, even if the cloud server has deleted some outsourced data, it can still generate valid proof that the server had accurately stored the data. We then propose a new scheme that provides the same functionalities and is secure against the above attacks. Moreover, we compare the results with other schemes in terms of computation and communication costs.

INDEX TERMS Public auditing, privacy-preserving, shared data, group management.

I. INTRODUCTION

Cloud storage provides users with significant storage capacity and advantages such as a cost reduction, scalability, and convenient access to the stored data. Therefore, cloud storage that is managed and maintained by professional cloud service providers (CSPs) is widely used by many enterprises and personal clients [1]. Once the data are stored in cloud storage, the clients lose direct control over the stored files. Despite this, the CSPs must ensure that the client data are placed in cloud storage without any modification or substitution. The simplest way to achieve this is by checking the integrity of the stored data after downloading. When the capacity of the stored data is large, it is quite inefficient, and thus many methods for verifying the integrity of the data stored in the cloud without a full download have been proposed [2]–[34].

These techniques are called cloud storage auditing and can be classified into private auditing and public auditing according to the subject of the integrity verification. In private auditing, verification is achieved by users who have ownership of the stored data. Public auditing is conducted by a third-party auditor (TPA) on behalf of the users to reduce their burden, and thus public auditing schemes are more widely employed for cloud storage auditing. Public auditing schemes provide various properties depending on the environment, such as privacy preservation [5]–[9], data dynamics [10]–[13], and shared data [14]–[33]. Privacy-preserving auditing is used to conduct an integrity verification while protecting data information from the TPA, and dynamic data auditing is where legitimate users are free to add, delete, or change the stored data. Shared data auditing means freely sharing data within a legitimate user group. In this case, a legitimate user group should be defined, and user addition and revocation should be carefully considered. Recently, schemes that satisfy identity traceability, a concept that can trace the abnormal behavior of legitimate users in shared data auditing, have also been proposed.

Tian *et al.* [25] proposed a scheme that supports privacy preservation, data dynamics, and identity traceability in shared data auditing. For efficient user enrollment and revocation, the authors adopted the lazy revocation technique. Moreover, to secure the design against collusion attacks between the revoked user and server, they apply a technique in which the group manager manages messages and tag blocks generated by the revoked user to the scheme. Because the lazy-revocation technique is applied to the scheme, even if a user is revoked, no additional operation occurs until additional changes are made to the block.

In this paper, we show that Tian *et al.*'s scheme [25] is insecure against two types of attacks, a tag forgery and a proof forgery, and proposed a new scheme that provides the same functionality and is secure against the above attacks. In this scheme, a tag forgery is possible by exploiting the vulnerability in which the tag is created in a malleable way, and a

The associate editor coordinating the review of this manuscript and approving it for publication was Junggab Son.

proof forgery is possible by exploiting the secret value being exposed to the server when additional changes to the block occur after the user is revoked. In general, the contributions of this study can be summarized as follows:

1. We show that Tian *et al.*'s scheme [25] is insecure against two types of attacks: tag and proof forgeries. In tag forgery, we show that an attacker can create a valid tag for the modified message without knowing any secret values. In the proof forgery, we show that an attacker can create a valid proof for the given challenged message even if some files stored on the cloud have been deleted.
2. We design a new public auditing scheme that is secure against the above attacks and has the same functionalities, such as privacy preservation, data dynamics, data sharing, and identity traceability. We changed the tag generation method to eliminate the malleable property and the data proof generation method to enhance the privacy preservation. We also changed the lazy revocation process to protect the secret information from the CSP and proposed an active revocation process to flexibly apply the various environments.
3. We formally prove the security of the proposed scheme. According to the theorems, the attacker cannot generate a valid tag and proof without knowing the secret values or the original messages, respectively. We also provide comparison results with other schemes in terms of the computation and communication costs.

The rest of this paper is organized as follows. In Section II, we introduce the background, and in Section III, we review Tian *et al.*'s scheme [25]. We present our detailed scheme for public auditing in Section IV, and provide the security and efficiency of our scheme in Section V. Finally, we conclude this paper in Section VI.

A. RELATED WORK

Ateniese *et al.* [2] first introduced a provable data possession scheme called PDP and provided two provably secure PDP schemes using RSA-based homomorphic authenticators. This supports public verification with lower communication and computation costs. At the same time, Juels *et al.* [3] first proposed the concept and a formal security model of proof of retrievability (POR) and a sentinel-based POR scheme with certain properties. Later, Shacham *et al.* [4] improved the POR scheme and proposed a new public auditing scheme that was built from the BLS signature [36] and is secure in the random oracle model. In recent years, many studies have been conducted on cloud storage auditing, supporting various functionalities such as data privacy preservation, data dynamics, and shared data.

Erway *et al.* [10] first proposed the PDP scheme using a rank-based authenticated skip list to support data dynamics. However, the scheme suffers from high computational and communication costs, and to address this concern, Wang *et al.* [11] proposed a new auditing scheme employing the Merkle Hash Tree (MHT), which is much simpler. However, this scheme does not provide data privacy; in other words, it cannot protect data confidentiality from a third-party auditor (TPA). Although Wang *et al.* [5] proposed a privacy-preserving public auditing scheme, their approach requires heavy communication and computation costs in the audit and data update process. Zhu *et al.* [12] also proposed a new scheme using another authenticated data structure, called an index hash table (IHT), to support data dynamics. Although this scheme succeeded in reducing the communication and computation costs, it did not resolve the inefficient problem of lookup and updating operations. Shen *et al.* [13] proposed a new efficient scheme with a doubly linked information table and location array. Tian *et al.* [25] recently proposed a more efficient scheme using a dynamic hash table (DHT), which has been proven to be more effective than IHT for data updating [12]. In terms of data privacy, Wang *et al.* [5] first proposed a privacy-preserving public auditing scheme to protect data privacy through random masking, and many schemes for predicting data privacy have been studied [6]–[9]. As a follow-up, many studies that satisfy additional properties and data privacy have been proposed. In particular, many studies have been conducted on public auditing schemes for shared data. Wang *et al.* [14] proposed an efficient public auditing scheme called Knox for shared data. The scheme supports hiding the identity of individual users based on a group signature, but does not support a user revocation. In Oruta [15], a ring signature is used to hide the identity of individual users; however, the scheme also has a problem in that all user keys and block tags must be regenerated to provide a user revocation. Wang *et al.* [16] also proposed a scheme that can achieve a user revocation using a proxy re-signature. The scheme utilizes a proxy for a resigning used to update the tag generated by the revoked user; however, it is vulnerable to collusion attacks between an invalid user and the cloud server. In addition, Jiang *et al.* [17] proposed a new public auditing scheme that combines a vector commitment [37] and a verifier-local-revocation group signature [38]. During the revocation phase, the computational costs are relatively high because it is necessary to first find the tags generated by the revoked user and regenerate them. Yu *et al.* [18], [19] also proposed a new scheme using polynomial authentication tags and proxy re-signatures. Although this scheme can reduce the communication overhead during verification, it can suffer from a collusion attack because the revoked user still has a valid private key and might collude with the CSP. Subsequently, another cloud storage auditing scheme for data sharing based on group signatures was proposed [20]–[33]. Wu *et al.* [24] proposed an efficient threshold privacy-preserving cloud storage auditing scheme, which does not rely on a group or ring signature; therefore, it is more efficient than other schemes using heavy cryptographic primitives. Chang *et al.* [26] also proposed a new scheme with an oblivious transfer and stateless lazy re-encryption to provide an efficient user revocation. Most of these studies have a problem in that it is possible to access sensitive informa-

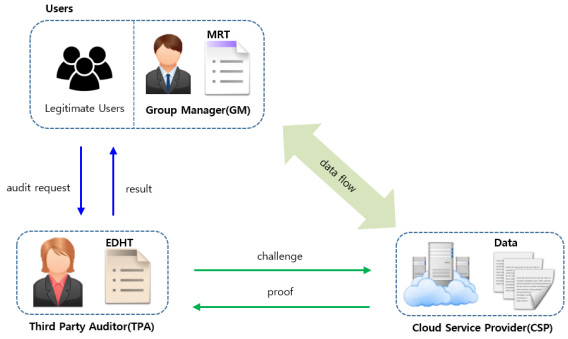


FIGURE 1. System Model.

tion of the data owner. To address this, Shen *et al.* [22] proposed a practical identity-based shared data integrity auditing scheme using a sanitizable signature [39]. However, it can be vulnerable because the scheme needs a secure channel between the data owner and the sanitizer, and any user can access the shared data. Xu *et al.* [27] proposed a PP-CSA scheme for data sharing, where only the authorized user can access the data, and there is no need to establish a secure channel between the data owner and the sanitizer. These schemes provide a function to hide sensitive information but have limitations in not providing user revocation or data dynamics. In addition, the integrity auditing scheme [35] using keywords in encrypted data and a more secure auditing scheme [34] providing forward security have been recently proposed, but these schemes are not suitable for our target environment. Many schemes have been proposed, but none of these schemes satisfy the above-mentioned security except for the scheme proposed by Tian *et al.* [25]. We point out the security issue of Tian *et al.*'s scheme [25] and propose an improved scheme.

II. BACKGROUND

A. SYSTEM MODEL

The system model of our scheme consists of a cloud service provider (CSP); a group of users, who can be divided into legitimate users and a group manager (GM); and a third-party auditor (TPA). The CSP manages many cloud servers and provides cloud storage services to a group of users. A legitimate user can access and modify all shared data. They can generate a block tag for a modified data block. The GM manages a group of users, including user enrollment, revocation, and tracing user operations. The TPA conducts an integrity check for shared data on behalf of the group of users.

B. COMPLEXITY ASSUMPTIONS

COMPUTATIONAL DIFFIE-HELLMAN (CDH) ASSUMPTION. An algorithm $\mathcal{A}$ is given a tuple $g, g^a, g^b \in \mathbb{G}$ and tries to find $g^{ab} \in \mathbb{G}$. We assume that there is no algorithm $\mathcal{A}$ with a non-negligible probability ε such that

$$Pr[\mathcal{A}(g, g^a, g^b) = g^{ab}] \geq \varepsilon,$$

where the probability is over the random choice of $g \in \mathbb{G}$, the random choice of $a, b \in \mathbb{Z}_p^*$, and the random bits of $\mathcal{A}$.

C. REVOCATION

We can consider two types of user revocation: active and lazy revocations. An *active revocation* means that when a certain user is revoked, the user's revocation process is applied by immediately updating information related to the user. A *lazy revocation* refers to a method of waiting for other legitimate users to update the related information without immediately updating information related to that user, even if a user is revoked. This method is more efficient than an active revocation because it does not need to apply the revocation process whenever the user is revoked. In this study, we designed both revocation methods such that the revocation method can be flexibly applied to various environments.

D. EXTENDED DYNAMIC HASH TABLE

To manage the data blocks handled by revoked users, we use an extended dynamic hash table (EDHT) [25]. The table consists of the index of the file (No), the identifier (ID) of file F, and a pointer indicating its first block element, and all block elements are implemented using a linked list. Each block element contains five fields, namely, the current version of the block $v_{i,j}$, its time stamp $t_{i,j}$, a revocation flag, a revocation parameter $\lambda_{i,k}$, and a pointer indicating the next node. The revocation flag can be recorded as true of false. If true, it means that the block was most recently changed by a revoked user; otherwise, it means that the data block was changed by a legitimate user. The revocation parameter is recorded only when the revocation flag is true, and the value is generated by the group manager during the revocation process.

E. MODIFICATION RECORD TABLE

The modification record table (MRT) is a table in which the group manager records operations for each block to provide identity traceability and is a two-dimensional data structure [25]. It records all modified blocks and related operation information by managing two types of data: block items and operation items. All block items are implemented using a linked list, and each item contains its block identifier b_i, a pointer indicating the next block, and a pointer indicating the first operation item. All operation items are implemented using a linked stack in which the first recorded operation is the latest operation, and each item contains the identifier $u_{i,j}$ of the user who modified the data block, the operation $op_{i,j}$, and the timestamp $t_{i,j}$. The MRT is maintained by the group manager, and it should be updated when a user modifies some data blocks.

III. REVIEW TIAN *et al.*'s SCHEME

A. TIAN *et al.*'s SCHEME

Let groups $\mathbb{G}_1$ and $\mathbb{G}_T$ be multiplicative cyclic groups of a large prime order p, i.e., g and g_1, and μ be the generators of group $\mathbb{G}_1$. A bilinear map is defined as $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_T$. Here, H is a secure hash function such that $H : \{0, 1\}^* \rightarrow \mathbb{Z}_p^*$, and $F = \{m_1, \ldots, m_n\}$ is the shared data file divided into

TABLE 1. Definitions and notations.

Symbol	Definition
$\mathbb{G}_1, \mathbb{G}_T$	multiplicative cyclic groups of a large prime order p
H	secure hash function such that $H : \{0,1\}^* \rightarrow \mathbb{Z}_p^*$
H_1	secure hash function such that $H_1 : \{0,1\}^* \rightarrow \mathbb{G}_1$
H_2	secure hash function such that $H_2 : \mathbb{G}_T \rightarrow \mathbb{Z}_p^*$
e	bilinear map such that $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_T$
U	set of all users in group
U_R	set of revoked users
U_L	set of legitimate users
F	shared data file divided into n blocks
sk_i, pk_i	secret key/public key pair of the user u_i
σ_i	tag for the data block m_i
ID	unique file identifier
δ	file tag of the file
$RU-blocks$	data blocks generated by the revoked users
IDX	index set of all challenged blocks
S	set of random numbers for the challenged blocks
Φ	tag proof for the challenged blocks
T	tag proof of RU-blocks
Λ	post-revocation authenticator
Ω_1, ν_1	data proof for the challenge blocks
Ω_2, ν_2	data proof of RU-blocks
$\Theta, \Theta_1, \Theta_2$	auxiliary proof factor
λ_i	revocation parameter of the i-th block
Ψ	revocation message from the group manager to revoke a user

n blocks. In the data sharing group $U = \{u_1, u_2, \ldots, u_w\}$, there are w users, where u_1 is the data owner and also acts as the group manager, with the others being valid group users. Tian *et al.*'s scheme [25] consists of five algorithms: **KeyGen**, **TagGen**, **Challenge**, **ProofGen**, and **Verify**.

KeyGen. The group manager randomly chooses $a_1 \leftarrow \mathbb{Z}_p^*$ and calculates $y_1 = g^{a_1}$ as his secret/public key pair and randomly chooses $a_i \leftarrow \mathbb{Z}_p^*$ as u_i's secret key and calculates $y_i = {y_1}^{1/a_i}$ as u_i's public key.

TagGen. For each block m_i, user u_j generates a signature σ_i on his secret key $sk_j = a_j$ as

$$\sigma_i = (\mu^{H(v_i||t_i)} \cdot {g_1}^{m_i})^{a_j}, \tag{1}$$

where v_i is the version number, and t_i is the time stamp of block m_i. User u_j uploads $\{m_i, \sigma_i\}$ to the CSP, and records additional information to the EDHT. When a data block is modified, the group manager adds the operation information into the MRT and asks the TPA to update the additional information of the block in the EDHT. The group manager also computes a file tag $\delta = ID||SIG(sk_1, ID)$ to ensure the integrity of the ID, where $SIG(sk_1, ID)$ is the signature on ID under the private key sk_1 and sends it to the CSP.

Challenge. The TPA first retrieves the file tag δ and verifies the signature $SIG(sk_1, ID)$ using the public key pk_1 of the group manager. If the verification fails, the TPA directly quits the verification by emitting a FALSE; otherwise, it generates challenge information as follows: (1) The TPA selects c random data blocks, which are not modified by the revoked users, and generates a challenge block set $IDX = \{idx_i | 1 \leq i \leq c, c \leq n\}$ and chooses a set $S = \{s_i | i \in IDX\}$, where $s_i \in \mathbb{Z}_p^*$. (2) Let RU-blocks be the data blocks modified by the revoked users, and D be the index set of all the RU-blocks. For each $i \in D$, the TPA calculates $\beta_i = \lambda_i / \eta$, where η is a random number, and λ_i is the revocation parameter of the i-th block. (3) Letting the set of all legal users in the group be U_L and the set of all revoked users in the group be U_R for each $i \in U_L$, the TPA computes $\gamma_i = {y_i}^w = g^{w \cdot a_1 / a_i}$, and for each $i \in U_R$, it computes $\tau_i = {y_i}^{w/\epsilon} = g^{w/\epsilon \cdot a_1 / a_i}$, where $\epsilon \in \mathbb{Z}_p^*$ and $w \in \mathbb{Z}_p^*$ are two random numbers. The TPA then sends the challenge values $chal = (IDX, S, \{\beta_i | i \in D\}, \{\gamma_i | i \in U_L\}, \{\tau_i | i \in U_R\})$ to the CSP.

ProofGen. After receiving the challenge message $chal = (IDX, S, \{\beta_i | i \in D\}, \{\gamma_i | i \in U_L\}, \{\tau_i | i \in U_R\})$, the CSP provides proof for the challenged blocks. For the challenged blocks, the CSP calculates the tag proof Φ as

$$\Phi = \prod_{i \in IDX} e(\sigma_i, \gamma_j)^{s_i}, \tag{2}$$

where $j \in U_L$, and the i-th block is last modified by the jth user. The CSP then calculates the data proof Ω_1 as

$$\Omega_1 = \sum_{i \in IDX} m_i \cdot s_i + r, \tag{3}$$

where $r \in \mathbb{Z}_p^*$ is a random number that is used to protect data privacy against TPA.

For all RU-blocks, the CSP calculates the tag proof T as follows:

$$T = \prod_{i \in D} e(\sigma_i, \tau_j)^{\beta_i}, \tag{4}$$

where $j \in U_R$, and the i-th block is last modified by the jth user. The CSP then calculates the data proof Ω_2 as

$$\Omega_2 = \sum_{i \in D} m_i \cdot \beta_i + r. \tag{5}$$

During a user revocation, the group manager produces the post-revocation authenticator Λ, which is maintained by the CSP. Moreover, the CSP computes the auxiliary auditing factor as

$$\Theta = e(g_1, y_1)^{-r}. \tag{6}$$

Finally, the CSP sends the proof $P = \{\Phi, T, \Omega_1, \Omega_2, \Lambda, \Theta\}$ to the TPA.

Verify. With the proof that $P = \{\Phi, T, \Omega_1, \Omega_2, \Lambda, \Theta\}$, the TPA verifies the integrity of the chosen blocks. If there is no revoked user or all RU-blocks have been modified by the other users in U_L, the TPA can verify the correctness of the data file; otherwise, the TPA verifies the post-revocation authenticator by computing the following equation:

$$\Lambda = T^{\eta \cdot \epsilon / w}. \tag{7}$$

If (7) does not hold, the TPA outputs *Reject*; otherwise, the TPA can be verified as

$$\Phi \cdot T^{\eta \cdot \epsilon} = e(\mu^{\Delta B} \cdot {g_1}^{\Omega}, y_1)^w \cdot \Theta^{w \cdot (\eta + 1)}, \tag{8}$$

where $B_i = H(v_i||t_i)$, $\Delta B = \prod_{i \in IDX} B_i s_i + \prod_{k \in D} B_k \lambda_k$, and $\Omega = \Omega_1 + \Omega_2 \cdot \eta$. If (8) holds, the TPA outputs *Accept*; otherwise, it outputs *Reject*.

Revocation. When user u_x needs to be revoked, the group manager sends revocation messages to the CSP as follows:

$$\Psi = \{y_x^q, \{\theta_i | i \in D\}\}, \tag{9}$$

where D is the index set of the blocks modified by the revoked user u_x, and $q \in \mathbb{Z}_p^*$ and $\theta_i \in \mathbb{Z}_p^* (i \in D)$ are random numbers. Upon receiving the revocation message, the CSP aggregates the tags corresponding to index set D as

$$\Lambda' = \prod_{i \in D} e(\sigma_i, y_x^q)^{\theta_i}, \tag{10}$$

and sends Λ' to the group manager. The group manager then generates the post-revocation authenticator Λ and its parameter λ_i as

$$\Lambda = \Lambda'^{\rho}, \tag{11}$$

$$\lambda_i = q \cdot \rho \cdot \theta_i, i \in D, \tag{12}$$

where ρ is a random number. Finally, the group manager sends $(R, \{\lambda_i | i \in D\})$, where R is the index set of the RU-blocks to the TPA.

When a data block m_i handled by the revoked user u_x is modified by another legal user, the CSP first retrieves the revocation parameter λ_i from the TPA and updates the post-revocation authenticator as

$$\Lambda = \Lambda / e(\sigma_i, y_x)^{\lambda_i}. \tag{13}$$

B. TAG FORGERY ATTACK

In Tian *et al.*'s scheme [25], they provide the security analysis for the forging attacks, but it does not include our attacks because the tag generation method using a homomorphic verifiable authenticator (HVA) is not well designed. They claimed their tag generation method satisfies the non-malleability property, but the tag can be forged by malleable property because the same base μ is used for each message when generating a tag (HVA). We show that it is possible to generate valid tags for the modified messages using valid messages and tag pairs with publicly known values in Tian *et al.*'s scheme [25].

We assume that σ_1 and σ_2 are the valid tags generated by the data owner u_1 for the messages m_1 and m_2, respectively, where $\sigma_1 = (\mu^{H(v_1||t_1)} \cdot g_1^{m_1})^{a_1}$ and $\sigma_2 = (\mu^{H(v_2||t_2)} \cdot g_1^{m_2})^{a_1}$. With the public parameters (μ, g_1), (v_i, t_i) values in the EDHT, and the original messages m_1, m_2, the server computes

$$\begin{aligned}
\frac{\sigma_1}{\sigma_2} &= \mu^{\Delta H \cdot a_1} \cdot g_1^{(m_1 - m_2) \cdot a_1}, \\
(\frac{\sigma_1}{\sigma_2})^{\frac{1}{\Delta H}} &= \mu^{a_1} \cdot g_1^{(\frac{m_1 - m_2}{\Delta H}) \cdot a_1}, \\
(\frac{\sigma_1}{\sigma_2})^{\frac{H(v_1||t_1)}{\Delta H}} &= \mu^{H(v_1||t_1) \cdot a_1} \cdot g_1^{H(v_1||t_1) \cdot (\frac{m_1 - m_2}{\Delta H}) a_1}, \\
\sigma_1^* &= \mu^{H(v_1||t_1) \cdot a_1} \cdot g_1^{m_1^* \cdot a_1}, \\
\sigma_1^* &= (\mu^{H(v_1||t_1)} \cdot g_1^{m_1^*})^{a_1},
\end{aligned}$$

where $\Delta H = H(v_1||t_1) - H(v_2||t_2)$ and $m_1^* = H(v_1||t_1) \cdot (\frac{m_1 - m_2}{\Delta H})$. The value σ_1^* is then a valid tag for the modified message m_1^*. We show that a valid tag can be generated without changing any values in the public tables and knowing the secret value a_1. Note that the TPA verifies the proof of challenge messages with publicly known values in *public auditing*. Tian *et al.*'s scheme [25] satisfies public auditing, and the TPA in their scheme applies the verification stage using the values v_i, t_i; therefore, these values are considered as public values but cannot be revised without the permission of the group manager.

C. PROOF OF FORGERY ATTACK

In Tian *et al.*'s scheme [25], they prove that each elements in proof P was unforgeable. However, they omit some of the information that the attacker can obtain and prove under false assumption that Θ is unforgeable without any proof. In our attack, we show that the attacker can make a forged proof that passes the verification even when some data are deleted.

When a data block m_i generated by a revoked user u_x is modified by another user, the CSP first retrieves the revocation parameter λ_i from the TPA and updates the post-authenticator as

$$\Lambda = \Lambda / e(\sigma_i, y_x)^{\lambda_i}. \tag{14}$$

The revocation parameter λ_i is generated by the group manager and is managed by the TPA. For the security of the scheme, λ_i should not be known to the CSP; however, as mentioned above, λ_i is exposed to the CSP to update the post-authenticator. The malicious CSP can extract the random number η through λ_i and challenge values β_i. Using η, the malicious CSP can generate a valid proof that passes the verification even when some data are deleted.

In the following, we describe a proof forgery attack on the scheme in [25], where a malicious CSP can collude with a revoked user to make a valid proof given the challenge message even if some data stored on the cloud have been deleted.

Given the challenge information $chal = (IDX, S, \{\beta_i | i \in D\}, \{\gamma_i | i \in U_L\}, \{\tau_i | i \in U_R\})$, we assume that $D = D_d \cup D_o$, where D_d is a block set deleted by the CSP, D_o is a block set stored on the cloud, and $IDX = IDX_d \cup IDX_o$, where IDX_d is a block set deleted by the CSP, and IDX_o is a block set stored on the cloud.

- After receiving the challenge information $chal = (IDX, S, \{\beta_i | i \in D\}, \{\gamma_i | i \in U_L\}, \{\tau_i | i \in U_R\})$, the CSP computes $\eta = {\beta_i}^{-1} \cdot \lambda_i$.
- Using η, the CSP computes $\lambda_i = \beta_i \cdot \eta$ for all $i \in D$.
- For the challenged blocks, the CSP generates the tag proof $\Phi = \prod_{i \in IDX_o} (e(\sigma_i, \gamma_j))^{s_i}$ and the data proof $\Omega_1 = \sum_{i \in IDX_o} m_i \cdot s_i + r$. The CSP also generates a tag proof for the RU-blocks $T^* = \prod_{i \in D_o} e(\sigma_i, \tau_j)^{\beta_i}$ and the data proof $\Omega_2 = \sum_{i \in D_o} m_i \cdot \beta_i + r$.
- The CSP calculates the auxiliary auditing factor as

$$\Theta^* = e(g_1, y_1)^{-r} \cdot e(\mu^{-\Delta B_d}, y_1)^{\frac{1}{\eta+1}}, \tag{15}$$

where $\Delta B_d = \sum_{i \in IDX_d} B_i s_i + \sum_{i \in D_d} B_i \lambda_i$.

Then, the forged proof $P^* = \{\Phi, T^*, \Omega_1, \Omega_2, \Lambda, \Theta^*\}$ is a valid proof for the challenge information *chal* because it can pass the **Verify** algorithm as follows:

$$\Phi \cdot T^{*\eta\cdot\epsilon} = e(\mu^{\Delta B} \cdot {g_1}^{\Omega}, y_1)^w \cdot \Theta^{*w\cdot(\eta+1)}, \tag{16}$$

where $\Delta B = \sum_{i\in IDX} B_i s_i + \sum_{k\in D} B_k \lambda_k$ and $\Delta B_o = \Delta B - \Delta B_d = \sum_{i\in IDX_o} B_i s_i + \sum_{k\in D_o} B_k \lambda_k$.

The left side of (16) can be written as follows:

$$\begin{aligned}
&\Phi \cdot T^{*\eta\cdot\epsilon} \\
&= \prod_{i\in IDX_o} e(\sigma_i, \gamma_j)^{s_i} \cdot \prod_{k\in D_o} e(\sigma_k, \tau_j)^{\beta_k\cdot\eta\cdot\epsilon}, \\
&= \prod_{i\in IDX_o} e((\mu^{B_i} \cdot g_1^{m_i})^{a_j}, g^{w\cdot a_1/a_j})^{s_i} \\
&\quad \cdot \prod_{k\in D_o} e((\mu^{B_k} \cdot g_1^{m_k})^{a_j}, g^{w/\epsilon\cdot a_1/a_j})^{\lambda_k\cdot\epsilon} \\
&= \prod_{i\in IDX_o} e(\mu^{B_i} \cdot g_1^{m_i}, y_1)^{ws_i} \cdot \prod_{k\in D_o} e(\mu^{B_k} \cdot g_1^{m_k}, y_1)^{\lambda_k w} \\
&= e(\prod_{i\in IDX_o} \mu^{B_i s_i} \cdot g_1^{\sum_{i\in IDX_o} m_i s_i}, y_1)^w \\
&\quad \cdot e(\prod_{k\in D_o} \mu^{B_k\lambda_k} \cdot g_1^{\sum_{k\in D_o} m_k\lambda_k}, y_1)^w \\
&= e(\mu^{\sum_{i\in IDX_o} B_i s_i} \cdot g_1^{\Omega_1 - r}, y_1)^w \\
&\quad \cdot e(\mu^{\sum_{k\in D_o} B_k\lambda_k} \cdot g_1^{(\Omega_2 - r)\cdot\eta}, y_1)^w \\
&= e(\mu^{\sum_{i\in IDX_o} B_i s_i + \sum_{k\in D_o} B_k\lambda_k} \cdot g_1^{\Omega_1 - r + (\Omega_2 - r)\cdot\eta}, y_1)^w \\
&= e(\mu^{\sum_{i\in IDX_o} B_i s_i + \sum_{k\in D_o} B_k\lambda_k} \cdot g_1^{\Omega_1 + \Omega_2\cdot\eta - (1+\eta)r}, y_1)^w \\
&= e(\mu^{\sum_{i\in IDX_o} B_i s_i + \sum_{k\in D_o} B_k\lambda_k} \cdot g_1^{\Omega}, y_1)^w \\
&\quad \cdot e(g_1, y_1)^{-w(\eta+1)r} \\
&= e(\mu^{\Delta B_o} \cdot g_1^{\Omega}, y_1)^w \cdot e(g_1, y_1)^{-w(\eta+1)r}.
\end{aligned}$$

The right side of (16) can be written as

$$\begin{aligned}
&e(\mu^{\Delta B} \cdot g_1^{\Omega}, y_1)^w \cdot \Theta^{*w(\eta+1)} \\
&\quad = e(\mu^{\Delta B_o + \Delta B_d} \cdot g_1^{\Omega}, y_1)^w \cdot \Theta^{*w(\eta+1)} \\
&\quad = e(\mu^{\Delta B_o + \Delta B_d} \cdot g_1^{\Omega}, y_1)^w \\
&\qquad \cdot (e(g_1, y_1)^{-r} \cdot e(\mu^{-\Delta B_d}, y_1)^{\frac{1}{\eta+1}})^{w(\eta+1)} \\
&\quad = e(\mu^{\Delta B_o + \Delta B_d} \cdot g_1^{\Omega}, y_1)^w \cdot e(g_1, y_1)^{-r(\eta+1)w} \\
&\qquad \cdot e(\mu^{-\Delta B_d}, y_1)^w \\
&\quad = e(\mu^{\Delta B_o} \cdot g_1^{\Omega}, y_1)^w \cdot e(g_1, y_1)^{-r(\eta+1)w}.
\end{aligned}$$

Therefore, the forged proof $P^* = \{\Phi, T^*, \Omega_1, \Omega_2, \Lambda, \Theta^*\}$ can pass the verification algorithm. In other words, the cloud server can deceive the TPA to believe that the data have been completely stored even though they have been deleted.

IV. OUR PROPOSED SCHEME

Let groups $\mathbb{G}_1$ and $\mathbb{G}_T$ be multiplicative cyclic groups with prime order p, and let $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_T$ be a bilinear map. In addition, g and u are the generators of $\mathbb{G}_1$. Moreover, $H_1 : \{0,1\}^* \to \mathbb{G}_1$ and $H_2 : \mathbb{G}_T \to \mathbb{Z}_p^*$ are collision-resistant hash functions. Assume that a file F is divided into n data blocks $F = \{m_1, \ldots, m_n\}$ and there are w users in the user group $U = \{u_1, u_2, \ldots, u_w\}$, where the user u_1 acts as a group manager (GM) and the others are common users. Let U_L be the set of all legal users, and U_R be the set of all revoked users in the group. Our proposed scheme consists of six probabilistic polynomial-time algorithms: **KeyGen**, **TagGen**, **Update**, **Challenge**, **Prove**, **Verify**, and **Revocation**.

KeyGen. The group manager randomly chooses $a_1 \leftarrow \mathbb{Z}_p^*$ and calculates $y_1 = g^{a_1}$ as the manager's secret/public key pair and randomly chooses $a_i \leftarrow \mathbb{Z}_p^*$ as the secret key of u_i and calculates $y_i = {y_1}^{1/a_i}$ as the public key of u_i.

TagGen. User u_j generates an authenticated tag σ_i on the message block m_i of F as

$$\sigma_i = (h_{i,j} \cdot u^{m_i})^{a_j}, \tag{17}$$

where $h_{i,j} = H_1(ID||i||j||v_i||t_i)$, ID is the file identifier of F, v_i is the version number, and t_i is the time stamp. User u_j uploads $\{m_i, \sigma_i\}$ to the CSP, and records additional information to the EDHT. When a data block is modified, the group manager adds the operation information into the MRT and asks the TPA to update the additional information of the block in the EDHT.

Update. Suppose that the valid group user u_k modifies the i-th block m_i to m_i'. Then, u_k computes the authenticated tag σ_i' on the modified block m_i' as

$$\sigma_i' = (h_{i,k} \cdot u^{m_i'})^{a_k}, \tag{18}$$

where $h_{i,k} = H_1(ID||i||k||v_i'||t_i')$, and v_i' and t_i' denote the updated version number and time stamp, respectively. User u_k uploads (m_i', σ_i') to the CSP and asks the group manager to update the additional information of the block in the EDHT and MRT.

Challenge. Because the challenge process is the same as that of Tian *et al.*'s scheme [25], a detailed description is omitted here.

Prove. After receiving the challenge message $chal = (IDX, S, \{\beta_i | i \in D\}, \{\gamma_i | i \in U_L\}, \{\tau_i | i \in U_R\})$, the CSP provides proof for the challenged blocks. For the challenge blocks, the CSP calculates the tag proof Φ as

$$\Phi = \prod_{i\in IDX} e(\sigma_i, \gamma_j)^{s_i}, \tag{19}$$

where $j \in U_L$ and the i-th block is modified by the j-th user. The CSP then calculates the data proof ν_1 as

$$\nu_1 = \sum_{i\in IDX} m_i \cdot s_i \cdot H_2(\Theta_1) + r_1, \tag{20}$$

where $r_1 \in \mathbb{Z}_p^*$ is a random number, and $\Theta_1 = e(u, y_1)^{r_1}$. These values are used to protect data privacy against the TPA.

For all RU-blocks, the CSP calculates the tag proof T as follows:

$$T = \prod_{i\in D} (e(\sigma_i, \tau_j))^{\beta_i}, \tag{21}$$

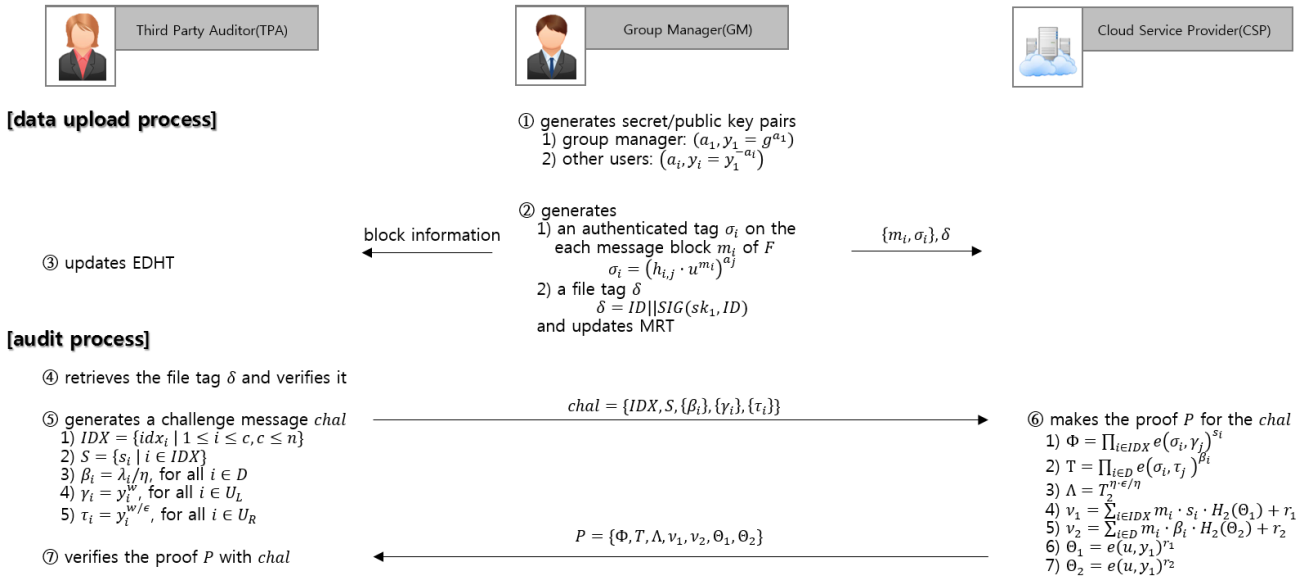


FIGURE 2. Data upload and audit process.

where $j \in U_R$ and the i-th block is modified by the j-th user. The CSP then calculates the data proof ν_2 as

$$\nu_2 = \sum_{i \in D} m_i \cdot \beta_i \cdot H_2(\Theta_2) + r_2, \tag{22}$$

where $r_2 \in \mathbb{Z}_p^*$ is a random number, and $\Theta_2 = e(u, y_1)^{r_2}$. These values are used to protect data privacy against the TPA.

Finally, the CSP sends the proof $P = \{\Phi, T, \Lambda, \nu_1, \nu_2, \Theta_1, \Theta_2\}$ to the TPA.

Verify. With the proof that $P = \{\Phi, T, \Lambda, \nu_1, \nu_2, \Theta_1, \Theta_2\}$, the TPA verifies the integrity of the chosen blocks. If there is no revoked user or all RU-blocks have been modified by the other users in U_L, the TPA can then verify the correctness of the data file; otherwise, the TPA verifies the post-revocation authenticator by computing the following equation:

$$\Lambda = T^{\eta \cdot \epsilon/w}. \tag{23}$$

If (23) does not hold, the TPA outputs *Reject*; otherwise, it can be verified as

$$\Theta_1{}^w \cdot \Phi^{H_2(\Theta_1)} = e(\prod_{i \in IDX} h_{i,j}^{s_i \cdot H_2(\Theta_1)} \cdot u^{\nu_1}, y_1)^w, \tag{24}$$

$$\Theta_2{}^{\eta \cdot w} \cdot T^{\eta \cdot \epsilon \cdot H_2(\Theta_2)} = e(\prod_{i \in D} h_{i,j}^{\lambda_i \cdot H_2(\Theta_2)} \cdot u^{\nu_2 \cdot \eta}, y_1)^w, \tag{25}$$

where $h_{i,j} = H_1(ID||i||j||v_i||t_i)$. If (24) and (25) hold, the TPA outputs *Accept*; otherwise, it outputs *Reject*.

Revocation. Our scheme provides two types of revocation: lazy revocation (LR) and active revocation (AR). With the LR method, when user u_k needs to be revoked, the group manager sends a revocation message to the CSP as follows:

$$\Psi = \{y_k{}^q, \{\theta_i | i \in D\}\}, \tag{26}$$

where D is the index set of the blocks modified by the revoked user u_k, and $q \in \mathbb{Z}_q^*$ and $\theta_i \in \mathbb{Z}_q^* (i \in D)$ are random numbers. Upon receiving the revocation message, the CSP aggregates the tags corresponding to index set D as

$$\Lambda' = \prod_{i \in D} e(\sigma_i, y_k^q)^{\theta_i}, \tag{27}$$

and sends Λ' to the group manager. The group manager then generates the post-revocation authenticator Λ and its parameter λ_i as

$$\Lambda = \Lambda'^{\rho}, \tag{28}$$

$$\lambda_i = q \cdot \rho \cdot \theta_i, i \in D, \tag{29}$$

where ρ is a random number. Finally, the group manager sends $(R, \{\lambda_i | i \in D\})$, where R is the index set of the RU-blocks to the TPA.

When a data block m_i modified by the revoked user u_k is modified by another legal user, the TPA updates the post-revocation authenticator as

$$\Lambda = \Lambda / e(\sigma_i, y_k)^{\lambda_i}. \tag{30}$$

Finally, the TPA sends the post-authenticator Λ to the CSP.

With the AR method, when user u_k needs to be revoked, the process of creating the post-revocation authenticator Λ is the same as in the LR method, but the subsequent process is different. First, the group manager chooses the revocation parameters z_{0i}, z_{1i} for all $i \in D$, computes the revocation factor z_i as $z_i = z_{0i}/z_{1i}$, and updates the public key y_k of user u_k as $y_k^{z_{1i}}$. The group manager also generates the challenge message *chal* and sends all revocation factors z_i with the challenge message *chal* to the CSP. Then, the CSP re-computes the tag for the data block handled by the revoked user $\sigma_i = \sigma_i^{z_i}$ and the proof P for the challenge message *chal*, and sends the proof P to the group manager. With proof P, the group manager verifies the integrity of the chosen blocks.

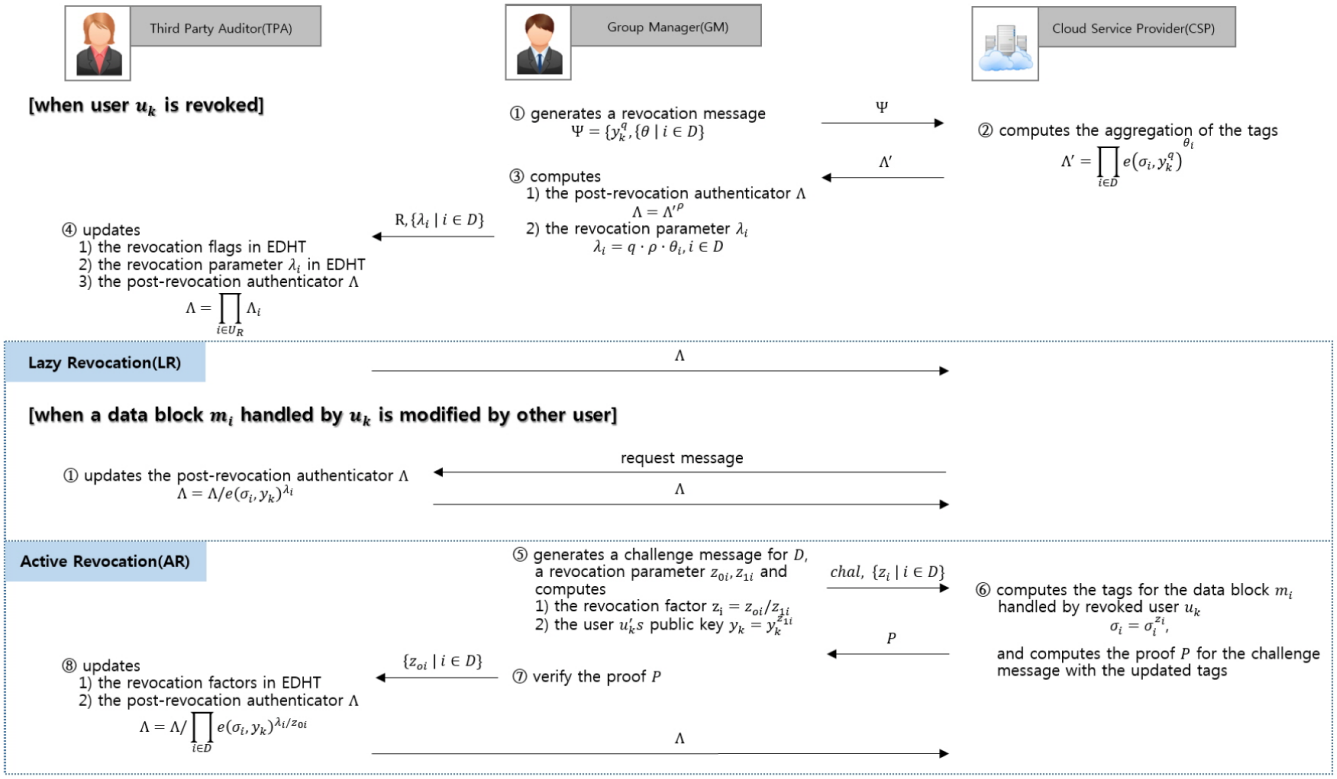


FIGURE 3. Lazy revocation (LR) and active revocation (AR) processes.

If it is valid, the group manager sends the revocation factor z_{0i} to the TPA. The TPA then updates the revocation factor in the EDHT and the post-authenticator Λ as

$$\Lambda = \Lambda / \prod_{i \in D} e(\sigma_i, y_k)^{\lambda_i / z_{0i}}. \tag{31}$$

Finally, the TPA sends the post-authenticator Λ to the CSP.

A. CORRECTNESS

The correctness of our scheme can be proved as follows:

$$\begin{aligned}
&{\Theta_1}^w \cdot \Phi^{H_2(\Theta_1)} \\
&= e(u, y_1)^{r_1 w} \cdot \prod_{i \in IDX} ((e(\sigma_i, \gamma_j))^{s_i})^{H_2(\Theta_1)}, \\
&= e(u, y_1)^{r_1 w} \cdot \prod_{i \in IDX} e((h_{i,j} \cdot u^{m_i})^{a_j}, g^{w \cdot a_1 / a_j})^{s_i H_2(\Theta_1)} \\
&= e(u, y_1)^{r_1 w} \cdot \prod_{i \in IDX} e((h_{i,j} \cdot u^{m_i}), {y_1}^w)^{s_i H_2(\Theta_1)}, \\
&= e(u, y_1)^{r_1 w} \cdot (\prod_{i \in IDX} e(h_{i,j}, {y_1}^w)^{s_i H_2(\Theta_1)}), \\
&\quad \cdot \prod_{i \in IDX} e(u^{m_i s_i H_2(\Theta_1)}, {y_1}^w) \\
&= (\prod_{i \in IDX} e(h_{i,j}^{s_i H_2(\Theta_1)}, {y_1}^w)) \cdot e(u^{\Sigma m_i s_i H_2(\Theta_1)}, {y_1}^w) \\
&\quad \cdot e(u^{r_1}, {y_1}^w) \\
&= (\prod_{i \in IDX} e(h_{i,j}^{s_i H_2(\Theta_1)}, {y_1}^w)) \cdot e(u^{\Sigma m_i s_i H_2(\Theta_1) + r_1}, {y_1}^w) \\
&= (\prod_{i \in IDX} e(h_{i,j}^{s_i H_2(\Theta_1)}, {y_1}^w)) \cdot e(u^{v_1}, y_1)^w \\
&= e(\prod_{i \in IDX} h_{i,j}^{s_i H_2(\Theta_1)} \cdot u^{v_1}, y_1)^w \\
&{\Theta_2}^{\eta \cdot w} \cdot T^{\eta \cdot \epsilon \cdot H_2(\Theta_2)} \\
&= e(u, y_1)^{r_2 \cdot \eta \cdot w} \cdot \prod_{i \in D} ((e(\sigma_i, \tau_j))^{\beta_i})^{\eta \cdot \epsilon \cdot H_2(\Theta_2)} \\
&= e(u, y_1)^{r_2 \cdot \eta \cdot w} \\
&\quad \cdot \prod_{i \in D} e((h_{i,j} \cdot u^{m_i})^{a_j}, g^{(w \cdot a_1)/(a_j \cdot \epsilon)})^{\beta_i \cdot \eta \cdot \epsilon \cdot H_2(\Theta_2)} \\
&= e(u, y_1)^{r_2 \cdot \eta \cdot w} \cdot \prod_{i \in D} e((h_{i,j} \cdot u^{m_i}), {y_1}^w)^{\beta_i \cdot \eta \cdot H_2(\Theta_2)} \\
&= e(u, y_1)^{r_2 \cdot \eta \cdot w} \cdot (\prod_{i \in D} e(h_{i,j}, {y_1}^w)^{\beta_i \cdot \eta \cdot H_2(\Theta_2)}) \\
&\quad \cdot \prod_{i \in D} e(u^{m_i \cdot \beta_i \cdot \eta \cdot H_2(\Theta_2)}, {y_1}^w) \\
&= e(u, y_1)^{r_2 \cdot \eta \cdot w} \cdot (\prod_{i \in D} e(h_{i,j}, {y_1}^w)^{\lambda_i H_2(\Theta_2)}), \\
&\quad \cdot \prod_{i \in D} e(u^{m_i \cdot \beta_i \cdot \eta \cdot H_2(\Theta_2)}, {y_1}^w) \\
&= (\prod_{i \in D} e(h_{i,j}^{\lambda_i H_2(\Theta_2)}, {y_1}^w)) \cdot e(u^{\Sigma m_i \beta_i H_2(\Theta_2)}, {y_1}^w)^{\eta} \\
&\quad \cdot e(u, y_1)^{r_2 \cdot \eta \cdot w} \\
&= (\prod_{i \in D} e(h_{i,j}^{\lambda_i H_2(\Theta_2)}, {y_1}^w)) \cdot e(u^{\Sigma m_i \beta_i H_2(\Theta_2) + r_2}, {y_1}^{\eta w})
\end{aligned}$$

$$= (\prod_{i\in D} e(h_{i,j}^{\lambda_i H_2(\Theta_2)}, {y_1}^w)) \cdot e(u^{v_2\cdot\eta}, y_1)^w$$
$$= e(\prod_{i\in D} h_{i,j}^{\lambda_i H_2(\Theta_2)} \cdot u^{v_2\cdot\eta}, y_1)^w$$

V. ANALYSIS

A. SECURITY ANALYSIS

We need to show that the CSP cannot generate a valid proof for the challenge query of the TPA without faithfully storing the message blocks. We also need to show that CSP cannot generate a valid tag for the message without any secret information. These security notions can be guaranteed by proving the following theorems:

Theorem 5.1: If the CDH assumption holds in $\mathbb{G}_1$, our public auditing scheme satisfies tag-unforgeability under the random oracle model.

Proof of Theorem 5.1. Suppose $\mathcal{A}$ is a forger that $(t, q_h, q_t, q_u, q_k, \epsilon)$-breaks our public auditing scheme by generating a valid message and tag pair under the secret key a_k of the user u_k. We can then construct an algorithm $\mathcal{B}$ that solves the CDH problem on $\mathbb{G}_1$.

The algorithm $\mathcal{B}$ is given by $(g, g^a, g^b) \in \mathbb{G}_1$ as an input of the CDH problem. The algorithm $\mathcal{B}$ simulates the challenger and interacts with the forger $\mathcal{A}$ in the tag-unforgeability game.

1. The forger A outputs a user u_k^*, where it wishes to be forged.
2. The algorithm $\mathcal{B}$ sets $y_1 = (g^b)^{a_1}$ as the public key of user u_1 and computes the public key of other users as $y_k = (g^{b\cdot a_1})^{1/a_k}{}_{1\le k<k^*,k^*<k\le K-1}$, where $a_1, a_k \leftarrow Z_p^*$ are random number. The algorithm $\mathcal{B}$ maintains the tuple (k, u_k, a_k, y_k) in *Tab*1. The algorithm $\mathcal{B}$ computes $u = g^d$, where $d \leftarrow Z_p^*$ is a random number, and sets $y_k^* = g^{a_1 v}$ as the public key of user u_k^*, where $v \leftarrow Z_p^*$. The algorithm $\mathcal{B}$ sends PK to forger $\mathcal{A}$.
3. Algorithm $\mathcal{B}$ simulates the HASH, TAGGEN, UPDATE, and KEYEXT oracles as follows:
   (*For* HASH *oracle*) At any time the forger $\mathcal{A}$ can query the random hash oracle H_1 for $(ID||i||k||v_i||t_i)$. To respond to hash queries, the algorithm $\mathcal{B}$ maintains the tuple $(i, \delta_i, \xi_i, c_i, k, v_i, t_i)$ in *Tab*2 as generated below. *Tab*2 was initially empty. When the forger $\mathcal{A}$ makes a query $(i, \delta_i, \xi_i, c_i, k, v_i, t_i)$, the algorithm $\mathcal{B}$ responds as follows:
   a) If $i \notin$ *Tab*2, the algorithm $\mathcal{B}$ flips a random coin $c_i \in \{0, 1\}$ such that $Pr[c_i = 0] = 1/(q_t + 1)$, where q_t is the maximum number of TAGGEN queries. The algorithm $\mathcal{B}$ chooses a random number $\delta_i \in Z_p^*$ and computes $\xi_i = (g^a)^{(1-c_i)} \cdot g^{\delta_i}$ as the hash value. The algorithm $\mathcal{B}$ then stores the tuple $(i, \delta_i, \xi_i, c_i, k, *, *)$ in *Tab*2 and responds to forger $\mathcal{A}$ by setting $H_1(\cdot) = \xi_i$.
   b) If $i \in$ *Tab*2, the algorithm $\mathcal{B}$ responds with $H_1(\cdot) = \xi_i$ in *Tab*2.

   (*For the* TAGGEN *oracle*) At any time, the forger $\mathcal{A}$ can query (i, m_i) to the TAGGEN oracle. When the forger $\mathcal{A}$ queries (i, m_i) to the TAGGEN oracle, the algorithm $\mathcal{B}$ responds as follows:
   a) If $i \notin$ *Tab*2, the algorithm $\mathcal{B}$ picks a random number δ_i and computes $\xi_i = g^{\delta_i}$ and $\sigma_i = (g^b)^{(\delta_i + d\cdot m_i)/v} = (g^{\delta_i} \cdot g^{d\cdot m_i})^{b/v}$. The algorithm $\mathcal{B}$ responds with σ_i and stores the tuple $(i, \delta_i, \xi_i, *, m_i, \sigma_i)$ in *Tab*3.
   b) If $i \in$ *Tab*2, the algorithm $\mathcal{B}$ retrieves the tuple $(i, \delta_i, \xi_i, c_i, k, v_i, t_i)$. If $c_i = 0$, then the algorithm $\mathcal{B}$ reports a failure and terminates. If $c_i = 1$, the algorithm $\mathcal{B}$ computes $\sigma_i = (g^b)^{(\delta_i + d\cdot m_i)/v} = (g^{\delta_i} \cdot g^{d\cdot m_i})^{b/v}$ with δ_i and m_i. The algorithm $\mathcal{B}$ responds with σ_i and stores the tuple $(i, \delta_i, \xi_i, c_i, m_i, \sigma_i)$ in *Tab*3.

   If $u_k \neq u_k^*$, the algorithm $\mathcal{B}$ retrieves the tuple (u_k, a_k) and responds with $\sigma_i = (g^{\delta_i} \cdot g^{d\cdot m_i})^{a_k}$ as in UPDATE oracle.
   (*For the* UPDATE *oracle*) At any time, the forger $\mathcal{A}$ can query the UPDATE oracle. When the algorithm $\mathcal{A}$ queries (i, m_i', u_k) to the UPDATE oracle, the algorithm $\mathcal{B}$ responds as follows:
   a) If $u_k \neq u_k^*$ and $i \notin$ *Tab*3, the algorithm $\mathcal{B}$ picks a random number $\delta_i \in Z_p^*$, and computes $\sigma_i' = (g^{\delta_i} \cdot g^{d\cdot m_i'})^{a_k}$ with δ_i, m_i', and the secret key a_k of user u_k. The adversary $\mathcal{B}$ responds with σ_i' and stores the tuple $(i, \delta_i, \xi_i, *, m_i', \sigma_i', u_k)$ in *Tab*4.
   b) If $u_k \neq u_k^*$ and $i \in$ *Tab*3, the algorithm $\mathcal{B}$ retrieves the tuple $(i, \delta_i, \xi_i, c_i, , m_i, \sigma_i)$ and computes $\sigma_i' = (g^{\delta_i} \cdot g^{d\cdot m_i'})^{a_k}$ with δ_i, m_i', and the secret key a_k of the user u_k. The adversary $\mathcal{B}$ responds with σ_i' and stores the tuple $(i, \delta_i, \xi_i, c_i, m_i', \sigma_i', u_k)$ in *Tab*4.
   c) If $u_k = u_k^*$ and $i \notin$ *Tab*3, the algorithm $\mathcal{B}$ picks a random number $\delta_i \in Z_p^*$, and computes $\sigma_i' = (g^b)^{(\delta_i + d\cdot m_i')/v}$ with δ_i and m_i'. The adversary $\mathcal{B}$ responds with σ_i' and stores the tuple $(i, \delta_i, \xi_i, *, m_i', \sigma_i', u_k)$ in *Tab*4.
   d) If $u_k = u_k^*$ and $i \in$ *Tab*3, the algorithm $\mathcal{B}$ retrieves the tuple $(i, \delta_i, @_i, c_i, m_i, \sigma_i)$. If $c_i = 0$, the algorithm $\mathcal{B}$ then reports a failure and terminates. If $c_i = 1$, the algorithm $\mathcal{B}$ computes $\sigma_i' = (g^b)^{(\delta_i + d\cdot m_i')/v}$ with δ_i and m_i'. The adversary $\mathcal{B}$ responds with σ_i' and stores the tuple $(i, \delta_i, \xi_i, c_i, m_i', \sigma_i', u_k)$ in *Tab*4.

   (*For the* KEYEXT *oracle*) At any time the forger $\mathcal{A}$ can query the KEYEXT oracle. When the algorithm $\mathcal{A}$ queries u_k to the KEYEXT oracle, the algorithm $\mathcal{B}$ responds as follows:
   a) If $u_k = u_k^*$, the algorithm $\mathcal{B}$ reports a failure and terminates.
   b) If $u_k \neq u_k^*$, the algorithm $\mathcal{B}$ retrieves the tuple (u_k, a_k) and responds with a_k.
4. Finally, the forger $\mathcal{A}$ outputs $(u_k^*, i, m_i^*, \sigma_i^*)$ as the forged message and tag pair such that no TAGGEN and UPDATE query were issued for m_i^*, and no KeyExt query was issued for u_k^*. Here, $\mathcal{B}$ checks whether the pair

(m_i^*, σ_i^*) is valid under the given public key. If (m_i^*, σ_i^*) is invalid, then the algorithm $\mathcal{B}$ aborts. If $i \in Tab3$ and $c_i = 0$, σ_i^* satisfies the following equation

$$\begin{aligned}\sigma_i^* &= ((g^{a+\delta_i}) \cdot g^{t \cdot m_i})^{b/v} \\ &= (g^{a+\delta_i})^{b/v} \cdot g^{(t \cdot m_i) \cdot b/v} \\ &= g^{ab/v} \cdot (g^b)^{\delta_i/v} \cdot (g^b)^{(t \cdot m_i)/v},\end{aligned}$$

then the algorithm $\mathcal{B}$ outputs

$$g^{ab} = (\sigma_i^*)^v / ((g^b)^{\delta_i} \cdot (g^b)^{(t \cdot m_i)}).$$

We refer to the technique in [36] for probability analysis. To succeed in the above game, the algorithm $\mathcal{B}$ does not abort during the TagGen queries of forger $\mathcal{A}$, the Update queries and the KeyExt queries, and the forger $\mathcal{A}$ outputs a valid message and tag pair under the condition that the algorithm $\mathcal{B}$ does not abort. In addition, the forger $\mathcal{A}$ outputs a valid message and tag pair when $c_i = 0$ for the tuple containing i in *Tab*3. The probability that algorithm $\mathcal{B}$ will not abort in the TagGen queries of forger $\mathcal{A}$, the Update queries and KeyExt queries is at least $1/e^3$, where e is a mathematical constant and is the base of the natural logarithm. The probability that the forger $\mathcal{A}$ outputs a valid message and tag pair under the condition that the algorithm $\mathcal{B}$ does not abort is at least ε, and the probability that the forger $\mathcal{A}$ outputs a valid message and tag pair when $c_i = 0$ for the tuple containing i on *Tab*3 is at least $1/(q_s+1)$. Therefore, the algorithm $\mathcal{B}$ outputs a correct answer with probability $\varepsilon/(e^3 \cdot (q_s + 1))$. Thus, if the CDH assumption holds, no algorithm exists that breaks the tag-unforgeability of our scheme with a non-negligible probability.

Theorem 5.2: Outputting *Accept* of the verification algorithm implies that the CSP must possess the uncompromised selected data, as expected by the user.

Proof of Theorem 5.2. The proof can be completed by showing the existence of an extractor of $\sum_{i \in IDX} m_i \cdot s_i$ and $\sum_{j \in D} m_j \cdot \beta_j$ in the random oracle model. With this proof, we can show that our scheme is secure against a proof forgery attack because only the CSP, which possesses the uncompromised selected data, can pass the verification process. Our proof technique follows from [8]. Here, the extractor and H_2 can be modeled as an adversary and a random oracle, respectively. For $H_2(\Theta_1) = h_1$, CSP returns valid Φ and ν_1 such that the following equation holds:

$$\Theta_1{}^w \cdot \Phi^{h_1} = \prod_{i \in IDX} e(h_{i,j}^{s_i \cdot h_1} \cdot u^{\nu_1}, y_1)^w. \tag{32}$$

The extractor can rewind the challenge-proof protocol to the point just before Θ_1 is queried to H_2. The extractor then sets $H_2(\Theta_1)$ to $h_1' \neq h_1$. CSP returns a valid Φ and ν_1' such that the following equation holds:

$$\Theta_1{}^w \cdot \Phi^{h_1'} = \prod_{i \in IDX} e(h_{i,j}^{s_i \cdot h_1'} \cdot u^{\nu_1'}, y_1)^w. \tag{33}$$

We have the following equations by dividing (32) by (33). Recall that $\Phi = \prod_{i \in IDX} (e(\sigma_i, \gamma_j))^{s_i}$.

$$\begin{aligned}\Phi^{h_1 - h_1'} &= \prod_{i \in IDX} e(h_{i,j}^{s_i \cdot (h_1 - h_1')} \cdot u^{\nu_1 - \nu_1'}, y_1)^w \\ (\Leftrightarrow) \quad & \prod_{i \in IDX} e(\sigma_i, \gamma_j)^{s_i \cdot (h_1 - h_1')} \\ &= \prod_{i \in IDX} e(h_{i,j}^{s_i \cdot (h_1 - h_1')} \cdot u^{\nu_1 - \nu_1'}, y_1)^w \\ (\Leftrightarrow) \quad & \prod_{i \in IDX} e(h_{i,j} \cdot u^{m_i}, y_1^w)^{s_i \cdot (h_1 - h_1')}, \\ &= \prod_{i \in IDX} e(h_{i,j}^{s_i \cdot (h_1 - h_1')} \cdot u^{\nu_1 - \nu_1'}, y_1^w) \\ (\Leftrightarrow) \quad & \prod_{i \in IDX} h_{i,j}^{s_i \cdot (h_1 - h_1')} \cdot u^{n_i \cdot s_i \cdot (h_1 - h_1')}, \\ &= \prod_{i \in IDX} h_{i,j}^{s_i \cdot (h_1 - h_1')} \cdot u^{\nu_1 - \nu_1'} \\ (\Leftrightarrow) \quad & \prod_{i \in IDX} u^{m_i \cdot s_i \cdot (h_1 - h_1')} = u^{\nu_1 - \nu_1'}, \\ (\Leftrightarrow) \quad & (h_1 - h_1') \cdot \sum_{i \in IDX} m_i \cdot s_i = \nu_1 - \nu_1', \\ (\Leftrightarrow) \quad & \sum_{i \in IDX} m_i \cdot s_i = (\nu_1 - \nu_1')/(h_1 - h_1'),\end{aligned}$$

Similarly, we can obtain the following equations from a valid T and ν_2.

$$\begin{aligned}T^{\eta \cdot \epsilon \cdot (h_2 - h_2')} &= \prod_{i \in D} e(h_{i,j}^{\lambda_i \cdot (h_2 - h_2')} \cdot u^{(\nu_2 - \nu_2') \cdot \eta}, y_1)^w \\ (\Leftrightarrow) \quad & \prod_{i \in D} e(\sigma_i, \tau_j)^{\beta_i \cdot \eta \cdot \epsilon \cdot (h_2 - h_2')}, \\ &= \prod_{i \in D} e(h_{i,j}^{\lambda_i \cdot (h_2 - h_2')} \cdot u^{(\nu_2 - \nu_2') \cdot \eta}, y_1)^w \\ (\Leftrightarrow) \quad & \prod_{i \in D} e(h_{i,j} \cdot u^{m_i}, y_1^w)^{\beta_i \cdot \eta \cdot (h_2 - h_2')}, \\ &= \prod_{i \in D} e(h_{i,j}^{\lambda_i \cdot (h_2 - h_2')} \cdot u^{(\nu_2 - \nu_2') \cdot \eta}, y_1^w) \\ (\Leftrightarrow) \quad & \prod_{i \in D} h_{i,j}^{\beta_i \cdot \eta \cdot (h_2 - h_2')} \cdot u^{m_i \cdot \beta_i \cdot \eta \cdot (h_2 - h_2')}, \\ &= \prod_{i \in D} h_{i,j}^{\lambda_i \cdot (h_2 - h_2')} \cdot u^{(\nu_2 - \nu_2') \cdot \eta} \\ (\Leftrightarrow) \quad & \prod_{i \in D} u^{m_i \cdot \beta_i \cdot \eta \cdot (h_2 - h_2')} = u^{(\nu_2 - \nu_2') \cdot \eta}, \\ (\Leftrightarrow) \quad & (h_2 - h_2') \cdot \sum_{i \in D} m_i \cdot \beta_i \cdot \eta = (\nu_2 - \nu_2') \cdot \eta, \\ (\Leftrightarrow) \quad & \sum_{i \in D} m_i \cdot \beta_i = (\nu_2 - \nu_2')/(h_2 - h_2'),\end{aligned}$$

Then, the extractor obtains $\sum_{i \in IDX} m_i \cdot s_i$ and $\sum_{j \in D} m_j \cdot \beta_j$ as a valid response of the proof system.

Theorem 5.3: The TPA cannot obtain any information on the challenged messages from the proof of the CSP.

TABLE 2. Comparisons of computation cost.

Schemes	Tag generation	Proof verification
[16]	$1Mul_{\mathbb{G}_1} + 2Exp_{\mathbb{G}_1} + 1hash_{\mathbb{G}_1} + 2Pair$	$(c+2d)Mul_{\mathbb{G}_1} + (c+d)Exp_{\mathbb{G}_1} + chash_{\mathbb{G}_1} + dMul_{\mathbb{G}_T} + (d+1)Pair$
[19]	$sMul_{\mathbb{G}_1} + (s+2)Exp_{\mathbb{G}_1} + 1hash$	$(c+1)Mul_{\mathbb{G}_1} + 6Exp_{\mathbb{G}_1} + chash + 2Mul_{\mathbb{G}_T} + 3Pair$
[25]	$1Mul_{\mathbb{G}_1} + 3Exp_{\mathbb{G}_1} + 1hash$	$(\|IDX\|+\|D\|+4)Mul_{\mathbb{G}_1} + 2Exp_{\mathbb{G}_1} + (\|IDX\|+\|D\|)hash + 2Mul_{\mathbb{G}_T} + 3Exp_{\mathbb{G}_T} + 1Pair$
Ours	$1Mul_{\mathbb{G}_1} + 2Exp_{\mathbb{G}_1} + 1hash_{\mathbb{G}_1}$	$(\|IDX\|+\|D\|+6)Mul_{\mathbb{G}_1} + (\|IDX\|+\|D\|+2)Exp_{\mathbb{G}_1} + (\|IDX\|+\|D\|)hash_{\mathbb{G}_1} + 2hash + 2Mul_{\mathbb{G}_T} + 6Exp_{\mathbb{G}_T} + 2Pair$

Note: $Mul_{\mathbb{G}_1}$ denotes one multiplication in $\mathbb{G}_1$, $Exp_{\mathbb{G}_1}$ denotes one exponentiation in $\mathbb{G}_1$, $hash$ denotes one hashing operation in $\mathbb{G}_1$, $Mul_{\mathbb{G}_T}$ denotes one multiplication in $\mathbb{G}_T$, $Exp_{\mathbb{G}_T}$ denotes one exponentiation in $\mathbb{G}_T$, and $Pair$ denotes one pairing operation in $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_T$. In addition, c and $|IDX|$ are the number of challenged blocks, d and s are the number of subsets of the challenged blocks, and $|D|$ is the number of data blocks modified by the revoked users.

TABLE 3. Comparisons of communication cost.

Schemes	Challenge	Response
[16]	$c(\|n\| + \|q\|)$	$2d\|p\| + c\|id\|$
[19]	$c(\|n\| + \|q\|)$	$2\|q\|$
[25]	$c(\|n\| + \|p\|) + (\|D\| + \|U_L\| + \|U_R\|)\|p\|$	$6\|p\|$
Ours	$c(\|n\| + \|p\|) + (\|D\| + \|U_L\| + \|U_R\|)\|p\|$	$7\|p\|$

Note: Here, c is the number of challenged blocks, $|n|$ is the size of an element of set $[1, n]$, q is the size of an element of $\mathbb{Z}_q$, p is the size of an element of $\mathbb{Z}_p$, $|id|$ is the size of the block identifier, $|D|$ is the number of data blocks modified by the revoked users, $|U_L|$ is the number of legal users, and $|U_R|$ is the number of the revoked users.

Proof of Theorem 5.3. To prove the theorem, it is sufficient to show that a valid poof can be simulated without any message block information in the random oracle model. In this proof, the TPA and H_2 can be modeled as an adversary and a random oracle, respectively. Given a valid Φ and T from the CSP, the TPA randomly chooses $h_i^*, v_i^* \leftarrow \mathbb{Z}_p^*$ and sets Θ_i^* as follows: $(i \in 1, 2)$

$$\Theta_1^* = e(\prod_{i \in IDX} h_{i,j}^{s_i \cdot h_1^*} \cdot u^{v_1^*}, y_1)/\Phi^{-h_1^* \cdot w} \tag{34}$$

$$\Theta_2^* = e(\prod_{i \in D} h_{i,j}^{\lambda_i \cdot h_2^*} \cdot u^{v_2^* \cdot \eta}, y_1)^{-\eta}/T^{-w \cdot \epsilon \cdot h_2^*}. \tag{35}$$

Finally, the random oracle H_2 is programmed as $H_2(\Theta_1^*) = h_1^*$ and $H_2(\Theta_2^*) = h_2^*$. We can easily check the validity of $P^* = \{\Phi, T, v_1^*, v_2^*, \Theta_1^*, \Theta_2^*\}$.

B. EFFICIENCY ANALYSIS

In this section, we compare our proposed scheme with Panda [16]–[19] and Tian *et al.*'s scheme (PASCD) [25] in terms of the computation and communication costs.

1) COMPUTATION COST

We first compare the computational complexity of all four schemes in the tag generation and proof verification phases. In the tag generation phase, from Table 2, [19] has the highest computation costs, which is $sMul_{\mathbb{G}_1} + (s+2)Exp_{\mathbb{G}_1} + 1hash$. PASCD [25] and our scheme have lower computation costs than Panda [16] because they do not require pairing operations. During the proof verification phase, Panda [16] has the highest computation costs, i.e., $(c + 2d)Mul_{\mathbb{G}_1} + (c + d)Exp_{\mathbb{G}_1} + chash_{\mathbb{G}_1} + dMul_{\mathbb{G}_T} + (d+1)Pair$, as d increases, whereas PASCD [25] and our scheme perform a constant pairing operation regardless of the size of d. Comparing our scheme with [19] and PACSD [25], our scheme has a slightly higher computation cost; however, meaningfully, it solves the security problem of PASCD [25] with almost the same computational cost.

2) COMMUNICATION COST

Table 3 compares the communication complexity of all four schemes in the challenge and response phases. In the challenge phase, Panda [16] is affected only by the number of challenged blocks, c, whereas [19], PACSD [25] and our scheme are affected not only by c, but also by the total number of users. Conversely, during the response phase of delivering the generated proof to the TPA, Panda [16] increases the communication cost according to the sizes of d and c, and PACSD [25] and our scheme both have a constant computation cost regardless of the sizes of d and c. Even though [19] has a lower computation cost, it does not provide the identity traceability and does not guarantee the security against the collusion attack. Comparing our scheme with PACSD [25], our scheme provides similar communication costs while providing higher security.

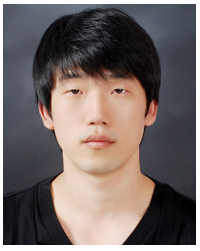
VI. CONCLUSION

Cloud storage auditing is an extremely important technique for resolving the problem of ensuring the integrity of stored data in cloud storage. Because the need for the concept is shared, many schemes providing different functions and security levels have been proposed. In 2019, Tian *et al.* [25] proposed a scheme that supports data privacy, identity traceability, and group dynamics and claimed that their scheme is secure against collusion attacks between the CSPs and revoked users. In this paper, we showed in their scheme that a tag can be forged from a valid message and tag pair without knowing any secret values. We also showed that a proof can be forged by a collusion attack, even if some challenged messages have been deleted. We then proposed a new scheme that is secure against the above attacks while providing the same functionality as their approach. We also provided formal security proofs and an analysis of the computation costs of both schemes.

REFERENCES

[1] (Apr. 2021). *Cloud Storage-Global Market Trajectory and Analytics*. [Online]. Available: https://www.researchandmarkets.com/reports/5140992/cloud-storage-global-market-trajectory-and

[2] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, and D. Song, "Provable data possession at untrusted stores," in *Proc. 14th ACM Conf. Comput. Commun. Secur. (CCS)*, 2007, pp. 598–609.
[3] A. Juels and B. S. Kaliski, "PORs: Proofs of retrievability for large files," in *Proc. 14th ACM Conf. Comput. Commun. Secur. (CCS)*, Oct. 2007, pp. 584–597.
[4] H. Shacham and B. Waters, "Compact proofs of retrievability," in *Proc. Int. Conf. Theory Appl. Cryptol. Inf. Secur.* Berlin, Germany: Springer, 2008, pp. 90–107.
[5] C. Wang, Q. Wang, K. Ren, and W. Lou, "Privacy-preserving public auditing for data storage security in cloud computing," in *Proc. IEEE INFOCOM*, Mar. 2010, pp. 1–9.
[6] Z. Hao, S. Zhong, and N. Yu, "A privacy-preserving remote data integrity checking protocol with data dynamics and public verifiability," *IEEE Trans. Knowl. Data Eng.*, vol. 23, no. 9, pp. 1432–1437, Sep. 2011.
[7] K. Yang and X. Jia, "An efficient and secure dynamic auditing protocol for data storage in cloud computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 9, pp. 1717–1726, Sep. 2013.
[8] C. Wang, S. S. M. Chow, Q. Wang, K. Ren, and W. Lou, "Privacy-preserving public auditing for secure cloud storage," *IEEE Trans. Comput.*, vol. 62, no. 2, pp. 362–375, Feb. 2013.
[9] K. He, C. Huang, K. Yang, and J. Shi, "Identity-preserving public auditing for shared cloud data," in *Proc. IEEE 23rd Int. Symp. Quality Service (IWQoS)*, Jun. 2015, pp. 159–164.
[10] C. Erway, A. Küpçü, C. Papamanthou, and R. Tamassia, "Dynamic provable data possession," in *Proc. 16th ACM Conf. Comput. Commun. Secur. (CCS)*, New York, NY, USA, 2009, pp. 213–222.
[11] Q. Wang, C. Wang, K. Ren, W. Lou, and J. Li, "Enabling public auditability and data dynamics for storage security in cloud computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 22, no. 5, pp. 847–859, May 2011.
[12] Y. Zhu, G.-J. Ahn, H. Hu, S. S. Yau, H. G. An, and C.-J. Hu, "Dynamic audit services for outsourced storages in clouds," *IEEE Trans. Services Comput.*, vol. 6, no. 2, pp. 227–238, Apr./Jun. 2013.
[13] J. Shen, J. Shen, X. Chen, X. Huang, and W. Susilo, "An efficient public auditing protocol with novel dynamic structure for cloud data," *IEEE Trans. Inf. Forensics Security*, vol. 12, no. 10, pp. 2402–2415, Oct. 2017.
[14] B. Wang, B. Li, and H. Li, "Knox: Privacy-preserving auditing for shared data with large groups in the cloud," in *Proc. 10th Interfaces Conf. Appl. Crypto. Netw. Secur.*, 2012, pp. 507–525.
[15] B. Wang, B. Li, and H. Li, "Oruta: Privacy-preserving public auditing for shared data in the cloud," *IEEE Trans. Cloud Comput.*, vol. 2, no. 1, pp. 43–56, Jan./Mar. 2014.
[16] B. Wang, B. Li, and H. Li, "Panda: Public auditing for shared data with efficient user revocation in the cloud," *IEEE Trans. Serv. Comput.*, vol. 8, no. 1, pp. 92–106, Jan./Feb. 2015.
[17] T. Jiang, X. Chen, and J. Ma, "Public integrity auditing for shared dynamic cloud data with group user revocation," *IEEE Trans. Comput.*, vol. 65, no. 8, pp. 2363–2373, Aug. 2016.
[18] J. Yuan and S. Yu, "Efficient public integrity checking for cloud data sharing with multi-user modification," in *Proc. IEEE Conf. Comput. Commun. (IEEE INFOCOM)*, Apr. 2014, pp. 2121–2129.
[19] J. Yuan and S. Yu, "Public integrity auditing for dynamic data sharing with multiuser modification," *IEEE Trans. Inf. Forensics Security*, vol. 10, no. 8, pp. 1717–1726, Aug. 2015.
[20] G. Yang, J. Yu, W. Shen, Q. Su, Z. Fu, and R. Hao, "Enabling public auditing for shared data in cloud storage supporting identity privacy and traceability," *J. Syst. Softw.*, vol. 113, pp. 130–139, Mar. 2016.
[21] A. Fu, S. Yu, Y. Zhang, H. Wang, and C. Huang, "NPP: A new privacy-aware public auditing scheme for cloud data sharing with group users," *IEEE Trans. Big Data*, vol. 8, no. 1, pp. 14–24, Feb. 2022, doi: 10.1109/TBDATA.2017.2701347.
[22] W. Shen, J. Qin, J. Yu, R. Hao, and J. Hu, "Enabling identity-based integrity auditing and data sharing with sensitive information hiding for secure cloud storage," *IEEE Trans. Inf. Forensics Security*, vol. 14, no. 2, pp. 331–346, Feb. 2018.
[23] Y. Zhang, C. Chen, D. Zheng, R. Guo, and S. Xu, "Shared dynamic data audit supporting anonymous user revocation in cloud storage," *IEEE Access*, vol. 7, pp. 113832–113843, 2019.
[24] G. Wu, Y. Mu, W. Susilo, F. Guo, and F. Zhang, "Threshold privacy-preserving cloud auditing with multiple uploaders," *Int. J. Inf. Secur.*, vol. 18, no. 3, pp. 321–331, Jun. 2019.
[25] H. Tian, F. Nan, H. Jiang, C.-C. Chang, J. Ning, and Y. Huang, "Public auditing for shared cloud data with efficient and secure group management," *Inf. Sci.*, vol. 472, pp. 107–125, Jan. 2019.
[26] S.-C. Chang and J.-L. Wu, "A privacy-preserving cloud-based data management system with efficient revocation scheme," *Int. J. comput. Sci. Eng.*, vol. 20, no. 2, pp. 190–199, 2019.
[27] Y. Xu, L. Ding, J. Cui, H. Zhong, and J. Yu, "PP-CSA: A privacy-preserving cloud storage auditing scheme for data sharing," *IEEE Syst. J.*, vol. 15, no. 3, pp. 3730–3739, Sep. 2020.
[28] Y. Zhang, J. Yu, R. Hao, C. Wang, and K. Ren, "Enabling efficient user revocation in identity-based cloud storage auditing for shared big data," *IEEE Trans. Dependable Secure Comput.*, vol. 17, no. 3, pp. 608–619, Jun. 2020.
[29] S. Thokchom and D. K. Saikia, "Privacy preserving integrity checking of shared dynamic cloud data with user revocation," *J. Inf. Secur. Appl.*, vol. 50, Feb. 2020, Art. no. 102427.
[30] Y. Xu, S. Sun, J. Cui, and H. Zhong, "Intrusion-resilient public cloud auditing scheme with authenticator update," *Inf. Sci.*, vol. 512, pp. 616–628, Feb. 2020.
[31] J. Liu, X. A. Wang, Z. Liu, H. Wang, and X. Yang, "Privacy-preserving public cloud audit scheme supporting dynamic data for unmanned aerial vehicles," *IEEE Access*, vol. 8, pp. 79428–79439, 2020.
[32] X. Yang, M. Wang, T. Li, R. Liu, and C. Wang, "Privacy-preserving cloud auditing for multiple users scheme with authorization and traceability," *IEEE Access*, vol. 8, pp. 130866–130877, 2020.
[33] Y. Su, Y. Li, K. Zhang, and B. Yang, "A privacy-preserving public integrity check scheme for outsourced EHRs," *Inf. Sci.*, vol. 542, pp. 112–130, Jan. 2021.
[34] X. Gao, J. Yu, Y. Chang, H. Wang, and J. Fan, "Checking only when it is necessary: Enabling integrity auditing based on the keyword with sensitive information privacy for encrypted cloud data," *IEEE Trans. Dependable Secure Comput.*, early access, Aug. 24, 2021, doi: 10.1109/TDSC.2021.3106780.
[35] C. Hu, Y. Xu, P. Liu, J. Yu, S. Guo, and M. Zhao, "Enabling cloud storage auditing with key-exposure resilience under continual key-leakage," *Inf. Sci.*, vol. 520, pp. 15–30, May 2020.
[36] D. Boneh, B. Lynn, and H. Shacham, "Short signatures from the Weil pairing," in *Proc. ASIACRYPT*, 2001, pp. 514–533.
[37] D. Catalano and D. Fiore, "Vector commitments and their applications," in *Proc. Public Key Cryptogr.*, 2013, pp. 55–72.
[38] D. Boneh and H. Shacham, "Group signatures with verifier-local revocation," in *Proc. 11th ACM Conf. Comput. Commun. Secur. (CCS)*, 2004, pp. 168–177.
[39] G. Ateniese, D. H. Chou, B. De Medeiros, and G. Tsudik, "Sanitizable signatures," in *Proc. Eur. Symp. Res. Comput. Secur.*, 2005, pp. 159–177.



DONGMIN KIM received the M.S. and Ph.D. degrees from the Graduate School of Information Security, Korea University, Seoul, South Korea, in 2011 and 2017, respectively. He has been a Senior Researcher at the Affiliated Institute of ETRI, Daejeon, South Korea, since 2016. His research interests include cryptography, database security, privacy-enhancing technology, public auditing protocols, and cryptographic module validation programs.

KEE SUNG KIM received the M.S. and Ph.D. degrees from the Graduate School of Information Security, Korea University, Seoul, South Korea, in 2011 and 2015, respectively. He was a Senior Researcher at the Affiliated Institute of ETRI, Daejeon, South Korea, from 2014 to 2018. He is currently an Assistant Professor at the School of Information Technology Engineering, Daegu Catholic University, South Korea. His research interests include cryptography, database security, privacy-enhancing technology, and secure protocols.