Hindawi
Security and Communication Networks
Volume 2022, Article ID 1557233, 9 pages
https://doi.org/10.1155/2022/1557233

WILEY | Hindawi

Research Article

An Improved Secure Public Cloud Auditing Scheme in Edge Computing

Zhengge Yi, Lixian Wei, Haibin Yang, Xu An Wang, Wenyong Yuan, and Ruifeng Li

Engineering University of PAP, Xi'an, China

Correspondence should be addressed to Xu An Wang; wangxazjd@163.com

Received 4 November 2021; Revised 24 January 2022; Accepted 28 February 2022; Published 26 April 2022

Academic Editor: Xiaolong Xu

Cloud storage plays an important role in the data processing of edge computing. It is very necessary to protect the integrity of these data and the privacy of users. Recently, a cloud auditing scheme which can be used to smart cities has been proposed, which is lightweight and privacy-preserving. Although this scheme has very good performance and is a very valuable work, we find that there is insecurity in it. By giving two kinds of attacks, we prove that a malicious cloud server provider (CSP) can forge auditing proof and can successfully pass the verification of the third-party auditor (TPA) even if the CSP deletes the user's data. Then, based on this scheme, we propose an improved scheme, which can resist the forgery attack from malicious CSP. Through security analysis, our scheme improves the security compared to the original scheme without reducing the efficiency.

1. Introduction

The rise of the Internet of Things and the 5G network has led to many new services, including intelligent transportation, smart city, location service, and so on [1, 2]. The number of smartphones, wearable devices, Internet-connected televisions, and other sensor devices shows an explosive growth trend, followed by "sea-scale" data generated by these Internet of Things terminals [3–6].

In edge computing, some or all of the private data of end users need to be outsourced to third parties (such as cloud computing data centers and edge data centers) [7–9]. By using the cheap storage and computing services provided by the cloud server, users with limited resources can be freed from the complex hardware system, reduce the storage burden, and at the same time be able to easily access their own data [10–12]. Compared to the traditional cloud computing model which relies solely on the computing center, the edge computing can handle the big data at the network edge effectively.

However, the users' data stored in the third-party data center have the features of separation of control, storage randomization, and so on, which can easily lead to data security problems such as data loss, data leakage, and so on [13, 14]. When the integrity of users' data is destroyed, the interests of these users may receive huge losses. Therefore, it is significant to design a cloud auditing scheme for edge computing.

1.1. Related Work. Recently, in order to meet different application requirements, various cloud storage audit schemes have been proposed. At present, the research on data integrity audit is mainly focused on four functional requirements, namely, dynamic audit, batch audit, privacy protection, and lightweight computing.

At the CCS conference in 2007, Jules and Ateniese et al. proposed proofs of retrievability (POR) and provable data possession (PDP), respectively, to audit cloud storage data [15, 16]. Both of them use the idea of sampling testing to audit the integrity of the data. That is, only a small part of the data in the cloud can ensure the integrity and reliability of all data with a high probability. Then, Ateniese et al. proposed a scalable PDP scheme based on the original PDP [17], which is the first verifiable data holding protocol that supports partial dynamic operation. The design of this protocol provides a new idea for the construction of the cloud audit protocol and takes an important step towards the more

practical PDP protocol. Inspired by Ateniese et al., Erway et al. [18] extended the above PDP protocol and designed a protocol that supports the dynamic update of cloud data. The audit protocol uses jump tables to support complete dynamic operation of data. Compared with the protocol of Ateniese et al., it has a greater breakthrough in practical value and the probability of detecting cloud data errors. However, the protocol does not have the performance of privacy protection, batch auditing, and so on.

Wang et al. [19] proposed a distributed data audit system to protect privacy in order to solve the problems of privacy disclosure and batch audit in the process of data integrity audit. The system uses a third-party audit platform to perform integrity audits, and the data owner can delete the local original data after the data are outsourced and stored to the cloud server. At the same time, homomorphic MAC and random mask technology are used to ensure that the third-party audit platform cannot know the content of the stored data in the effective audit process to achieve privacy protection. Subsequently, Wang et al. further improved the scheme in reference [20] by constructing a Merkle hash tree structure based on block authentication tags to improve the proof of the storage model. A study [20] further improved the bilinear aggregation signature method and improved the batch audit efficiency of TPA. Yang et al. [21] proposed an efficient and privacy-protected dynamic auditing protocol, which can be extended to realize dynamic data operations and batch auditing. At the same time, combining cryptography and bilinear properties, this scheme can protect the data privacy. In view of mobile devices with insufficient computing power, a lightweight integrity audit scheme supporting privacy protection is proposed in reference [22]. This scheme uses an online/offline signature method where the offline phase undertakes a lot of computing work. When the data file to be outsourced is given, the user just needs to construct the outsourced data signature in the online phase, which is lightweight.

1.2. Motivation. At present, in most public audit systems, in order to ensure the integrity of user data, the third-party auditor usually initiates an integrity challenge to the CSP, and then the CSP generates evidence to prove that it honestly stores user data. In this model, we first need to ensure that the cloud service provider cannot complete the forgery attack; that is, the forged evidence cannot be verified by the third-party auditor.

Recently, a public cloud auditing scheme has been proposed by Jing Han et al. [23]. This scheme is pairing-free and allows a third-party auditor to generate authentication metasets on behalf of users, which can achieve lightweight computing. It can protect the privacy of a user's data by blinding the raw data before storing them in the CSP and sending to the third-party auditor. At the same time, this scheme can realize batch auditing. Their proposed scheme is very valuable.

However, we find this scheme is not secure. A malicious CSP can easily forge auditing proof. Even if the CSP deletes all the data of a user, it can still generate the correct data possession proof to pass the verification of TPA. According to our findings, we have carried out the following work:

(1) We give two attack methods to prove the insecurity of Han's scheme. The first attack proves that the audit proof can be forged by the CSP, and the second attack proves that the CSP can pass the verification of the TPA even if it deletes the user's data.

(2) Based on the original scheme, we propose an improved scheme, which can effectively resist the forgery attack from CSP.

2. System Model and Design Goals

2.1. The System Model. The cloud storage system (CSS) includes three entities as depicted in Figure 1: users, CSP, and TPA. The specific definitions are as follows:

(1) *Users*: the owner of the data, outsources the data to the CSP for storage, and delivers the audit work to the TPA.

(2) *CSP*: a provider of cloud storage services, has large storage space and powerful computing capabilities, and can realize data sharing.

(3) *TPA*: the third-party auditor, generates the authentication metaset for users' data and audits the integrity of data stored in the cloud for users.

As depicted in Figure 2, the workflow of this scheme is as follows:

(1) When a user needs to store a data file in the cloud server, they blind it and send the blinded data file to the CSP and TPA. Then, they delete the local data;

(2) After receiving the blinded data from user, the TPA generates the tags for the data and sends it to the CSP;

(3) In order to ensure whether their data is correctly stored in the cloud server, the user sends an auditing request to TPA;

(4) Upon receiving the auditing request, the TPA randomly selects a small set of data blocks as the audit objects and sends an auditing challenge to the CSP;

(5) The cloud server, based on the challenge and the authentication metaset, generates a proof and sends it to the TPA.

(6) After receiving the proof, the TPA verifies the correctness of it. Finally, the TPA sends the auditing report to the user.

2.2. The Design Goals. Our cloud storage audit scheme would achieve the requirements of public auditability, correctness, and unforgeability.

(1) *Pubic auditability*: TPA can replace the user to remotely audit the integrity of the data when the user does not need to download the data stored in the cloud.

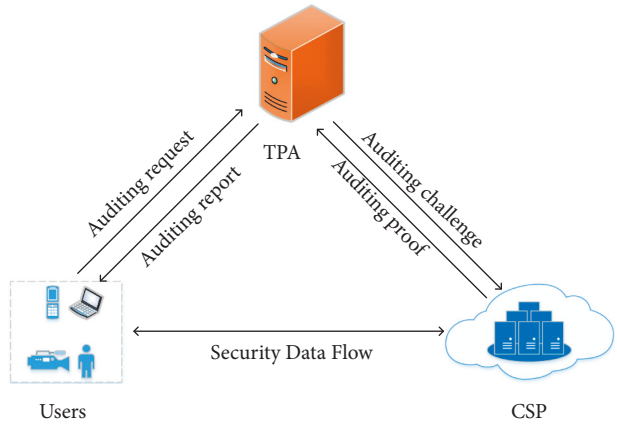

Figure 1: System model.

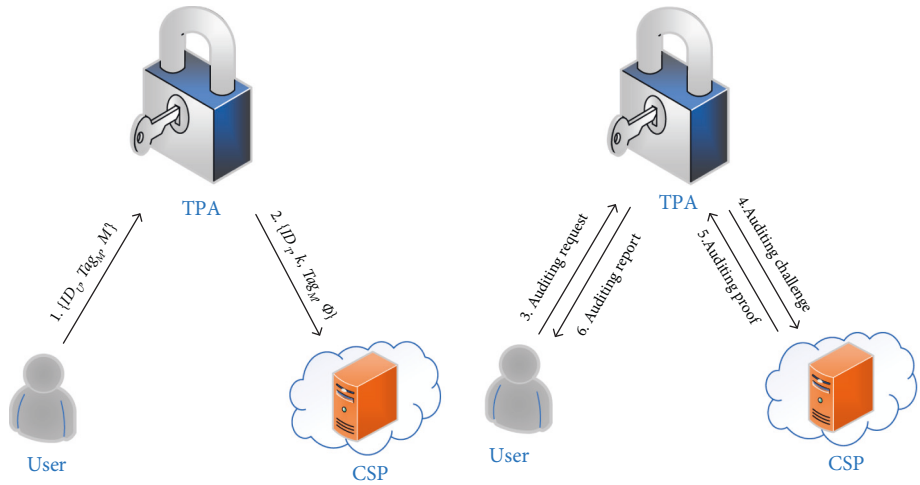

Figure 2: Workflow.

(2) *Correctness*: if CSP honestly stores user data, it can be audited by TPA. Otherwise, the generated proof cannot be verified by TPA.

(3) *Unforgeability*: any party cannot forge the authentication meta set of a user's data unless it has the user's secret key.

3. Review of Han's Scheme

Jing Han et al. (2020) proposed a public cloud auditing scheme, which consists of six algorithms as follows. Before reviewing this scheme, we first introduce the concept of HomMAC (homomorphic message authentication code). For a more specific definition, please refer to [24]. For the specific descriptions of the symbols that appear below, please refer to Table 1.

Given a data block $m_i = (m_{i1}, m_{i2} \ldots, m_{is}) \in Z_q^s$, then computes

$$\rho_i = \sum_{l=1}^{s} \omega_l m_{il} + \varpi_i, \tag{1}$$

where ρ_i is the HomMAC of m_i and $\omega_1, \omega_2, \ldots, \omega_s \in Z_q$, $\varpi \in Z_q$.

(1) *Setup*: input a security parameter λ , and then outputs p, q , which are two large primes. The CSS selects $h(\cdot)$: $\{0,1\}^* \longrightarrow Z_q$ and G . The CSS sets a PRG: $\mathfrak{R}_{prg} \longrightarrow Z_q^s$ and PRF: $\mathfrak{R}_{prf} \times \Gamma \longrightarrow Z_q$. Besides, the CSS set two time upper limits Δ_S and Δ_A, where Δ_S is the longest time for CSP to generate auditing proof, Δ_A is the longest time for the TPA to generate authentication meta set. Finally, the $cp = \{p, q, G, g, \text{PRF}, \text{PRG}, h(\cdot), \Delta_S, \Delta_A\}$ is made public.

(2) *KeyGen*: the identifier of TPA is $\text{ID}_T \in Z_q$ and $\text{ID}_U \in Z_q$. TPA generate their secret $sk_T \in Z_q^*$ and public key $pk_T = g^{sk_T}$. The user generates this secret/public key pair $(sk_u pk_u)$ from cp . Besides, the user chooses s random values $\alpha_1 \alpha_2 \ldots, \alpha_s \in Z_q^*$ and keeps them secret.

(3) SigGen:

3.1. SigGen1. First, the user divided file M into n data blocks. Then, divided each data block into s segments. Then, they establishe a unique tag $\text{Tag}_M = \text{name} \| \text{SSig}_{skU}(\text{name})$ for the file M.

Table 1: Symbols.

Symbols	Descriptions
G	A multiplicative cyclic group.
h	A secure hash function such that $h(\cdot)$: $\{0,1\}^* \longrightarrow Z_p$.
Z_q	A prime field.
q	The order of group G.
g	The generator of group G.
Γ	The index set of data blocks.
$m = \{(m_{11}, ..., m_{ns})\}$	The user's data file with n blocks and s slices.
α	The secret random values of user.
Δ	Time upper limit.
k	a key pair, where $k_g \in \mathfrak{R}_{prg}$ and $k_f \in \mathfrak{R}_{prf}$.
ID_U	The identifier of user.
ID_T	The identifier of TPA.
sk_U	The secret key of user.
pk_U	The public key of user.
sk_T	The secret key of TPA.
pk_T	The public key of TPA.
t	Time stamp.
Tag_M	The unique tag of the file M.
σ_i	The authentication label for the i-th data block.
Φ	The authentication meta set of data blocks.
chal	The challenge set.
p	The proof generated by CSP or auditor.

The user blinds each data blocks to protect the privacy of the file M as follows:

Chooses a random value $u \in_R G$ and then compute $\beta_l = u^{\alpha_l} \in G$ and $\varphi_l = h(\beta_l)$, where $l = 1, 2, \ldots, s$. Blind each data block m_i:

$$\begin{aligned} m'_{il} &= (\alpha_l m_{il} + \varphi_l) \bmod q, \quad l = 1, 2, \ldots, s, \\ m'_i &= (m'_{i1}, m'_{i2}, \ldots, m'_{is}). \end{aligned} \tag{2}$$

The blinded file is $M' = (m'_1, m'_2, \ldots, m'_n)$.

Finally, the user sends $\{M', \mathrm{Tag}_M, \mathrm{ID}_U\}$ to TPA and sends $\{\mathrm{ID}_U, \mathrm{Tag}_M, M', t_{s1}\}$ to CSP.

3.2. SigGen2. The TPA choose a key pair $k = (k_g, k_f)$, where $k_g \in \mathfrak{R}_{prg}$ and $k_f \in \mathfrak{R}_{prf}$. Then, they compute

$$\begin{aligned} \omega = (\omega_1, \omega_2, \ldots, \omega_s) &\longleftarrow \mathrm{PRG}\left(k_g\right), \omega_1, \omega_2, \ldots, \omega_s \in Z_q, \\ \varpi_i &\leftarrow \mathrm{PRF}\left(k_f\right), \varpi_1, \varpi_2, \ldots, \varpi_n \in Z_q. \end{aligned} \tag{3}$$

and the HomMAC:

$$\rho_i = \sum_{l=1}^{s} \omega_l m'_{il} + \varpi_i, i = 1, 2, \ldots, n. \tag{4}$$

The TPA compute $r_i = g^{\eta_i}$ and $s_i = (r_i \eta_i + \rho_i sk_T) \bmod q$, and then output $\sigma_i = (r_i, s_i)$, where $\eta_i \in Z_q^*$ is random value. Let $\Phi = \{\sigma_i\}$ be the authentication meta set of data blocks m'_i. Then, $\{\mathrm{ID}_T, k, \mathrm{Tag}_M, \Phi\}$ is sent to the CSP.

3.3. Storage. The CSP stores file M'.

When receiving the data from TPA, the CSP records time stamp t_{s2}, and computes:

$$\Delta'_s = t_{s2} - t_{s1}. \tag{5}$$

If $\Delta'_S > \Delta_S$, the CSP refuse to store data. Otherwise, they store data.

Next, the CSP computes the validity of Φ by performing the following computations:

$$\omega = (\omega_1, \omega_2, \ldots, \omega_s) \leftarrow \mathrm{PRG}\left(k_g\right), \quad \omega_1, \omega_2, \ldots, \omega_s \in Z_q, \tag{6}$$

$$\varpi_i \leftarrow \mathrm{PRF}\left(k_f\right), \varpi_1, \varpi_2, \ldots, \varpi_n \in Z_q, \tag{7}$$

$$g^{s_i} = r_i^{r_i} \cdot pk_T^{\sum_{l=1}^{s} \omega_l \cdot m'_{il} + \varpi_i} \bmod p. \tag{8}$$

If the (8) holds, the CSP stores the file and other information.

3.4. Challenge. The user sends an auditing request to the TPA. If it is validity, the TPA generates an auditing challenge chal as follows:

The TPA randomly chooses c elements as a subset $I \in \Gamma$ and chooses a random value $v_i \in Z_q^*$. Then, output $\mathrm{chal} = \{(i, v_i)\}$. Finally, they send the chal to the CSP.

3.5. ProofGen. The CSP computes:

$$\begin{aligned} R &= \prod_{i \in I} r_i^{v_i s_i} \bmod q, \\ S &= \sum_{i \in I} v_i s_i \bmod q, \\ \mu_l &= \sum_{i \in I} v_i m'_{il} \bmod q, \quad l = 1, 2, \ldots, s. \end{aligned} \tag{9}$$

Then, the proof $\{\mu, R, S\}$ is sent to the TPA, in which $\mu = (\mu_1, \mu_2, \ldots, \mu_s)$.

ProofVer: after receiving the proof, the TPA records time stamp t_{A2} immediately and computes $\Delta_A^{'} = t_{A2} - t_{A1}$. If $\Delta_A^{'} > \Delta_A$, stop audit work and return "*Expiration*" to the CSP. Otherwise, proceed to the following steps.

Compute:

$$\tau = \left(\sum_{l=1}^{s} (\omega_l \mu_l) + \sum_{i \in I} (v_i \varpi_i) \right) \bmod q. \tag{10}$$

Then, verify the following equation:

$$g^s = R \cdot pk_T^{\tau} \bmod p. \tag{11}$$

If the (11) does not hold, the TPA concludes that the user's data is corrupted. Otherwise, the TPA believe the user's data is integrity. Finally, the TPA sends the auditing report to the user.

3.6. Attack I. In this section, we will show the scheme of Jing Han et al. is not secure by giving the attack I. From the protocol of *ProofVer*, we can know that the TPA verifying the integrity of the data stored in the CSP by determines whether the following equation holds:

$$g^{s'} = R \cdot pk_T^{\tau} \bmod p. \tag{12}$$

Through observation, we can obtain the following information:

(1) The pk in this equation is the public key of user, which can be obtained by the CSP

(2) $\tau = (\sum_{l=1}^{s} (\omega_l \mu_l) + \sum_{i \in I} (v_i \varpi_i)) \bmod q$, the CSP can obtain the ω_l, ϖ_i and v_i , and the μ is computed by CSP

(3) The S and R is generated by CSP

Through the abovementioned points, the CSP can forge an auditing proof. The specific process is as follows:

(1) In the *audit phase*, after receiving the chal from TPA, the CSP randomly chooses s numbers as $\mu_l^{'} \in Z_q$, $l = 1, 2, \ldots, s$.

(2) The CSP computes the $\tau^{'}$ based on the $\mu_l^{'} \in Z_q$, $l = 1, 2, \ldots, s$:

$$\tau^{'} = \left(\sum_{l=1}^{s} (\omega_l \mu_l^{'}) + \sum_{i \in I} (v_i \varpi_i) \right) \bmod q, \tag{13}$$

and then computes the value of pk_T^{τ}.

(3) The CSP randomly selects a number as $S\prime \in Z_q$, and computes $g^{S\prime}$

(4) With the value of pk_T^{τ} and $g^{S\prime}$, the CSP computes $R\prime$:

$$R^{'} = \left(\frac{g^{s'}}{pk_T^{\tau}} \right) \bmod q, \tag{14}$$

(5) Finally, the CSP generates the forged proof $p^{'} = \{\mu^{'}, R^{'}, S^{'}\}$ and sends it to the TPA, where $\mu\prime = (\mu_1^{'}, \mu_2^{'}, \ldots, \mu_s^{'})$.

3.7. Attack II. Our attack II is based on this observation because the CSP can forge the auditing proof without using the blinded data of the user, which has been proved in the attack I. The malicious CSP can even delete the data stored in the cloud server but can still pass the verification of the TPA. Concretely, the attack is as follows:

(1) In the *storage phase*, after receiving the message from the TPA, the CSP verifies the validity of it and the correctness of Φ as the original scheme. If the message is valid and the Φ is correct, the CSP computes:

$$\omega = (\omega_1, \omega_2, \ldots, \omega_s) \leftarrow \mathrm{PRG}(k_g), \quad \omega_1, \omega_2, \ldots, \omega_s \in Z_q. \tag{15}$$

$$\varpi_i \leftarrow \mathrm{PRF}(k_f), \varpi_1, \varpi_2, \ldots, \varpi_n \in Z_q, \tag{16}$$

then the CSP deletes the user's data file.

(2) In the *audit phase*, to verify the integrity of the data in the CSP, TPA sends a challenge chal to the CSP. Upon receiving the chal, the CSP generates an auditing proof $p^{'} = \{\mu^{'}, R^{'}, S^{'}\}$ according to the method in the attack II. Note that the user's data are not stored when the CSP generates proof at this time

(3) After receiving the proof $p^{'}$ from the CSP, the TPA verifies the correctness of $p^{'}$. First, the TPA generates $\omega = (\omega_1, \omega_2, \ldots, \omega_s), \omega_l \in Z_q \neq$ and $\varpi_i \in Z_q$, where $l = 1, 2, \ldots, s$. Then they compute

$$\tau\prime = \left(\sum_{l=1}^{s} (\omega_l \mu_l^{'}) + \sum_{i \in I} (v_i \varpi_i) \right) \bmod q, \tag{17}$$

based on the $\mu_l^{'}$ from the CSP. Finally, the TPA verifies the whether the following equation holds:

$$g^{s'} = R^{'} \cdot pk_T^{\tau'} \bmod p. \tag{18}$$

Because the $S\prime$ and $R\prime$ in the $p\prime$ all are generated by the CSP, here we can prove the forged proof $p\prime$ is a valid one for the eq. holds:

$$\begin{aligned} g^{s'} &= R^{'} \cdot pk_T^{\tau'} \bmod p \\ &= \left[\left(\frac{g^{s'}}{pk_T^{\tau}} \right) pk_T^{\tau'} \right] \bmod p, \\ &= g^{s'}. \end{aligned} \tag{19}$$

4. Our Improved Scheme

In order to resist the abovementioned attack, in this section, we give our improved security scheme. The details of this scheme are as follows.

4.1. A Single-User Scenario. Based on the original scheme, our scheme consists of six algorithms: Setup, KeyGen, SigGen, Challenge, ProofGen, and ProofVer.

(1) *Setup*: the cloud storage system (CSS) inputs a security parameter λ, and then outputs p, q, which are two large primes. The CSS chooses a secure hash function $h(\cdot)$: $\{0,1\}^* \longrightarrow Z_q$ a multiplicative cyclic group G, where the order of G is q and the generator of G is g. The CSS sets a PRG: $\mathfrak{R}_{prg} \longrightarrow Z_q^s$ and PRF: $\mathfrak{R}_{prf} \times \Gamma \longrightarrow Z_q$. $\Gamma = \{1, 2, \ldots, n\}$ is the index set of data blocks. Besides, the CSS set two time upper limits Δ_S and Δ_A, where Δ_S is the longest time for CSP to generate auditing proof, Δ_A is the longest time for the TPA to generate authentication meta set. Finally, the $cp = \{p, q, G, g, \mathrm{PRF}, \mathrm{PRG}, h(\cdot), \Delta_S, \Delta_A\}$ is made public.

(2) *KeyGen*: the identifier of TPA is $\mathrm{ID}_T \in Z_q$ and $\mathrm{ID}_U \in Z_q$. TPA generate their secret $sk_T \in Z_q^*$ and public key $pk_T = g^{sk_T}$. The user generate this secret/public key pair (sk_u, pk_u) from cp. Besides, the user chooses s random values $\alpha_1, \alpha_2, \ldots, \alpha_s \in Z_q^*$ and keeps them secret.

(3) *SigGen*: this algorithm is run by user, TPA, and CSP, including three subalgorithms SigGen1, SigGen2, and storage.

4.1.1. SigGen 1. The user processes the file and generates the tags of data blocks.

First, the user divided file M into n data blocks and each data block is divided into s segments.

$$\begin{aligned} M &= \{m_1, m_2, \ldots, m_n\}, \\ m_i &= m_{i1}, m_{i2}, \ldots, m_{is}, \quad i \in 1, 2, \ldots, n. \end{aligned} \tag{20}$$

Then, they establishe a unique tag $\mathrm{Tag}_M = \mathrm{name} \| \mathrm{SSig}_{skU}(\mathrm{name})$ for the file M, where the $\mathrm{SSig}_{skU}(\mathrm{name})$ is the signature of the file's name using sk_U.

The user blinds each data blocks to protect the privacy of the file M as follows:

Chooses a random value $u \in_R G$ and then compute $\beta_l = u^{\alpha_l} \in G$ and $\varphi_l = h(\beta_l)$, where $l = 1, 2, \ldots, s$. Blind each data block m_i:

$$\begin{aligned} m'_{il} &= (\alpha_l m_{il} + \varphi_l) \mathrm{mod} q, \; l = 1, 2, \ldots, s, \\ m'_i &= (m'_{i1}, m'_{i2}, \ldots, m'_{is}). \end{aligned} \tag{21}$$

The blinded file is $M\prime = (m'_1, m'_2, \ldots, m'_n)$.

Finally, the user sends $\{\mathrm{ID}_U, \mathrm{Tag}_M, M\prime\}$ to the TPA and sends $\{\mathrm{ID}_U, \mathrm{Tag}_M, M\prime, t_{s1}\}$ to CSP.

4.1.2. SigGen 2. The TPA generates authentication meta set for the user.

The TPA choose a key pair $k = (k_g, k_f)$, where $k_g \in \mathfrak{R}_{prg}$ and $k_f \in \mathfrak{R}_{prf}$. Then, they compute

$$\begin{aligned} \omega = (\omega_1, \omega_2, \ldots, \omega_s) &\leftarrow \mathrm{PRG}(k_g), \omega_1, \omega_2, \ldots, \omega_s \in Z_q, \\ \varpi_i &\leftarrow \mathrm{PRF}(k_f), \varpi_1, \varpi_2, \ldots, \varpi_n \in Z_q, \end{aligned} \tag{22}$$

and the HomMAC:

$$\rho_i = \sum_{l=1}^{s} \omega_l m'_{il} + \varpi_i, \; i = 1, 2, \ldots, n. \tag{23}$$

The TPA compute $r_i = g^{\eta_i}$ and $s_i = (r_i \eta_i + \rho_i sk_T) \mathrm{mod} q$, and then output $\sigma_i = (r_i, s_i)$, where $\eta_i \in Z_q^*$ is random value. Let $\Phi = \{\sigma_i\}$ be the authentication meta set of data blocks m'_i for $i = 1, 2, \ldots, n$. Then, the TPA send $\{\mathrm{ID}_T, k, \mathrm{Tag}_M, \Phi\}$ to the CSP and delete the file $M\prime$ from their local record.

4.1.3. Storage. The CSP stores file M'.

When receiving the data from TPA, the CSP records time stamp t_{s2}, and computes:

$$\Delta'_S = t_{s2} - t_{s1}. \tag{24}$$

If $\Delta'_S > \Delta_S$, the CSP refuse to store data. Otherwise, they store data.

Next, the CSP computes the validity of Φ by performing the following computations:

$$\begin{aligned} \omega = (\omega_1, \omega_2, \ldots, \omega_s) &\leftarrow \mathrm{PRG}(k_g), \omega_1, \omega_2, \ldots, \omega_s \in Z_q, \\ \varpi_i &\leftarrow \mathrm{PRF}(k_f), \varpi_1, \varpi_2, \ldots, \varpi_n \in Z_q. \end{aligned} \tag{25}$$

$$g^{s_i} = r_i^{r_i} \cdot pk_T \sum_{l=1}^{s} s\omega_l \cdot m'_{il} + \varpi_i \, \mathrm{mod} p. \tag{26}$$

If the (26) holds, the CSP returns "Correct" to the user and stores the file, the file tag Tag_M and Φ. Otherwise, the CSP does not store the file and returns "Error" to the user.

4.1.4. Challenge. The user sends an auditing request to the TPA. If it is validity, the TPA generates an auditing challenge chal as follows:

The TPA randomly chooses c elements as a subset $I \in \Gamma$ and chooses a random value $v_i \in Z_q^*$ for each element $i \in I$. Then, output chal $= \{(i, v_i)\}$ for $i \in I$. Finally, they send the chal to the CSP and record the time stamp t_{A1} immediately.

4.1.5. ProofGen. After receiving the chal, the CSP computes the proof p.

The CSP computes:

$$\begin{aligned} S &= \sum_{i \in I} v_i s_i \mathrm{mod} q, \\ \mu_l &= \sum_{i \in I} v_i m'_{il} \mathrm{mod} q, \; l = 1, 2, \ldots, s. \end{aligned} \tag{27}$$

Then, they generates the proof $p = \{\mu, S\}$ and sends it to the TPA, where $\mu = (\mu_1, \mu_2, \ldots, \mu_s)$. Note that CSP no longer needs to generate R an element of audit proof.

4.1.6. ProofVer. After receiving the proof, the TPA records time stamp t_{A2} immediately and computes $\Delta'_A = t_{A2} - t_{A1}$. If $\Delta'_A > \Delta_A$, stop audit work and return "*Expiration*" to the CSP. Otherwise, proceed to the following steps.

Compute:

$$R = \prod_{i \in I} r_i^{v_i s_i} \bmod q,$$
$$\tau = \left(\sum_{l=1}^{s} (\omega_l \mu_l) + \sum_{i \in I} (v_i \varpi_i) \right) \bmod q. \tag{28}$$

Then, verify the following equation:

$$g^s = R \cdot pk_T^{\tau} \bmod p. \tag{29}$$

If the (29) does not hold, the TPA concludes that the user's data are corrupted. Otherwise, the TPA believe the user's data are integrity. Finally, the auditing report is sent to the user.

4.2. A Multiuser Scenario. In edge computing, it is common for multiple end users to apply for an audit at the same time. Compared with the single-user scheme, batch auditing can reduce the computational consumption and thus improve the auditing efficiency. In this section, we extend the scheme in section 6.1 to the one that TPA can conduct batch auditing for multiple users.

Suppose there are N users. They send their auditing requests to the TPA. In the three phases of Setup, KeyGen and SigGen, users, TPA and CSP do the same as described in section 6.1.

(1) *Challenge*: upon receiving the auditing requests, the TPA randomly chooses c elements as a subset $I \in \Gamma$ and chooses a random value $v_i \in Z_q^*$ for each element $i \in I$. Then, output chal $= \{(i, v_i), Ms\}$, where Ms includes the message of the N users. Finally, they send the chal to the CSP and record the time stamp t_{A1} immediately.

(2) *ProofGen*: after receiving the chal from TPA, CSP perform the following calculations:

$$S = \sum_{\theta=1}^{N} \sum_{i \in I} v_i s_i^{(\theta)} \bmod q, \quad \theta = 1, 2, \ldots, \mathrm{N},$$
$$\mu_l^{(\theta)} = \sum_{i \in I} v_i m_{il}^{(\theta)} \bmod q, \quad l = 1, 2, \ldots, s. \tag{30}$$

Then, the TPA send auditing proof $p = \{\mu^{(\theta)}, S\}$ to the CSP, where $\mu^{(\theta)} = (\mu_1^{(\theta)}, \mu_2^{(\theta)}, \ldots, \mu_s^{(\theta)})$.

(3) *ProofVer*: after receiving the proof, the TPA records time stamp t_{A2} immediately and computes $\Delta_A' = t_{A2} - t_{A1}$. If $\Delta_A' > \Delta_A$, stop audit work and return "*Expiration*" to the CSP. Otherwise, the TPA computes:

$$\omega^{(\theta)} = \left(\omega_1^{(\theta)}, \omega_2^{(\theta)}, \ldots, \omega_s^{(\theta)}\right) \leftarrow \mathrm{PRG}\left(k_g^{(\theta)}\right)$$
$$\omega_1^{(\theta)}, \omega_2^{(\theta)}, \ldots, \omega_s^{(\theta)} \in Z_q \leftarrow \mathrm{PRF}\left(k_f^{(\theta)}, i^{(\theta)}\right). \tag{31}$$

Then, compute:

$$R = \prod_{\theta=1}^{N} \prod_{i \in I} r_i^{v_i s_i} \bmod q,$$
$$\tau^{(\theta)} = \left(\sum_{l=1}^{s} \left(\omega_l^{(\theta)} \mu_l^{(\theta)}\right) + \sum_{i \in I} \left(v_i \varpi^{(\theta)}\right) \right) \bmod q, \tag{32}$$
$$\widetilde{\tau} = \sum_{\theta=1}^{N} \tau^{(\theta)} \bmod q.$$

Finally, verify the following equation:

$$g^s = R \cdot pk_T^{\widetilde{\tau}} \bmod p. \tag{33}$$

If the (29) does not hold, the TPA concludes that the users' data is corrupted. Otherwise, the TPA believes the users' data is integrity. Finally, the auditing reports are sent to the users.

5. Security Analysis

In this section, we first prove the correctness of the improved scheme. Then, we prove that the auditing proof cannot be forged, which proves that our proposed scheme can resist attack I and attack II. The proof process of privacy preserving users' data can refer to Han's scheme.

5.1. Correctness. The correctness of verification (8) is proved as follows:

$$\begin{aligned} g^{s_i} &= g^{r_i \eta_i + \rho_i \cdot sk_T} \bmod p \\ &= g^{r_i \eta_i} + g^{\rho_i \cdot sk_T} \bmod p \\ &= r_i^{r_i} pk_T^{\sum_{l=1}^{s} \omega_l m_{il} + \varpi_i} \bmod p. \end{aligned} \tag{34}$$

The correctness of verification (11) is elaborated as follows:

$$\begin{aligned} g^S &= g^{\sum_{i \in I} v_i s_i} \bmod p \\ &= g^{\sum_{i \in I} v_i (r_i \eta_i + \rho_i \cdot sk_T)} \bmod p \\ &= R \cdot pk_T^{\sum_{l=1}^{s} \omega_l \sum_{i \in I} v_i m_{il}' + \sum_{i \in I} v_i \varpi_i} \bmod p \\ &= R \cdot pk_T^{\sum_{l=1}^{s} \omega_l \mu_l + \sum_{i \in I} v_i \varpi_i} \bmod p \\ &= R \cdot pk_T^{\tau} \bmod p. \end{aligned} \tag{35}$$

The correctness of verification (33) is proved in the following:

$$
\begin{aligned}
g^{S} &= g^{\sum_{\theta=1}^{N}\sum_{i\in I}\nu_i s_i^{(\theta)}} \bmod q \\
&= g^{\sum_{\theta=1}^{N}\sum_{i\in I}\nu_i\left(r_i^{(\theta)}\eta^{(\theta)}+\rho_i^{(\theta)}\cdot sk_T\right)} \bmod p \\
&= R\cdot\prod_{\theta=1}^{N} pk_T^{\sum_{i\in I}\nu_i\rho_i^{(\theta)}} \bmod p \\
&= R\cdot\prod_{\theta=1}^{N} pk_T^{\sum_{i\in I}\nu_i\left(\sum_{l=1}^{s}\omega_l^{(\theta)}m_{il}^{\prime(\theta)}\varpi_i^{(\theta)}\right)} \bmod p \\
&= R\cdot\prod_{\theta=1}^{N} pk_T^{\sum_{l=1}^{s}\omega_l^{(\theta)}\mu_l^{(\theta)}+\sum_{i\in I}\nu_i\varpi_l^{(\theta)}} \bmod p \\
&= R\cdot\prod_{\theta=1}^{N} pk_T^{\tau^{(\theta)}} \bmod p \\
&= R\cdot pk_T^{\widetilde{\tau}} \bmod p.
\end{aligned}
\tag{36}
$$

5.2. Unforgeability. In our improved scheme, a malicious CSP cannot forge a correct audit proof that can pass the verification of TPA.

Proof. the malicious CSP forge a proof $\widehat{p}=\left\{\widehat{\mu},\widehat{S}\right\}$. If it is valid, the (7) will hold.

$$
\begin{aligned}
g^{\widehat{s}} &= \widehat{R}\cdot pk_T^{\tau} \bmod p, \\
pk_T^{\tau} &= \frac{g^{\widehat{S}}}{\widehat{R}}.
\end{aligned}
\tag{37}
$$

Because P is valid, (38) must hold.

$$
\begin{aligned}
g^{s} &= R\cdot pk_T^{\tau} \bmod p, \\
pk_T^{\tau} &= \frac{g^{S}}{R}.
\end{aligned}
\tag{38}
$$

According to (37) and (38), we can get:

$$
\frac{g^{\widehat{S}}}{\widehat{R}}=\frac{g^{S}}{R}.
\tag{39}
$$

In the original scheme, both R and S are calculated by the CSP and sent to the TPA, so the CSP can easily calculates the value of g^S/R and then forges $g^{\widehat{S}}/\widehat{R}$ that makes the (39) hold according to the method in attack 1. However, in our improved scheme, R is generated by TPA, so the (40) must hold.

$$
g^{S-\widehat{S}}=R\widehat{R}^{-1}.
\tag{40}
$$

From the abovementioned equations, $S=\widehat{S}$ and $R=\widehat{R}$ must hold. Otherwise, we can easily get the value of $S-\widehat{S}$ when $g, g^{S-\widehat{S}}\in G$ is given. It means that there is a solution of a DLP instance in G. However, this contradicts to the proven DLP difficult problem. Therefore, a malicious CSP cannot forge a valid auditing proof to pass the verification of TPA. □

5.3. Privacy Preserving. The proposed scheme provides privacy preserving for users' data.

Proof. before sending the data to TPA and CSP, the user has blinded each data block by using random mask technique as follows:

$$
\begin{aligned}
m_{il}^{\prime} &= (\alpha_l m_{il}+\varphi_l)\bmod q, \quad l=1,2,\ldots,s, \\
m_i^{\prime} &= (m_{i1}^{\prime}, m_{i2}^{\prime},\ldots,m_{is}^{\prime}),
\end{aligned}
\tag{41}
$$

where $u\in_R G$, $\beta_l=u^{\alpha_l}\in G$, $\varphi_l=h(\beta_l)$, $l=1,2,\ldots,s$. The curious TPA or CSP may want to obtain some privacy information of user from the blinded data $M\prime$. Only know the value of $\alpha_l\in_R Z_q^*$ for $l=1,2,\ldots,s$ can they do that successfully. However, that computing α_l given $\beta_l=u^{\alpha_l}\in G$ is to solve the DLP in G, which is infeasible in calculation. Therefore, the curious TPA or CSP have no ability to get privacy information of user's data. □

6. Conclusion

In edge computing, it will do great harm to the running of terminal users if their data stored in the CSP can be deleted without being found. In this paper, we proved that Han's scheme is not secure because the cloud server provider can successfully forge auditing proof to prove to TPA that it honestly stores users' data. Then, we proposed an improved scheme that can effectively avoid the forgery attack from the cloud server.

In the future, cloud storage auditing schemes will be proposed to adapt to more different situations in edge computing, but we should give more attention to the security of the schemes.

Data Availability

The data of this article are available on request from the authors.

Conflicts of Interest

There are no potential conflicts of interest.

Authors' Contributions

Zhengge Yi and Lixian Wei contributed equally to this work. Zhengge Yi is responsible for the writing of the article and the construction of new scheme, Lixian Wei is responsible for the derivation of the formulas in the article and gives some significant ideas, Haibin Yang is responsible for the polishing of the language of the article, Xu An Wang gives the main ideas for the writing of this article, Wenyong Yuan is responsible for collecting the information related to this article, and Ruifeng Li is responsible for the verification of the security of this article.

Acknowledgments

This work is supported by the Foundation of Foundation of National Natural Science Foundation of China (No.

62172436), State Key Laboratory of Public Big Data (No. 2019BDKFJJ008), Engineering University of PAP's Funding for Scientific Research Innovation Team (No. KYTD201805), and Engineering University of PAP's Funding for Key Researcher (No. KYGG202011).

References

[1] L. Ren, Y. Laili, L. Xiang, and X. Wang, "Coding-based large-scale task assignment for industrial edge intelligence," *IEEE Transactions on Network Science and Engineering*, vol. 7, no. 4, pp. 2286–2297, 2020.

[2] M. Azrour, J. Mabrouki, A. Guezzaz, and Y. Farhaoui, "New enhanced authentication protocol for Internet of Things," *Big Data Mining and Analytics*, vol. 4, no. 1, pp. 1–9, 2021.

[3] L. Ren, Y. Liu, X. Wang, and J. Lu, "Cloud-edge-based lightweight temporal convolutional networks for remaining useful life prediction in IIoT," *IEEE Internet of Things Journal*, vol. 8, no. 16, Article ID 12587, 2021.

[4] L. Kong, L. Wang, W. Gong, C. Yan, Y. Duan, and L. Qi, "LSH-aware multitype health data prediction with privacy preservation in edge environment," *World Wide Web*, vol. 1, no. 9, 2021.

[5] R. Bi, Q. Liu, J. Ren, and G. Tan, "Utility aware offloading for mobile-edge computing," *Tsinghua Science and Technology*, vol. 26, no. 2, pp. 239–250, 2021.

[6] Y. Yi, Z. Zhang, L. T. Yang, X. Deng, L. Yi, and X. Wang, "Social interaction and information diffusion in social Internet of Things: dynamics, CloudEdge, traceability," *IEEE Internet of Things Journal*, vol. 8, no. 4, pp. 2327–4662, 2020.

[7] X. Xu, Z. Fang, J. Zhang et al., "Edge content caching with deep spatiotemporal residual network for IoV in smart city," *ACM Transactions on Sensor Networks*, vol. 17, no. 3, pp. 1–33, 2021.

[8] Z. He and J. Zhou, "Inference attacks on genomic data based on probabilistic graphical models," *Big Data Mining and Analytics*, vol. 3, no. 3, pp. 225–233, 2020.

[9] X. Xu, Q. Huang, H. Zhu et al., "Secure service offloading for Internet of vehicles in SDN-enabled mobile edge computing," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 6, pp. 3720–3729, 2021.

[10] X. Xie, X. Yang, X. Wang, H. Jin, D. Wang, and X. Ke, "BFSI-B: an improved K-hop graph reachability queries for cyber-physical systems," *Information Fusion*, vol. 38, no. 2, pp. 35–42, 2017.

[11] W. Zhang, X. Chen, and J. Jiang, "A multi-objective optimization method of initial virtual machine fault-tolerant placement for star topological data centers of cloud systems," *Tsinghua Science and Technology*, vol. 26, no. 1, pp. 95–111, 2021.

[12] G. Orsini, D. Bade, and W. Lamersdorf, "Computing at the mobile Edge: Designing Elastic Android Applications for Computation offloading," in *Proceedings of the 9th Conference on the Joint IFIP Wireless and Mobile Networking (WMNC'16)*, pp. 112–119, Colmar, France, July 2016.

[13] K. Yang and X. Jia, "Data storage auditing service in cloud computing: challenges, methods and opportunities," *World Wide Web*, vol. 15, no. 4, pp. 409–428, 2012.

[14] L. Qi, X. Wang, X. Xu, W. Dou, and S. Li, "Privacy-Aware cross-platform service recommendation based on enhanced locality-sensitive hashing," *IEEE Transactions on Network Science and Engineering*, vol. 8, no. 2, pp. 1145–1153, 2021.

[15] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson et al., "Provable Data Possession at Untrusted stores," in *Proceedings of the Acm Conference on Computer & Communications Security ACM*, Alexandria, Virginia, USA, October 2007.

[16] A. Juels and B. S. Kaliski, "PoRs: Proofs of Retrievability for Large Files," in *Proceedings of the ACM Conference on Computer and Communications Security (CCS' 07).ACM Press*, pp. 584–597, Alexandria, Virginia, USA, October 2007.

[17] G. Ateniese, R. D. Pietro, and L. V. Mancini, "Scalable and Efficient Provable Data possession," in *Proceedings of the 4th international conference on Security and Privacy in Communication Networks*, Istanbul Turkey, September 2008.

[18] C. C. Erway, A. Küpçü, C. Papamanthou, and T. Roberto, "Dynamic Provable Data Possession," in *Proceedings of the ACM Conference on Computer and Communications Security (CCS'09)*, pp. 17–38, ACM Press, IL, USA, November 2009.

[19] C. Wang, Q. Wang, K. Ren, and W. Lou, "Privacy-preserving public auditing for data storage security in cloud computing," in *Proceedings of the 29th IEEE Annual International Conference on Computer Communications (INFOCOM'10)*, pp. 1–9, San Diego, CA, USA, March 2010.

[20] Q. Wang, C. Wang, K. Ren, W. Lou, and J. Li, "Enabling public auditability and data dynamics for storage security in cloud computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 22, no. 5, pp. 847–859, 2011.

[21] K. Yang and X. Jia, "An efficient and secure dynamic auditing protocol for data storage in cloud computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 24, no. 9, pp. 1717–1726, 2013.

[22] J. Li, L. Zhang, J. K. Liu, H. Qian, and Z. Dong, "Privacy-preserving public auditing protocol for low-performance end devices in cloud," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 11, pp. 2572–2583, 2016.

[23] J. Han, Y. Li, and W. Chen, "A Lightweight And privacy-preserving public cloud auditing scheme without bilinear pairings in smart cities," *Computer Standards & Interfaces*, vol. 62, no. FEB, pp. 84–97, 2019.

[24] S. Agrawal and D. Boneh, "Homomorphic MACs: MAC-Based integrity for network coding," in *Proceedings of the International Conf. On Applied Cryptography and Network Security, Springer-Verlag*, pp. 292–305, Singapore, June 2009.