Hindawi
Wireless Communications and Mobile Computing
Volume 2021, Article ID 8886763, 13 pages
https://doi.org/10.1155/2021/8886763

Research Article

Improved Lightweight Cloud Storage Auditing Protocol for Shared Medical Data

Haibin Yang, Zhengge Yi, Xu An Wang, Yunxuan Su, Zheng Tu, and Xiaoyuan Yang

Engineering University of People's Armed Police, China

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

Received 17 August 2020; Revised 24 November 2020; Accepted 22 December 2020; Published 9 January 2021

Academic Editor: Arun K. Sangaiah

Now, it is common for patients and medical institutions to outsource their data to cloud storage. This can greatly reduce the burden of medical information management and storage and improve the efficiency of the entire medical industry. In some cases, the group-based cloud storage system is also very common to be used. For example, in an medical enterprise, the employees outsource the working documents to the cloud storage and share them to the colleagues. However, when the working documents are outsourced to the cloud servers, how to ensure their security is a challenge problem for they are not controlled physically by the data owners. In particular, the integrity of the outsourced data should be guaranteed. And the secure cloud auditing protocol is designed to solve this issue. Recently, a lightweight secure auditing scheme for shared data in cloud storage is proposed. Unfortunately, we find this proposal not secure in this paper. It's easy for the cloud server to forge the authentication label, and thus they can delete all the outsourced data when the cloud server still provide a correct data possession proof, which invalidates the security of the cloud audit protocol. On the basis of the original security auditing protocol, we provide an improved one for the shared data, roughly analysis its security, and the results show our new protocol is secure.

1. Introduction

With the rapid development of network information technology in the medical field, internet medical care has become a new medical mode, and the pace of information construction of medical and health undertakings has been quickened. It makes the vast amount of medical information transform from traditional paper documents to fast and convenient digital storage, which undoubtedly brings a lot of convenience to medical institutions and patients, and greatly improves the diagnosis and treatment efficiency and service level of the hospital. With the increasing promotion of "Internet + Medical", big data, cloud computing, and other technologies have been widely used in the medical field, which makes a large number of medical data generated in the process of collection, storage, and application that can be centrally stored in the cloud. However, due to the great value of medical big data, such as patient basic information and medical records, as well as unfair competition among medical institutions, and problems such as information disclosure, device intrusion, data abuse, and tampering may occur frequently. At the same time, cloud service providers are often dishonest. They may delete uncommonly used data in order to reduce storage costs or even do not report lost data for the sake of their reputation. These problems bring great threats and hidden dangers to the security of medical information. Therefore, how to protect the storage security of medical information has become an important part of Internet medical construction, and it is a difficult problem that the medical industry needs to face squarely and urgently to be solved. In recent years, the development of cloud secure audit technology can well solve many of these problems. Secure audit is a security mechanism independent of cloud service providers. Users or medical institutions only need to implement some kind of knowledge proof agreement with service providers to audit and check the data stored in the cloud, so as to ensure the security of medical data.

1.1. Related Work. Before the concept of cloud computing was put forward, people explored methods for auditing

remotely stored data. In 2004, Dewarte et al. [1] provide a remote storage integrity check scheme, which use RSA-based hash functions to implement integrity checks. This method requires the verifier to download the data locally and compare it with the signature after public key recovery. In the cloud storage environment, due to the large amounts of data, verifying the integrity of the data through the signature will cause a huge communication overhead. And for those archived data that need to be backed up or saved for a long time, it is very unreasonable to download all the data just for integrity check.

Thus, the cloud storage auditing mechanism is proposed to solve this challenge problem. It does not require the user to download the data locally, but only requires the service provider to calculate some integrity evidence by accessing the user's data, and then the cloud submit the evidence to the user for verification, which greatly reducing the communication overhead. More importantly, this conclusion of the data integrity obtained by users through auditing is more convincing than the announced conclusion by the service provider.

In cloud storage, integrity checking of outsourced data is a difficult problem to be solved. Only solving this problem effectively, the user's data security can be truly guaranteed. For this purpose, people have conducted extensive research on a cloud storage auditing which is not only efficient but secure.

Basing on the idea of sampling the remote data, Jules et al. [2] and Ateneses et al. [3] proposed the primitive of PoR (Proofs of Retrievability) and PDP (Provable Data Possession), which are both designed to verify the integrity of remote data. The commonality of the solutions is that the cloud service party only access part of the data (the accessed data is randomly formulated by the auditor) through some form of the challenge-response protocol, to generate a probabilistic proof and submit it to the auditor for verification. The difference is that PoR uses a combination of error correcting code in the scheme, so it can provide data recovery when data corruption is detected. Compared with the PoR scheme, the PDP scheme provides relatively weak security; that is, it only detects data corruption and does not guarantee the ability of data recovering. However, the design of the PDP is more flexible, making it more flexible when the solution is extended to support additional functions (such as dynamic data update, and fairness).

Since most of the current cloud storage auditing schemes are based on these two schemes [3–14], in the following, we mainly discuss these two audit schemes, as well as their derived schemes and applications [9, 15–18].

(1) PDP scheme. Based on the RSA ring and KEA-r assumption, Ateniese et al. proposed the PDP scheme in 2007. It is the first probabilistic audit scheme that is both safe and practical. The basic idea is to divide the file into blocks of fixed size based on Sebe's [19] scheme at first and then calculate a Homomorphic Verifiable Tag (HVT) for each data block. Finally, the user stores the original data blocks and the corresponding homomorphic tags on the server and deletes its local backup.

When initiating an audit request, the audit party randomly chooses a subset index of the original data blocks and sends the auditing request to the cloud server. The prover generates integrity proof based on these specified data blocks and their tags and sends the proof to the auditor for validation. The proof mainly includes two parts: one part is a linear combination generated from the specified data block, and the other part is the aggregate signature generated from the specified tag. The verifier checks the validity of the proof by verifying a certain consistency relationship between these two parts. Since only a small part of the original data needs to be accessed, the proof provided by the prover is a kind of probabilistic proof. If the proof is correct, it can ensure that each data block specified by the auditor is complete and guarantees the integrity of all original data blocks with a high confidence probability at the same time.

The HVT used in the PDP scheme and the homomorphic hash function used in the earlier scheme [20, 21] are essentially homomorphic functions. It has such a feature: the result of computing Homomorphic Verifiable Tag (HVT) of the sum of two messages is equal to the product of the results of Homomorphic Verifiable Tag (HVT) of these two messages. One obvious benefit of this is that, in the proof generation stage, the tags of multiple data blocks can be merged into one tag. By verifying the validity of this aggregated tag, the validity of each individual tag can be ensured, which greatly saves the bandwidth overhead.

In the PDP scheme, Ateniese et al. first proposed the concept of public verifiability for cloud storage auditing. The audit can be conducted by any third party auditor (TPA) or any other ones, which has special significance for the broader application of integrity auditing schemes in the cloud environment. Generally speaking, the computing power of data owner's equipment is limited, and the proof verification process in the audit scheme requires a lot of calculations. If the audit task can be delegated to a third party or any other ones trusted by the data owner, the burden on the data owner can be greatly reduced. However, it is worth noting that TPA is often regarded as the delegatee of the user, and the trust of the service provider is also limited. When there is a dispute between the TPA and the service provider about the proof, how to arbitrate fairly and effectively is also an important issue.

(2) PoR Scheme. Juels et al. proposed the PoR scheme in 2007 to detect the integrity of remote data and recover the data through error correction codes when data corruption is found. The main idea of PoR is to add a certain number of equal-length marker blocks to the data blocks and then disturb the positions of all data blocks and marker blocks through a random permutation algorithm. Because the content of the marked block is a random value without semantic meaning, they must be placed in the encrypted data block sequence, which make the marked block and the data block indistinguishable from each other and make it difficult for an attacker to identify the specific location of the marked block. During the audit, the auditor specifies the location of some marker blocks, and the prover sends these marker blocks to the auditor for verification to determine whether the data is integrity. This idea is based on

the fact that: the marked block and the ciphertext data block are indistinguishable, once a data block is damaged or lost, some marked blocks will also be damaged with a high probability.

The advantage of the Juels' PoR scheme is that it combines integrity detection and data recovery, which makes it possible to recover the damaged data block immediately when it is detected by the data block that is damaged and provide higher level data protection, but the use of erasure codes and the embedding of marker blocks will generally increase the amount of data by 15

(a) In order to make the original data blocks and the marked blocks indistinguishable, the original data must be encrypted. And it cannot be applied to the audit of plain text data, such as archived data (weather, space data etc.) which needs to be backed up for a long time

(b) The use of the marked data block is one-time. Once the location of the tag block is exposed, it cannot be used again in the next auditing, and each audits will expose a part of the location of the marked blocks, which make the total number of audits that can be conducted is restricted

(c) The strategy of encoding data with error correction code and then encrypting it which make the auditing scheme based on PoR be very difficult in supporting data block insertion, deletion, and modification

1.2. Motivation. Recently, a lightweight secure auditing scheme for shared data in cloud storage is proposed by Tian and Jing [1]. By introducing Hashgraph technology and designing a Third Party Medium (TPM) management strategy, their scheme achieves security management of the groups and a lightweight calculation for the group members. And by employing a blind method to blind the data, it protects the group members' privacy information. Unfortunately, we find their scheme not secure in this paper. Due to improper parameter settings of its signature algorithm, the adversary can easily forge authentication label. Even if all the outsourced data has been deleted by the cloud server, it can still give a correct data possession proof. The malicious cloud server can also modify the data blocks and the corresponding authentication labels arbitrarily without detection. At the same time, we noticed that the original scheme needs to be restored to the authentication label corresponding to the real data after the authentication label corresponding to the blind data is uploaded to the cloud server, resulting in a large amount of calculation. In order to solve these problems, we have improved the signature algorithm of the original scheme, specifically by changing the key parameter settings. In our solution, lightweight computing is also implemented, but the malicious cloud server cannot obtain the privacy information of the user key from the obtained authentication tag, so it is impossible to forge the authentication tag for the forged data block. If the cloud server dishonestly completes the storage work, it is impossible to pass the integrity verification of the third-party auditor. At the same time, we improved the processing method of blind data and its corresponding authentication tags on the cloud server, which reduced the computational complexity of the solution and improved the efficiency of the solution. With the increasing amount of medical data, a safe and efficient cloud storage solution is needed to manage this data, thereby reducing the storage burden of hospitals and patients. In order to solve this problem, we will design a management medical data based on our modified scheme to improve the efficiency and security of medical data cloud management. Our contribution can be summarized as follows:

(1) We first point out that Tian et al.'s lightweight auditing scheme for shared data in cloud storage is not secure. The authentication labels can be easily forged. We demonstrate two concrete attacks on their protocols

(2) We proposed an improved secure auditing protocol for shared data in cloud storage and analysis its security. The performance is also analyzed, and compared with other related work, the results show our protocol can be used in practical setting

1.3. Organization. We organize our paper as follows. In Section 2, we give the preliminaries which needs to understand our paper, including the mathematical tool, the definition, and security model for cloud storage auditing. In Section 3, we firstly review Tian et al.'s lightweight secure auditing scheme and show our attacks. In Section 4, we give our improved secure auditing protocol and roughly analyze its security. In the last section, we make our conclusion.

2. System Model

The system model of the lightweight secure cloud storage auditing protocol can be seen in Figure 1. There are four entities in this system model: group manager (GM), group member (M), the cloud (C), and the TPM.

For one group manager, there are multiple group members. After the data file is created by the data owner, he outsources it to the cloud server. And later,, the corresponding shared data can be accessed and modified by any group member. Here, the GM can be the original data owner. Here, we describe the functionality of four entities:

(1) Data storage services are provided by the cloud (C) for group members, and the cloud platform is also provided by the cloud (C) for group members to share data

(2) The group member (M) needs to complete the following tasks: (1) blind data, (2) record blind data, and broadcast it in the group

(3) The group manager (GM) needs to complete the following tasks: (1)the TPM management strategy

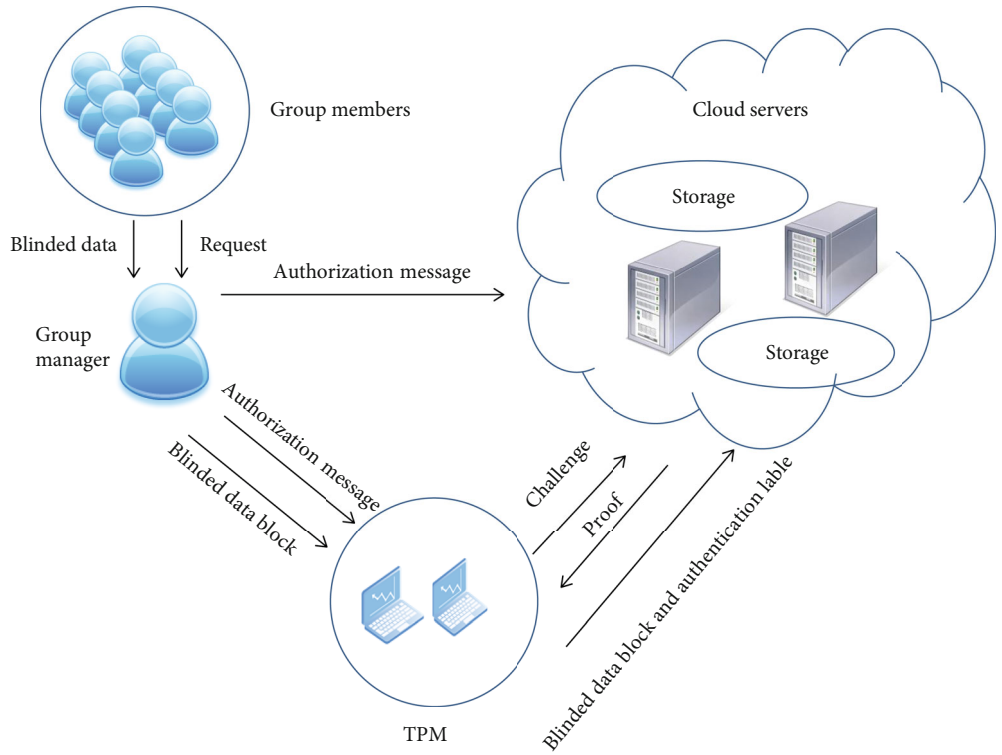

Figure 1: System model of the lightweight secure cloud storage auditing protocol.

should be given, (2) the TPM's public-private key pair is generated, (3) the secret seed is generated and used to blind the data for group members, and it is also used to recover the real data for the cloud

(4) The TPM needs to complete the following tasks: (1) it generates data authentication label for group members and (2) to verify the integrity of the cloud data on behalf of the group members

The execution of the cloud storage auditing protocol can be described as follows:

3. Data Upload Stage

(a) The group members (data owners) generate data and outsource it to the cloud server. The data is first blinded by the secret seed and recorded by the Hashgraph and then is sent to the group manager

(b) From the virtual TPM pool, a TPM for authorization is selected by the group manager according to the TPM management strategy; within the authorization time for these blinded data, the corresponding authentication labels is calculated by the authorized TPM

(c) Then, the authorized TPM sends the pair of blind data and authentication label to the cloud. Before receiving these messages, whether or not the authorization from the TPM is valid at the current time will be checked by the cloud

(d) If it is, the cloud will verify the authentication labels' correctness. The real data is recovered. If they are correct, their authentication labels are also computed. Finally, these real data and authentication labels are stored by the cloud

4. Audit Stage

(a) According to the TPM management strategy, a TPM is selected by the group manager, and it also creates the authorization

(b) Then, the challenge messages are sent to the authorized TPM by the cloud. Then, whether or not the authorization being valid from the TPM will be checked by the cloud. If it is, a proof of possession of the shared data is generated by the cloud

(c) Finally, by checking the correctness of the proof, the integrity of shared data in the cloud can be verified by the TPM

5. Review of Tian et al.'s Scheme

Before we review Tian et al.'s scheme, we give the symbols and the corresponding description in Table 1.

Table 1: Symbols.

Symbols	Description
G_1, G_2	Two multiplicative cyclic groups with a large prime order
H_1	A secure hash function such that $H_1 : \{0,1\}^* \times G_1 \rightarrow Z_p^*$
H_2	A secure hash function such that $H_2 : \{0,1\}^* \rightarrow G_1$
Z_p^*	A prime field with nonzero elements
g	The generator of group G_1
$m = \{(m_{11},\cdots,m_{ns}\}$	A shared data file with n blocks and s slices.
$m_{ij}{}'g$	The blind data block.
k_1	The secret seed (input key) of a pseudo-random function
ζ_{k_1}	A pseudo-random function such that $\zeta_{k_1} : Z_p^* \times Z_p^* \rightarrow Z_p^*$
α^j	The number of random values α is consistent with the number of slices j of the data block.
α_i	The blinding factor corresponding to the i-th data block such that $\alpha_i = \zeta_{k_1}(i, \text{name})$
β_i	TPM_i's private key used to generate data authentication label
$\theta_i{}'$	The authentication label for the i-th blind data block
θ_i	The authentication label for the i-th data block
d	The number of TPM
ID_{group}	The identity of the group manager

In Tian et al.'s scheme, there are four parties which are the group manager, the group members, the TPM, and the cloud. Concretely in their scheme, the following algorithms are involved:

(1) Key generation:

(a) A random $\beta_i \in Z_p^*$ is selected by the group manager as the TPM's private key. g^{β_i} is also calculated by him

(b) A random $k_1 \in Z_p^*$ is selected by the group manager and is sent to the group members and the cloud

(c) $\alpha^j \in Z_p^*$ is randomly selected by the group manager, g^{α^j} is also calculated, and the public key of TPM_i is computed as the following:

$$pk_{\text{TPM}_i} = \left(g^{\beta_i}, g^{\alpha^1\beta_i}, g^{\alpha^2\beta_i},\cdots,g^{\alpha^j\beta_i}\right) \quad (1)$$

(d) The interconnection function f and function sequence $f_i{}'$ are selected by the group manager, and it also sets the input sending and output sending window and then sends them to the cloud

(2) Data blind:

(a) The secret seed k_1 is used to calculate the blind factor $\alpha_i = \zeta_{k_1}(i, \text{name})$ by the group members, and the blind data $m_{ij}{}'$ is calculated as

$$m_{ij}{}' = m_{ij} + \alpha_i \quad (2)$$

(b) A request to upload the data $m_{ij}{}'$ is sent by a group member to the group manager, and it also calculates $\text{hash}(id_{i,j})$ (the hash value) and sends $id_{i,j}$, $m_{ij}{}'$, to the group manager securely. A new event is then created by the group member

(c) For the new event, a transaction record $\text{hash}(id_{i,j})$ will be used and will be broadcasted within the group. According to the same hash algorithm, the group manager verifies $\text{hash}(id_{i,j})$ after receiving the request. If the verification is passed, it receives $m_{ij}{}'$.

(3) Authorize:

(a) According to the TPM management strategy, the output port TPM_i in the virtual TPM pool is calculated by the group manager corresponding to the input port u_i (the requesting group member).

(b) The authorization message for TPM_i is generated by the group manager as follows:

$$\left\{\left(ID_{\text{group}} \| u_i \| \Delta t_i\right), \Delta t_i{}'\right\}, \quad (3)$$

where the group manager's identity is ID_{group}, denotes the time when the request is processed by the group manager as Δt_i, and denotes $\Delta t_i{}'$ as the time authorized for the *TPM* by the group manager.

(c) According to the authorization message, the group manager calculates the value H_1 as follows:

$$H_1 = H_1\left(\left(ID_{\text{group}} \| u_i \| \Delta t_i\right), \Delta t_i{}'\right) \tag{4}$$

(d) The authorization message is then sent by the group manager to the cloud and sends α^j, β_i, $m_{ij}{}'$ and H_1 to TPM_i

(4) Authentication label generation:

(a) Authentication label $\sigma_i{}'$ of $m_{ij}{}'$ is generated by TPM_i, after getting the blind data block $m_{ij}{}'$, is as the following:

$$\sigma_i{}' = \left(H_2(i) \cdot \prod_{j=1}^{s} \left(g^{\alpha^j}\right)^{m_{ij}{}'}\right)^{\beta_i}, \tag{5}$$

e.g., $m_5{}'$'s authentication label is generated as

$$\sigma_5{}' = \left(H_2(5) \cdot \prod_{j=1}^{s} \left(g^{\alpha^j}\right)^{m_{ij}{}'}\right)^{\beta_i} \tag{6}$$

Then, the data file $(m_{ij}{}', \sigma_i{}')$ and H_1 is sent by TPM_i to the cloud.

(b) After receiving the corresponding group manager's authorization message and the corresponding TPM_i's $(m_{ij}{}', \sigma_i{}')$, the output port TPM_i is first calculated by the cloud. If the message is just sent by TPM_i at $\Delta t_i{}'$, then H_1 $((ID_{\text{group}} \| u_i \| \Delta t_i), \Delta t_i{}')$ is calculated by the cloud, comparing with the value $H_1((ID_{\text{group}} \| u_i \| \Delta t_i), \Delta t_i{}')$ from TPM_i. If they are the same, run the next algorithm, otherwise stop the execution

(5) Authentication label check: the correctness of label $\sigma_i{}'$ is verified by the cloud as follows:

$$e\left(\sigma_i{}', g\right) = e\left(H_2(i) \cdot \prod_{j=1}^{s} \left(g^{\alpha^j}\right)^{m_{ij}{}'}, g^{\beta_i}\right) \tag{7}$$

$(m_{ij}{}', \sigma_i{}')$ is received and stored if the above equation is true, otherwise, rejected.

(6) Data recovery: based on k_1, the cloud calculates $\alpha_i = \zeta_{k_1}(i, \text{name})$, and using the following equation computes the real data m_{ij}:

$$m_{ij} = m_{ij}{}' - \alpha_i \tag{8}$$

According to pk_{TPM_i}, the real authenticator label σ_i is calculated by the cloud, i.e.,

$$\sigma_i = \sigma_i{}' \cdot \prod_{j=1}^{s} \left(g^{\alpha^j \beta_i}\right)^{-\alpha_i} = \left(H_2(i) \cdot \prod_{j=1}^{s} \left(g^{\alpha^j}\right)^{m_{ij}}\right)^{\beta_i} \tag{9}$$

Finally, the real data blocks $m_i = (m_{i1}, m_{i2}, \cdots, m_{\text{is}})$ and their real authenticator σ_i are stored by the cloud.

(7) Challenge. When a challenge is initiated by the group manager to the cloud, Δt is randomly selected by him as the authorization time to TPM_i, where u_i's sending window on the input side is Δt. An audit authorization command is sent by the group manager to the TPM_i through u_i at Δt, and $\{ID_{\text{group}} \| u_i \| \Delta t\}$ is sent by it as the audit authorization information to the cloud

The TPM_i implements the audit process after receiving the group manager's authorization command as the following:

(a) c blocks from all blocks of the shared data are randomly selected by TPM_i, and the indexes of the selected blocks are denoted as L

(b) Two random numbers $o, r \in Z_p^*$ are generated by TPM_i, and $X = g^o$ and $R = g^r$ are calculated

(c) $\{X^{\alpha^j}\}_{1 \le j \le s}$ is calculated by TPM_i

(d) The challenge information is outputted by TPM_i:

$$CM = \left\{L, R, \left\{X^{\alpha^j}\right\}_{1 \le j \le s}\right\} \tag{10}$$

Then, CM is sent by TPM_i to the cloud.

(8) Proof generation: after the challenge information CM is received, according to $\{ID_{\text{group}} \| u_i \| \Delta t\}$, the cloud first calculates the output port TPM. The authorization message $\{ID_{\text{group}} \| u_i \| \Delta t\}$ then is verified by the cloud. The proof of possessing shared data is generated by the cloud as follows:

(a) Subsets $L_1, \cdots, L_d$ are divided from the index set L, where the selected blocks that are signed by TPM_i are L_i

(b) $u_{ij} = \sum_{l \in L_i} m_{lj}$ and

$$\pi_i = \prod_{l \in L_i} e(\sigma_l, R) = e\left(\prod_{l \in L_i} H_2(l) \prod_{j=1}^{s} g^{\alpha^j \sum_{l \in L_i} m_{lj}}, g\right)^{\beta_i r}, \tag{11}$$

which are calculated by the cloud server for each subset L_i; here, $1 \le i \le d$ and $1 \le j \le s$.

(c) $w_i = \prod_{j=1}^{s} X^{\alpha^j u_{ij}}$ and $\pi = \prod_{i=1}^{d} \pi_i$ are calculated by the cloud servers, and then prf is the following

$$Prf = \left\{\{w_i\}_{1 \le i \le d}, \pi\right\}, \tag{12}$$

and it is sent to TPM_i as the proof.

(9) Proof check: based on the received prf and the challenge message CM, the correctness of the following equation is verified by TPM:

$$\prod_{i=1}^{d} e\left(\eta_i^o, pk_{\mathrm{TPM}_i}^r\right) e\left(w_i, pk_{\mathrm{TPM}_i}^r\right) = \pi^o, \tag{13}$$

where $\eta_i = \prod_{l \in L_i} H_2(l)$, $1 \le i \le d$. That is, it checks

$$\left(\prod_{i=1}^{d} e\left(\eta_i^o w_i, pk_{TPM_i}\right)\right)^r = \pi^o. \tag{14}$$

TPM_i outputs True if the equation is true, otherwise, outputs False.

6. Our Attack

6.1. Attack I. Our attack I is based on the following observation: the public key of TPM_i is

$$pk_{\mathrm{TPM}_i} = \left(g^{\beta_i}, g^{\alpha^1 \beta_i}, g^{\alpha^2 \beta_i}, \cdots, g^{\alpha^j \beta_i}\right), \tag{15}$$

and this public key is known to all; thus, the adversary can easily use it to forge authentication label. Concretely, the adversary launches the following attack:

(1) The adversary can observe many data block and their corresponding authentication labels by querying authentication label generation oracles, which is allowed in the security model of cloud auditing. That is, the adversary can get the following results:

$$\begin{aligned}
\sigma_1' &= \left(H_2(1) \cdot \prod_{j=1}^{s} \left(g^{\alpha^j}\right)^{m_{1j}'}\right)^{\beta_i};\\
\sigma_2' &= \left(H_2(2) \cdot \prod_{j=1}^{s} \left(g^{\alpha^j}\right)^{m_{2j}'}\right)^{\beta_i};\\
&\cdots\cdots\\
\sigma_n' &= \left(H_2(n) \cdot \prod_{j=1}^{s} \left(g^{\alpha^j}\right)^{m_{nj}'}\right)^{\beta_i}
\end{aligned} \tag{16}$$

(2) The above equations can be rewritten as

$$\begin{aligned}
\sigma_1' &= H_2(1)^{\beta_i} \cdot \left(\left(g^{\alpha^1}\right)^{\beta_i}\right)^{m_{11}'} \cdot \left(\left(g^{\alpha^2}\right)^{\beta_i}\right)^{m_{12}'} \cdots \left(\left(g^{\alpha^s}\right)^{\beta_i}\right)^{m_{1s}'};\\
\sigma_2' &= H_2(2)^{\beta_i} \cdot \left(\left(g^{\alpha^1}\right)^{\beta_i}\right)^{m_{21}'} \cdot \left(\left(g^{\alpha^2}\right)^{\beta_i}\right)^{m_{22}'} \cdots \left(\left(g^{\alpha^s}\right)^{\beta_i}\right)^{m_{2s}'};\\
&\cdots\cdots\\
\sigma_n' &= H_2(n)^{\beta_i} \cdot \left(\left(g^{\alpha^1}\right)^{\beta_i}\right)^{m_{n1}'} \cdot \left(\left(g^{\alpha^2}\right)^{\beta_i}\right)^{m_{n2}'} \cdots \left(\left(g^{\alpha^s}\right)^{\beta_i}\right)^{m_{ns}'}
\end{aligned} \tag{17}$$

(3) For the adversary knows the public key of TPM_i

$$pk_{TPM_i} = \left(g^{\beta_i}, g^{\alpha^1 \beta_i}, g^{\alpha^2 \beta_i}, \cdots, g^{\alpha^j \beta_i}\right), \tag{18}$$

$$m_{11}', m_{12}', \cdots, m_{1s}'; m_{21}', m_{22}', \cdots, m_{2s}'; \cdots\cdots; m_{n1}', m_{n2}', \cdots, m_{ns}'; \tag{19}$$

thus, it can compute

$$\begin{aligned}
H_2(1)^{\beta_i} &= \frac{\sigma_1'}{\left(\left(g^{\alpha^1}\right)^{\beta_i}\right)^{m_{11}'} \cdot \left(\left(g^{\alpha^2}\right)^{\beta_i}\right)^{m_{12}'} \cdots \left(\left(g^{\alpha^s}\right)^{\beta_i}\right)^{m_{1s}'}};\\
H_2(2)^{\beta_i} &= \frac{\sigma_2'}{\left(\left(g^{\alpha^1}\right)^{\beta_i}\right)^{m_{21}'} \cdot \left(\left(g^{\alpha^2}\right)^{\beta_i}\right)^{m_{22}'} \cdots \left(\left(g^{\alpha^s}\right)^{\beta_i}\right)^{m_{2s}'}};\\
&\cdots\cdots\\
H_2(n)^{\beta_i} &= \frac{\sigma_n'}{\left(\left(g^{\alpha^1}\right)^{\beta_i}\right)^{m_{n1}'} \cdot \left(\left(g^{\alpha^2}\right)^{\beta_i}\right)^{m_{n2}'} \cdots \left(\left(g^{\alpha^s}\right)^{\beta_i}\right)^{m_{ns}'}}
\end{aligned} \tag{20}$$

(4) The adversary (malicious cloud server) now modifies data blocks

$$m_{11}', m_{12}', \cdots, m_{1s}'; m_{21}', m_{22}', \cdots, m_{2s}'; \cdots\cdots; m_{n1}', m_{n2}', \cdots, m_{ns}' \tag{21}$$

to be any other data blocks

$$\bar{m}'_{11}, \bar{m}'_{12}, \cdots, \bar{m}'_{1s}; \bar{m}'_{21}, \bar{m}'_{22}, \cdots, \bar{m}'_{2s}; \cdots\cdots; \bar{m}'_{n1}, \bar{m}'_{n2}, \cdots, \bar{m}'_{ns}, \tag{22}$$

and it can also compute their authentication label as shown in the next step.

(5) For the adversary knows,

$$H_2(1)^{\beta_i}, H_2(2)^{\beta_i}, \cdots, H_2(n)^{\beta_i}, \tag{23}$$

and it also knows

$$\left(g^{\beta_i}, g^{\alpha^1\beta_i}, g^{\alpha^2\beta_i}, \cdots, g^{\alpha^j\beta_i}\right); \tag{24}$$

thus, it can compute the forged authentication label for modified data blocks as following:

$$\begin{aligned}
\bar{\sigma}_1{}' &= H_2(1)^{\beta_i} \cdot \left(\left(g^{\alpha^1}\right)^{\beta_i}\right)^{\bar{m}_{11}{}'} \cdot \left(\left(g^{\alpha^2}\right)^{\beta_i}\right)^{\bar{m}_{12}{}'} \cdots \left(\left(g^{\alpha^s}\right)^{\beta_i}\right)^{\bar{m}_{1s}{}'};\\
\bar{\sigma}_2{}' &= H_2(2)^{\beta_i} \cdot \left(\left(g^{\alpha^1}\right)^{\beta_i}\right)^{\bar{m}_{21}{}'} \cdot \left(\left(g^{\alpha^2}\right)^{\beta_i}\right)^{\bar{m}_{22}{}'} \cdots \left(\left(g^{\alpha^s}\right)^{\beta_i}\right)^{\bar{m}_{2s}{}'};\\
&\cdots\cdots\\
\bar{\sigma}_n{}' &= H_2(n)^{\beta_i} \cdot \left(\left(g^{\alpha^1}\right)^{\beta_i}\right)^{\bar{m}_{n1}{}'} \cdot \left(\left(g^{\alpha^2}\right)^{\beta_i}\right)^{\bar{m}_{n2}{}'} \cdots \left(\left(g^{\alpha^s}\right)^{\beta_i}\right)^{\bar{m}_{ns}{}'}
\end{aligned} \tag{25}$$

(6) For $\bar{\sigma}_1{}'$, $\bar{\sigma}_2{}'$, $\cdots, \bar{\sigma}_n{}'$ that are correct authentication labels, thus the adversary only need to follow the protocol's specification in the challenge-response auditing protocol by using these modified data blocks and forged authentication labels. And the forged proof and aggregated data blocks can pass through the verification equation.

6.2. Attack II. Our attack II is based on the following observation: even if the cloud server does not store any data blocks, it can forge proofs directly which can pass through the verification equation. Concretely, the attack is the following:

(1) In the cloud storage auditing protocol, the steps before proof generation is running as normal. The adversary (malicious cloud) does nothing in these steps except finally, it deletes all the stored blocks and their corresponding authentication labels

(2) 2. Proof generation: after the challenge information CM is received, according to $\{ID_{\text{group}} \| u_i \| \Delta t\}$, the malicious cloud first calculates the output port TPM. The authorization message $\{ID_{\text{group}} \| u_i \| \Delta t\}$ then is verified by the malicious cloud. The proof of possessing shared data is generated by the malicious cloud as follows:

(a) Subsets $L_1, \cdots, L_d$ are divided from the index set L, where the selected blocks that are signed by TPM_i are L_i

(b) Let u_{ij} randomly selected from Z_p^* and

$$\pi_i = \prod_{l \in L_i} e(\bar{\sigma}_l, R) = e\left(\prod_{l \in L_i} \left(H_2(l) \prod_{j=1}^{s} g^{\alpha^j u_{ij}}\right)^{\beta_i}, g^r\right) \tag{26}$$

that are calculated by the cloud server for each subset L_i; here, $1 \le i \le d$ and $1 \le j \le s$. Here, $\bar{\sigma}_l$ refers to the forged authentication label, and

$$\prod_{l \in L_i} \left(H_2(l) \prod_{j=1}^{s} g^{\alpha^j u_{ij}}\right)^{\beta_i} \tag{27}$$

can be calculated by the malicious cloud server as the above attack I.

(c) $w_i = \prod_{j=1}^{s} X^{\alpha^j u_{ij}}$ and $\pi = \prod_{i=1}^{d} \pi_i$ are calculated by the cloud servers, and then prf is the following

$$Prf = \left\{\{w_i\}_{1 \le i \le d}, \pi\right\}, \tag{28}$$

and it is sent to TPM_i as the proof

(3) Proof check: based on the received prf and the challenge message CM, the correctness of the following equation is verified by TPM:

$$\prod_{i=1}^{d} e\left(\eta_i^o, pk_{\text{TPM}_i}^r\right) e\left(w_i, pk_{\text{TPM}_i}^r\right) = \pi^o, . \tag{29}$$

where $\eta_i = \prod_{l \in L_i} H_2(l), 1 \le i \le d$. That is, it checks

$$\left(\prod_{i=1}^{d} e\left(\eta_i^o w_i, pk_{\text{TPM}_i}\right)\right)^r = \pi^o. \tag{30}$$

TPM_i outputs True if the equation is true, otherwise, outputs False.

Here, we show why this attack can be successful:

$$\begin{aligned}
&\prod_{i=1}^{d} e\left(\eta_i^o, pk_{TPM_i}^r\right) e\left(w_i, pk_{TPM_i}^r\right)\\
&= \prod_{i=1}^{d} e\left(\prod_{i \in L_i} H_2(l)^o, g^{\beta_i r}\right) e\left(\prod_{j=1}^{s} (g^o)^{\alpha^j u_{ij}}, g^{\beta_i r}\right)
\end{aligned}$$

$$= \prod_{i=1}^{d} e\left(\prod_{i\in L_i} H_2(l), g\right)^{o\beta_i r} \cdot e\left(\prod_{j=1}^{s} (g)^{\alpha^j u_{ij}}, g\right)^{o\beta_i r}$$
$$= \prod_{i=1}^{d} e\left(\prod_{i\in L_i} \prod_{j=1}^{s} (g)^{\alpha^j u_{ij}}, g\right)^{o\beta_i r}$$
$$= \prod_{i=1}^{d} \pi_i^o = \left(\prod_{i=1}^{d} \pi_i\right)^o = \pi^o. \tag{31}$$

7. Our Improved Secure Auditing Protocol for Shared Data

In this subsection, we give an improved secure auditing protocol for shared data, which is the following:

(1) Key generation:

(a) A random $\beta_i \in Z_p^*$ is selected by the group manager as the TPM's private key. g^{β_i} is also calculated by him

(b) A random $k_1 \in Z_p^*$ is selected by the group manager and is sent to the group members and the cloud

(c) $\alpha^j \in Z_p^*$ is randomly selected by the group manager and g^{α^j} is also calculated, and the public key of TPM_i is computed as the following:

$$pk_{\mathrm{TPM}_i} = g^{\beta_i} \tag{32}$$

(d) The interconnection function f and function sequence f_i' are selected by the group manager, and it also sets the input sending and output sending window and then sends them to the cloud

(2) Data blind:

(a) The secret seed k_1 is used to calculate the blind factor $\alpha_i = \zeta_{k_1}(i, \mathrm{name})$ by the group members, and the blind data m_{ij}' is calculated as

$$m_{ij}' = m_{ij} + \alpha_i \tag{33}$$

(b) A request to upload the data m_{ij}' is sent by a group member to the group manager, and it also calculates $\mathrm{hash}(id_{i,j})$ (the hash value) and sends $id_{i,j}$, m_{ij}' to the group manager securely. A new event is then created by the group member

(c) For the new event, a transaction record $\mathrm{hash}(id_{i,j})$ will be used and will be broadcasted within the group. According to the same hash algorithm, the group manager verifies $\mathrm{hash}(id_{i,j})$ after receiving the request. If the verification is passed, it receives m_{ij}'.

(3) Authorize:

(a) According to the TPM management strategy, the output port TPM_i in the virtual TPM pool is calculated by the group manager corresponding to the input port u_i (the requesting group member).

(b) The authorization message for TPM_i is generated by the group manager as follows:

$$\left\{\left(ID_{\mathrm{group}} \| u_i \| \Delta t_i\right), \Delta t_i'\right\}, \tag{34}$$

where the group manager's identity is ID_{group}, denotes the time when the request is processed by the group manager as Δt_i, and denotes $\Delta t_i'$ as the time authorized for the TPM by the group manager.

(c) According to the authorization message, the group manager calculates the value H_1 as follows:

$$H_1 = H_1\left(\left(ID_{\mathrm{group}} \| u_i \| \Delta t_i\right), \Delta t_i'\right) \tag{35}$$

(d) The authorization message is then sent by the group manager to the cloud and sends α^j, β_i, m_{ij}' and H_1 to TPM_i

(4) Authentication label generation:

(a) Authentication label σ_i' of m_{ij}' is generated by TPM_i, after getting the blind data block m_{ij}', and is as the following:

$$\sigma_i' = \left(H_2(i) \cdot \prod_{j=1}^{s} \left(g^{\alpha^j}\right)^{m_{ij}'}\right)^{\beta_i}, \tag{36}$$

e.g., m_5''s authentication label is generated. Note the public key of TPM is not the previous $pk_{\mathrm{TPM}_i} = (g^{\beta_i}, g^{\alpha^1\beta_i}, g^{\alpha^2\beta_i}, \cdots, g^{\alpha^j\beta_i})$, not only α^j is secret but g^{α^j} is also secret in our scheme. In this case, malicious cloud servers cannot obtain the value of $\{H_2(1), H_2(2), \cdots, H_2(n)\}$ from the public key of TPM and a large number of

authentication labels $\{\sigma_1, \sigma_2, \cdots, \sigma_n\}$. When there is no way to get the value of g^{α^j} and $H_2(i)$,the adversary naturally cannot forge the authentication labels. Therefore, our solution can effectively resist attack I.

$$\sigma_5{}' = \left(H_2(5) \cdot \prod_{j=1}^{s} \left(g^{\alpha^j} \right)^{m_{ij}{}'} \right)^{\beta_i}. \quad (37)$$

Then, the data file $(m_{ij}{}', \sigma_i{}')$ and H_1 is sent by TPM_i to the cloud.

(b) After receiving the corresponding group manager's authorization message and the corresponding TPM_i's $(m_{ij}{}', \sigma_i{}')$, the output port TPM_i is first calculated by the cloud. If the message is just sent by TPM_i at $\Delta t_i{}'$, then $H_1((ID_{\text{group}} \| u_i \| \Delta t_i), \Delta t_i{}')$ is calculated by the cloud, comparing with the value $H_1((ID_{\text{group}} \| u_i \| \Delta t_i), \Delta t_i{}')$ from TPM_i. If they are the same, run the next algorithm, otherwise stop the execution

(5) Authentication label check: the correctness of label $\sigma_i{}'$ is verified by the cloud as follows:

$$e(\sigma_{i'}, g) = e\left(H_2(i) \cdot \prod_{j=1}^{s} \left(g^{\alpha^j} \right)^{m_{ij}{}'}, g^{\beta_i} \right). \quad (38)$$

$(m_{ij}{}', \sigma_i{}')$ is received and stored if the above equation is true, otherwise rejected.

(6) Data recovery:

Based on k_1, the cloud calculates $\alpha_i = \zeta_{k_1}(i, \text{name})$, and using the following equation that computes the real data m_{ij}:

$$m_{ij} = m_{ij}' - \alpha_i. \quad (39)$$

Note here what the cloud server stores is the original data and corresponding blinded data blocks' authentication labels, instead of storing the original data blocks and the corresponding authentication labels. In this way, the cloud server needs not to compute the original data blocks' authentication labels σ_i. Finally, the real data blocks $m_i = (m_{i1}, m_{i2}, \cdots, m_{\text{is}})$ and the corresponding blinded authenticator $\sigma_i{}'$ are stored by the cloud.

(7) Challenge. When a challenge is initiated by the group manager to the cloud, Δt is randomly selected by him as the authorization time to TPM_i, where u_i's sending window on the input side is Δt. An audit authorization command is sent by the group manager to the TPM_i through u_i at Δt, and $\{ID_{\text{group}} \| u_i \| \Delta t\}$ is sent by it as the audit authorization information to the cloud

The TPM_i implements the audit process after receiving the group manager's authorization command as the following:

(a) c blocks from all blocks of the shared data are randomly selected by TPM_i, and the indexes of the selected blocks are denoted as L

(b) Two random numbers $o, r \in Z_p^*$ are generated by TPM_i, and $X = g^o$ and $R = g^r$ are calculated

(c) $\{X^{\alpha^j}\}_{1 \le j \le s}$ is calculated by TPM_i

(d) The challenge information is outputted by TPM_i:

$$\text{CM} = \left\{ L, R, \left\{ X^{\alpha^j} \right\}_{1 \le j \le s} \right\}. \quad (40)$$

Then, CM is sent by TPM_i to the cloud.

(8) Proof generation: after the challenge information CM is received, according to $\{ID_{\text{group}} \| u_i \| \Delta t\}$, the cloud first calculates the output port TPM. The authorization message $\{ID_{\text{group}} \| u_i \| \Delta t\}$ then is verified by the cloud. The proof of possessing shared data is generated by the cloud as follows:

(a) Subsets $L_1, \cdots, L_d$ are divided from the index set L, where the selected blocks that are signed by TPM_i are L_i

(b) Based on k_1, the cloud calculates $\alpha_i = \zeta_{k_1}(i, \text{name})$ and also computes $m_{ij}{}' = m_{ij} + \alpha_i$; then, it also computes $u_{ij} = \sum_{l \in L_i} m_{lj}{}'$ and

$$\pi_i = \prod_{l \in L_i} e\left(\sigma_l{}', R \right) = e\left(\prod_{l \in L_i} H_2(l) \prod_{j=1}^{s} g^{\alpha^j \sum_{l \in L_i} m_{lj}{}'}, g \right)^{\beta_i r} \quad (41)$$

that are calculated by the cloud server for each subset L_i; here, $1 \le i \le d$ and $1 \le j \le s$. Different from the original scheme, we processed the data again in a blind way and use blinded data blocks with their corresponding authentication labels to generate the proof. This processing is equivalent to replacing the operation of restoring the real authentication labels with the operation of data blind. The recovering of real authentication labels requires a lot of multiplication and exponentiation operations, while the blind processing of data only needs an addition operation. Obviously, the addition operation is less

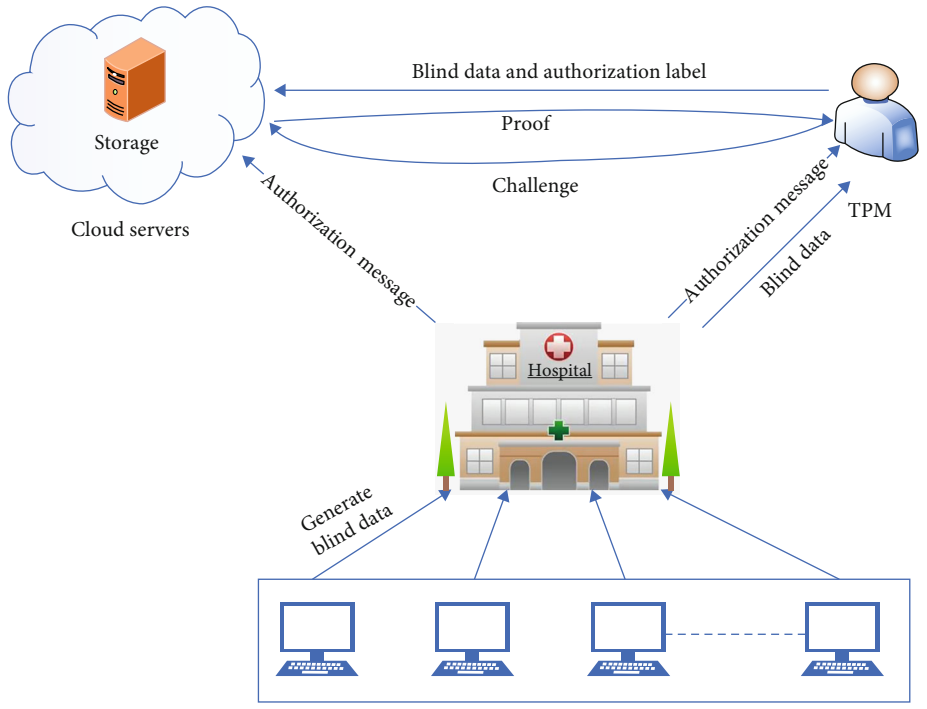

Figure 2: Cloud audit for medical information storage.

computationally expensive, so our scheme has a certain improvement in computing efficiency compared to the original one.

(c) $w_i = \prod_{j=1}^{s} X^{\alpha^j u_{ij}}$ and $\pi = \prod_{i=1}^{d} \pi_i$ are calculated by the cloud servers, and then prf is the following

$$Prf = \left\{\{w_i\}_{1\le i\le d}, \pi\right\}, \tag{42}$$

and it is sent to TPM_i as the proof.

(9) Proof check: based on the received prf and the challenge message CM, the correctness of the following equation is verified by TPM:

$$\prod_{i=1}^{d} e\left(\eta_i^o, pk_{\mathrm{TPM}_i}^r\right) e\left(w_i, pk_{\mathrm{TPM}_i}^r\right) = \pi^o, \tag{43}$$

where $\eta_i = \prod_{l\in L_i} H_2(l), 1 \le i \le d$. That is, it checks

$$\left(\prod_{i=1}^{d} e\left(\eta_i^o w_i, pk_{\mathrm{TPM}_i}\right)\right)^r = \pi^o. \tag{44}$$

TPM_i outputs True if the equation is true, otherwise, outputs False.

Here, we show the correctness of our protocol:

$$\begin{aligned}
&\prod_{i=1}^{d} e\left(\eta_i^o, pk_{\mathrm{TPM}_i}^r\right) e\left(w_i, pk_{\mathrm{TPM}_i}^r\right) \\
&= \prod_{i=1}^{d} e\left(\prod_{i\in L_i} H_2(l)^o, g^{\beta_i r}\right) e\left(\prod_{j=1}^{s} (g^o)^{\alpha^j u_{ij}}, g^{\beta_i r}\right) \\
&= \prod_{i=1}^{d} e\left(\prod_{i\in L_i} H_2(l), g\right)^{o\beta_i r} \cdot e\left(\prod_{j=1}^{s} (g)^{\alpha^j u_{ij}}, g\right)^{o\beta_i r} \\
&= \prod_{i=1}^{d} e\left(\prod_{i\in L_i}\prod_{j=1}^{s} (g)^{\alpha^j u_{ij}}, g\right)^{o\beta_i r} \\
&= \prod_{i=1}^{d} e\left(\prod_{i\in L_i}\prod_{j=1}^{s} (g)^{\alpha^j \sum_{l\in L_i} m_{lj}}, g\right)^{o\beta_i r} \\
&= \prod_{i=1}^{d} \pi_i^o = \left(\prod_{i=1}^{d} \pi_i\right)^o = \pi^o.
\end{aligned} \tag{45}$$

Through this verification, we can know that the cloud storage can still be implemented. At the same time, due to our reasonable setting of public key parameters of TPM, it is impossible for the adversary to forge authentication labels at will. Therefore, if the cloud server performs proof in a dishonest way, it will not be able to pass the verification of TPM, so our scheme can effectively resist the second attack ¢ò.

8. Application

Now, we apply this system model to real life, which can be seen in Figure 2. Take the hospital as an example patient information and other important information in the hospital are kept confidential. If there is too much patient information, the hospital needs to upload the data to the cloud platform and use the convenience of the cloud platform to solve the problem. To ensure the integrity of the data, the system model design is shown in the figure. First, doctors use the hospital's computer to blindly process medical data, integrate the blind data, and upload it to TPM. The TPM management strategy is designed by the hospital and the two exchanges of data and information after public and private key encryption. The TPM processes the blind data again, generates an authorization label, and then sends the blind data and the authorization label to the cloud server. Before this, the hospital must first send the authorization message to the TPM and cloud server to ensure that they can handle the blind data accordingly. The cloud server that receives the blind data then restores the data and stores the original data in the cloud database along with the authorization label. When you want to verify the integrity of the data, the hospital uses a third-party TPM to verify the integrity of the data. The challenge response is used to verify the authorization of the TPM, and if the verification is successful, the original data is sent to the TPM. In the whole process, the calculation cost of the hospital is reduced, and the integrity of the data can be verified. To a certain extent, it has been greatly improved compared with the original algorithm.

9. Conclusion

In this paper, We point out that Tian et al.'s lightweight secure auditing scheme for shared data in cloud storage is not secure. The authentication labels can be easily forged, and thus the cloud server can launch the following attacks: modify the data arbitrarily, delete the data arbitrarily, and add the data arbitrarily. In all these attacks, the cloud server still can give correct data possession proofs, which invalidates the security of cloud audit protocol. Then by optimizing the method of TPM's public key generation and the data integrity proof protocol framework, an improved secure cloud storage auditing protocol for shared data is given. Through comparative analysis, the article proves that, to a certain extent, our improved scheme is more of security and efficiency. Finally, the application scenarios of this paper in medical field are presented.

Data Availability

No data were used to support this study.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work is supported by the National Key Research and Development Program of China under Grant No. 2017YFB0802000, National Nature Science Foundation of China (Grant Nos. 61572521, U1636114), National Cryptography Development Fund of China under Grant No. MMJJ20170112, and an Open Project from Guizhou Provincial Key Laboratory of Public Big Data under Grant No. 2019BDKFJJ008. This work is also supported by the 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] J. Tian and X. Jing, "A lightweight secure auditing scheme for shared data in cloud storage," *IEEE Access*, vol. 7, pp. 68071–68082, 2019.

[2] A. Juels and B. S. Kaliski, "PoRs: proofs of retrievability for large files," in *Proceedings of ACM Conference on Computer and Communications Security*, pp. 584–597, Alexandria, Virginia, USA, 2007.

[3] G. Ateniese, R. C. Burns, R. Curtmola et al., "Provable data possesion at untrusted stores," pp. 598–609, 2007.

[4] H. Tian, Y. Chen, C. Chang et al., "Dynamic-hash-table based public auditing for secure cloud storage," *IEEE Transactions on Services Computing*, vol. 10, no. 5, pp. 701–714, 2017.

[5] H. Shacham and B. Waters, "Compact proofs or retrievability," pp. 90–107, 2008.

[6] J. Xu and E. C. Chang, "Towards efficient proofs of retrievability," pp. 79-80, 2012.

[7] D. Cash, A. Küpçü, and D. Wichs, "Dynamic proofs of retrievability via oblivious RAM," pp. 279–295, 2013.

[8] E. Shi, E. Stefanov, and C. Papamanthou, "Practical dynamic proofs of retrievability," pp. 325–336, 2013.

[9] J. Yuan and S. Yu, "Secure and constant cost public cloud storage auditing with deduplication," in *2013 IEEE Conference on Communications and Network Security (CNS)*,, pp. 145–153, National Harbor, MD, USA, 2013.

[10] 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.

[11] B. Wang, B. Li, and H. Li, "Public auditing for shared data with efficient user revocation in the cloud," *INFOCOM*, pp. 2904–2912, 2013.

[12] S. Halevi, D. Harnik, B. Pinkas, and A. Shulman-Peleg, "Proofs of ownership in remote storage systems," pp. 491–500, 2011.

[13] Y. Zhu, H. Hu, G. Ahn, and M. Yu, "Cooperative provable data possession for integrity verification in multicloud storage," *IEEE Transactions on Parallel and Distributed Systems*, vol. 23, no. 12, pp. 2231–2244, 2012.

[14] C. Wang, S. S. M. Chow, Q. Wang, K. Ren, and W. Lou, "Privacy-preserving public auditing for secure cloud storage," *IEEE Transactions on Computers*, vol. 62, no. 2, pp. 362–375, 2013.

[15] J. Zhang, B. Wang, D. He, and X. A. Wang, "Improved secure fuzzy auditing protocol for cloud data storage," *Soft Computing*, vol. 23, no. 10, pp. 3411–3422, 2019.

[16] Y. Wang, Q. Wu, B. Qin, X. Chen, X. Huang, and Y. Zhou, "Group-oriented proofs of storage," in *Proceedings of the 10th ACM Symposium on Information, Computer and Communications Security*, pp. 73–84, New York, NY, USA, 2015.

[17] X. A. Wang, Y. Liu, J. Zhang, X. Yang, and M. Zhang, "Improved group-oriented proofs of cloud storage in iot setting," *Concurrency and Computation: Practice and Experience*, vol. 30, no. 21, 2018.

[18] J. Wei, R. Zhang, J. Liu, J. Li, X. Niu, and Y. Yao, "Dynamic data integrity auditing for secure outsourcing in the cloud," *Concurrency and Computation: Practice and Experience*, vol. 29, no. 12, 2017.

[19] F. Sebé, J. Domingo-Ferrer, A. Martinez-Balleste, Y. Deswarte, and J. J. Quisquater, "Efficient remote data possession checking in critical information infrastructures," *IEEE Transactions on Knowledge and Data Engineering*, vol. 20, no. 8, pp. 1034–1038, 2008.

[20] A. Oprea, M. K. Reiter, and K. Yang, "Space-efficient block storage integrity," in *Proceedings of the Network and Distributed System Security Symposium (NDSS 05)*, 2005San Diego, California, USA.

[21] F. D. L. Gazzoni and P. S. L. M. Barreto, "Demonstrating data possession and uncheatable data transfer," *IACR Cryptology Print Archive*, p. 150, 2006.