Mathematical Biosciences and Engineering

http://www.aimspress.com/journal/mbe

MBE, 20(5): 7905–7921.
DOI: 10.3934/mbe.2023342
Received: 01 November 2022
Revised: 22 January 2023
Accepted: 06 February 2023
Published: 22 February 2023

Research article

Improved cloud storage auditing scheme with deduplication

Jindan Zhang[1,*], Urszula Ogiela[2], David Taniar[3] and Nadia Nedjah[4]

[1] Xianyang Vocational & Technical College, Xianyang, Shaanxi, China

[2] AGH University of Science and Technology, Institute of Computer Science, Krakow, Poland

[3] Faculty of Information Technology, Monash University, Australia

[4] Department of Electronics Engineering and Telecommunications of the Engineering Faculty, State University of Rio de Janeiro, Rio de Janeiro, Brazil

* **Correspondence:** Email: zhangjindan83@163.com.

Abstract: Cloud storage has become a crucial service for many users who deal with big data. The auditing scheme for cloud storage is a mechanism that checks the integrity of outsourced data. Cloud storage deduplication is a technique that helps cloud service providers save on storage costs by storing only one copy of a file when multiple users outsource the same file to cloud servers. However, combining storage auditing and deduplication techniques can be challenging. To address this challenge, in 2019 Hou et al. proposed a cloud storage auditing scheme with deduplication that supports different security levels of data popularity. This proposal is interesting and has practical applications. However, in this paper, we show that their proposal has a flaw: the cloud or other adversaries can easily forge the data block's authenticators, which means the cloud can delete all the outsourced encrypted data blocks but still provide correct storage proof for the third-party auditor. Based on Hou et al.'s scheme, we propose an improved cloud storage auditing scheme with deduplication and analyze its security. The results show that the proposed scheme is more secure.

Keywords: cloud storage; cloud audit; attack; deduplication; integrity

1. Introduction

1.1. Background

Nowadays, a large amount of data is generated every day. Securely storing and processing such a large amount of data is a significant challenge [1–3]. Cloud computing [4], artificial intelligence [5–8], and big data techniques [9] are promising ways to address this challenge. Among them, cloud storage is essential because it provides a basic way of storing such a vast amount of data. Cloud storage ser-

vices are becoming increasingly popular, and many people have outsourced their data, including files, movies, and photos, to the cloud. On the one hand, this service is very convenient for users since they do not need to maintain their data locally. Furthermore, they can access it via their mobile devices at any time and from anywhere. For some users, perhaps the most valuable aspect of cloud storage is the assurance that their outsourced data is almost never lost. On the other hand, the security of these outsourced files, movies, and photos cannot be guaranteed by the cloud storage service providers themselves. We need mechanisms to ensure their security, such as encryption, secure search, deduplication, and auditing techniques.

1.2. Motivation

Due to limited local storage, many data owners want to outsource their files to a cloud server, such as movies, pictures, or music. Before outsourcing the file to the cloud server, the data owners encrypt their files using convergent encryption. Then they outsource the encrypted files to the cloud server. If many data owners (more than the predefined threshold) outsource the same encrypted files to the cloud server, these encrypted files will be the same and denoted as "popular." They will then be deduplicated, and the cloud server will only store one copy for all the data owners. However, if only a few data owners (less than the predefined threshold) outsource the encrypted files to the cloud server, these encrypted files will be denoted as "unpopular," and they will not be deduplicated. In this way, the cloud server can save on storage costs. The technique of deduplication has been used by many cloud service providers, such as Amazon.

In this paper, we focus on a recently proposed scheme for cloud storage auditing [10] with deduplication. It supports different security levels and first introduces the concept of different security levels in this context. In this scheme, the outsourced data is categorized as popular or unpopular. If many data users have outsourced the same file to the cloud, this file can be considered popular. Otherwise, if the outsourced file has not been outsourced by many users, it can be categorized as unpopular. For popular files, Hou et al. suggest using convergent encryption to encrypt them, which is better for deduplication. In this way, cloud service providers can greatly reduce storage space. But for unpopular files, they suggest using probabilistic encryption to achieve semantic security, which is more secure than convergent encryption. Due to the unpopularity of these files, deduplication is no longer necessary. Generally speaking, this proposal is very interesting and valuable. However, we will show that it is not as secure as claimed, as there are some flaws in the scheme that invalidate its protocol's security. We also propose an improved scheme to achieve the security goal.

1.3. Our contribution

Our contribution can be summarized is two-fold. First, we focus on and demonstrate that the scheme proposed in [10] is not secure. Although it is the first relevant work on introducing data popularity to cloud auditing, this scheme is not entirely secure. We also analyze why their scheme has this security flaw and show how to avoid it. Then , we present an improved scheme building on the ideas proposed in [10], and provide a thorough analysis of its security. Our scheme addresses the security flaw present in the original proposal, and we explain in detail why it is more resistant to attacks.

1.4. Organization

In section 1, we provide the background, paper contribution, and organization. In section 2, we discuss related work. In section 3, we review the scheme proposed by Hou et al. In section 4, we present the attack. In section 5, we provide an improved proposal and briefly analyze its security. Finally, in section 6, we conclude the paper.

2. Related work

There is a large body of research in this context, and we have included the most relevant works to the one presented in this paper. This includes related work on encryption and deduplication techniques, as well as auditing schemes.

1. Encryption and deduplication are important techniques for ensuring the confidentiality and efficient management of outsourced data. While traditional encryption techniques, such as probabilistic public key encryption or symmetric encryption like AES, can achieve semantic security, they are not suitable for implementing functionality such as searching and deduplication. Therefore, novel encryption techniques have been developed specifically for use in cloud computing, including encryption with keyword search [11–14], encryption with access control [15, 16], convergent encryption with deduplication [17, 18], and others [19].
2. Auditing is an important way to ensure the integrity of the outsourced data. In 2007, Ateneo et al. [20] proposed the concept of provable data possession, which aims to allow the cloud servers to provide proof that they have stored the outsourced data well to the cloud users. Furthermore, the proof is very compact and the probability of cheating by the cloud servers is very low. This interesting primitive is actually a new auditing method for cloud storage. Since then, many cloud auditing schemes following this paradigm have been proposed, such as dynamic provable data possession [21], proof of retrievability [22, 23], compact proof of retrievability [24], publicly verifiable auditing [25, 26].
3. In 2016, Yu proposed a cloud data integrity checking scheme with an identity-based auditing mechanism from RSA [27]. Later, they proposed identity-based [28], attribute-based [29], and blockchain-based [30] cloud auditing schemes with different properties, and these are very interesting results in this field. Sometimes, the cloud service provider needs to use both auditing and deduplication techniques simultaneously. This way, the cloud service provider can reduce its costs when many users are outsourcing the same file, like popular movies, popular music, etc., and at the same time, the cloud users can ensure that their outsourced data, like files and photos, have not been lost or tampered with.

3. Description of Hou et al.'s scheme

The system model of Hou et al.'s cloud auditing scheme with deduplication [10] is shown in Figure 1. There are three roles in the system: the data owners, the cloud server, and the TPA (third-party auditor). The system operates as follows:

1. To check the integrity of the outsourced files, the data owners, cloud server, and TPA (third party auditor) run the auditing scheme proposed by Hou et al. In this scheme, the data owners compute

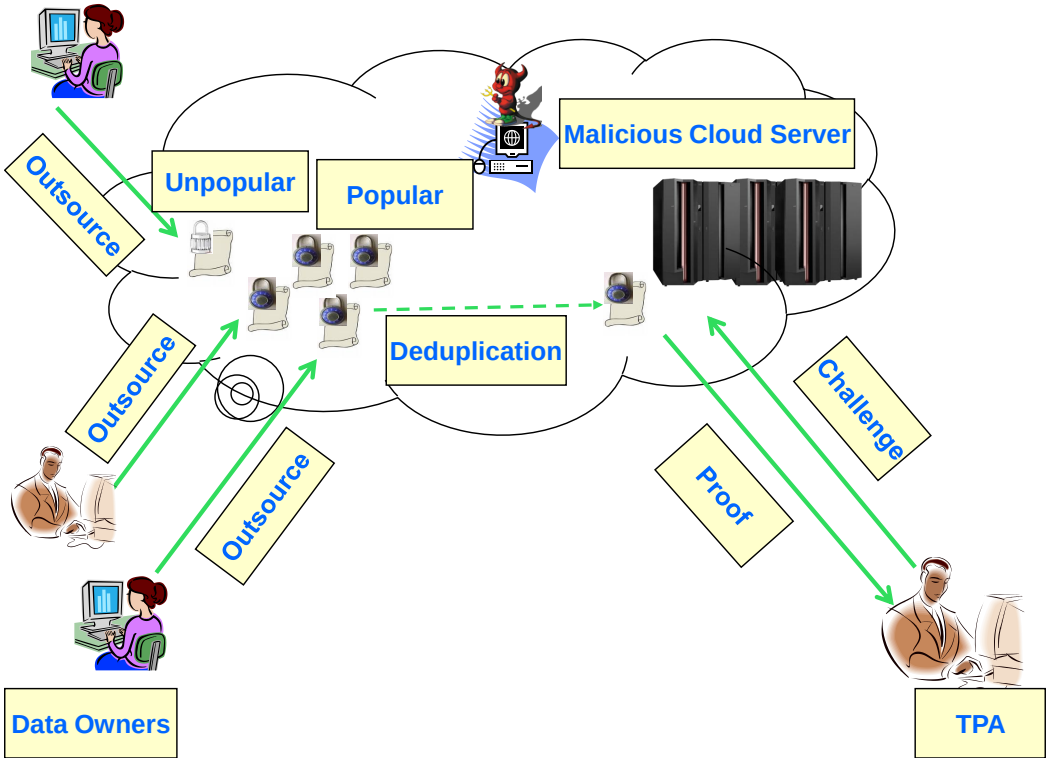

Figure 1. System model of Hou et al's cloud auditing scheme with deduplication.

the authenticators for the data blocks of the files and outsource both the files and the authenticators to the cloud server. When the data owners want to check the integrity of the outsourced files, they delegate this task to the TPA. The TPA launches a challenge-proof game with the cloud server. First, the TPA sends a challenge to the cloud server requesting the server to return the aggregated data blocks and the corresponding aggregated authenticators as proof for the integrity of the outsourced file. Then, the cloud server returns the proof to the TPA, who checks the correctness of the proof using verification equations.

2. However, in the auditing process described above, the cloud server may be malicious. In an effort to reduce storage space, it may delete or modify some outsourced files without being detected by the data owners or the TPA. This means that the malicious cloud server has a strong incentive to delete the outsourced files. In the following section, we will demonstrate an attack on Hou et al's auditing scheme. In this attack, the malicious cloud server is able to forge the authenticator for any data block, which in turn invalidates their auditing scheme.

We will now review Hou et al.'s scheme [10]. The core data flow between the data owner and cloud storage server, between the IS and cloud storage server, and between the TPA and cloud storage server can be seen in Figures 2, 3, and 4.

Notations: Assume the file outsourced to the cloud by the data owner is $F = \{m_1, m_2, \cdots, m_n\}$. Each $m_i = \{m_{i1}, m_{i2}, \cdots, m_{is}\}$, here $1 \leq i \leq n$. The file has its unique file identifier, it is signed with signature $SSig$ to prevent the attackers to modify it. The user (data owner) keeps his secret key for generating $SSig$ and publish the public key for signature.

1. Setup: With parameter k as the input,
   (a) Running $IG(1^k)$ to generate G_1 and G_2, which are two cyclic multiplicative groups of large prime order p. There exists a $e : G_1 \times G_1 \to G_2$ which is a bilinear pairing.
   (b) We denote $h : \{0,1\}^* \to \{0,1\}^*$ as an indexing function, $\phi : Z_p^* \times Z_p^* \to Z_p^*$ and $\pi : Z_p^* \times \{1,2,\cdots,n\}$ as a PRF and a PRP, denote $H_1 : \{0,1\}^* \to G_1$, $H_2 : G_2 \to \{0,1\}^l$, $H_3 : \{0,1\}^* \to G_1$ as three cryptographic hash functions.
   (c) Run $\varepsilon_\mu.Setup(k,n,t) \to (pk, sk, S)$ where $pk = \{p, G_1, G_2, e, H_1, H_2, H_3, h, g, g_{pub}\}$, n key shares $\{x_i\}_{i=0}^{n-1}$ also generated.
2. Join: this algorithm is not directly related with the attack.
3. Upload: this algorithm is not directly related with the attack.
4. AuthGen: With a ciphertext of file $C = \{c_1, c_2, \cdots, c_n\}$ (specially C is C_{ϵ_μ} or C_ϵ) and a secret key $k_{tag} \leftarrow Z_p^*$, the key $v \leftarrow g^{k_{tag}}$ is computed and published by the user. For each ciphertext block $c_i (1 \le i \le n)$, the authenticator T_i is generated by Ui and uploaded to the cloud.
   (a) $u_1, u_2, \cdots, u_s$ are s generators of G_1, which are chosen by Ui, $r \leftarrow Z_p^*$ is also randomly chosen by Ui.
   (b) Denote $\tau_0 = name\|n\|v^r\|u_1\|u_2\|\cdots\|u_s$. Let $ssk \leftarrow Z_p^*$ be signing key and $P_{ssk} \leftarrow g^{ssk}$ the corresponding verification key. These are randomly generated by the user. The file tag is $\tau \leftarrow \tau_0 \| SSig_{ssk}(\tau_0)$.
   (c) For each data block the authenticator is computed by Ui as
   $$T_i = \left(H_3(name\|i) \cdot \prod_{j=1}^{s} u_j^{c_{ij}} \right)^{k_{tag}}.$$
   (d)
   $$\left\{ u_1^{k_{tag}(r(C_{\epsilon_\mu})_{i,1} - (C_\epsilon)_{i,1})}, u_2^{k_{tag}(r(C_{\epsilon_\mu})_{i,2} - (C_\epsilon)_{i,2})}, \cdots, u_s^{k_{tag}(r(C_{\epsilon_\mu})_{i,s} - (C_\epsilon)_{i,s})} \right\}$$
   are computed by Ui and sent to IS.
   (e) The file tag and $\{T_i\}_{1 \le i \le n}$ are sent by Ui to the cloud.
5. PopularityChange: For the popularity threshold t, the algorithm is executed whenever the users' number that are submitting the same index is higer than it. The file F is not needed to upload to cloud again by the user Ui. IS sends the set $index$ to the cloud, and Ui sends it to the cloud. For all those users with file index in the set $index$, the storage cloud collects decryption shares of them. Then the ciphertext C_{ϵ_μ} uploaded by these users can be decrypted by the storage cloud. Then, the ciphertext F_C encrypted by the convergent encryption can be obtained by the storage cloud. Thus, as the ciphertext F_C coincides with that for file F, the deduplication can be achieved. Finally, $\left\{ u_1^{k_{tag}(r(C_{\epsilon_\mu})_{i,1} - (C_\epsilon)_{i,1})}, u_2^{k_{tag}(r(C_{\epsilon_\mu})_{i,2} - (C_\epsilon)_{i,2})}, \cdots, u_s^{k_{tag}(r(C_{\epsilon_\mu})_{i,s} - (C_\epsilon)_{i,s})} \right\}$ are sent to the cloud by the IS. The new data block authenticator*
   $$T_i' = T_i \cdot \prod_{j=1}^{s} u_j^{k_{tag}(rc_{\epsilon\mu i_j} - c_{\epsilon_{i_j}})}$$
   for each user are created by the clouds.

*In [10], $T_i' = T_i \cdot \prod_{j=1}^{s} u_j^{k_{tag}(rc_{\epsilon\mu i_j} - c_{\epsilon_{i_j}})}$, but we think it should be $T_i' = T_i \cdot \prod_{j=1}^{s} u_j^{k_{tag}(r(C_{\epsilon_\mu})_{i,j} - (C_\epsilon)_{i,j})}$.

6. ProofGen: With the $\{c_i\}_{1\le i\le n}$, $\{T_i\}_{1\le i\le n}$ as the input,
   - The auditing challenge is generated by TPA as the following:
     (a) The file tag gained by the TPA from the cloud and using the key $Pssk$ it checks whether the correctness of signature on τ_0. TPA rejects and halts if the signature is not correct.
     (b) Otherwise, filename $name$, n, v^r and $\{u_1, u_2, \cdots, u_s\}$ are recovered by the TPA. Then c, with $1 \le c \le n$ is chosen by him.
     (c) Parameters $k_1 \leftarrow Z_p^*$, $k_2 \leftarrow Z_p^*$ are randomly selected by the TPA.
     (d) The challenge $chal = (c, k_1, k_2)$ is sent by the TPA to the cloud.
   - The cloud yields $l_t = \pi_{k_1}(t)$ and $a_t = \phi_{k_2}(t)$ wherein $1 \le t \le c$ after receiving $chal$ from the TPA. And then the proof $T = \prod_{t=1}^{c} T_{l_t}^{a_t}$, $\eta_j = \sum_{t=1}^{c} a_t \cdot c_{l_t,j}$, $1 \le j \le s$ is computed.
7. ProofVerify: With the proof $P = (T, \eta)$ and the challenge massage $chal = (c, k_1, k_2)$, TPA computes $l_t = \pi_{k_1}(t)$ together with $a_t = \phi_{k_2}(t)$ wherein $1 \le t \le c$. Subsequently, the below verification equations are checked

$$e(T, g) = e\left(\prod_{t=1}^{c}\left(H_3\left(name\|l_t\right)^{a_t}\prod_{j=1}^{s}u_j^{\eta_j}\right), v\right),$$

$$e(T, g) = e\left(\prod_{t=1}^{c}\left(H_3\left(name\|l_t\right)^{a_t}\prod_{j=1}^{s}u_j^{\eta_j}\right), v^r\right).$$

   If one of them passed, the proof is valid.

4. Attack on the generation and updating of authenticators

The attack is executed according to the following steps:

1. The attacker can be the IS or the cloud. Note here the IS or the cloud can obtain

$$\left\{u_1^{k_{tag}(r(C_{\epsilon_\mu})_{i,1}-(C_\epsilon)_{i,1})}, u_2^{k_{tag}(r(C_{\epsilon_\mu})_{i,2}-(C_\epsilon)_{i,2})}, \cdots, u_s^{k_{tag}(r(C_{\epsilon_\mu})_{i,s}-(C_\epsilon)_{i,s})}\right\}, 1 \le i \le n$$

   from Ui by running algorithm AuthGen. Concretely the IS or the cloud can obtain

$$\begin{aligned}
&\left\{u_1^{k_{tag}(r(C_{\epsilon_\mu})_{1,1}-(C_\epsilon)_{1,1})}, u_2^{k_{tag}(r(C_{\epsilon_\mu})_{1,2}-(C_\epsilon)_{1,2})}, \cdots, u_s^{k_{tag}(r(C_{\epsilon_\mu})_{1,s}-(C_\epsilon)_{1,s})}\right\},\\
&\left\{u_1^{k_{tag}(r(C_{\epsilon_\mu})_{2,1}-(C_\epsilon)_{2,1})}, u_2^{k_{tag}(r(C_{\epsilon_\mu})_{2,2}-(C_\epsilon)_{2,2})}, \cdots, u_s^{k_{tag}(r(C_{\epsilon_\mu})_{2,s}-(C_\epsilon)_{2,s})}\right\},\\
&\left\{u_1^{k_{tag}(r(C_{\epsilon_\mu})_{3,1}-(C_\epsilon)_{3,1})}, u_2^{k_{tag}(r(C_{\epsilon_\mu})_{3,2}-(C_\epsilon)_{3,2})}, \cdots, u_s^{k_{tag}(r(C_{\epsilon_\mu})_{3,s}-(C_\epsilon)_{3,s})}\right\},\\
&\cdots\cdots\cdots,\\
&\left\{u_1^{k_{tag}(r(C_{\epsilon_\mu})_{n,1}-(C_\epsilon)_{n,1})}, u_2^{k_{tag}(r(C_{\epsilon_\mu})_{n,2}-(C_\epsilon)_{n,2})}, \cdots, u_s^{k_{tag}(r(C_{\epsilon_\mu})_{n,s}-(C_\epsilon)_{n,s})}\right\}.
\end{aligned}$$

2. Let $A_1 = u_1^{rk_{tag}}$, $B_1 = u_1^{k_{tag}}$, $A_2 = u_2^{rk_{tag}}$, $B_2 = u_2^{k_{tag}}$, $\cdots, \cdots, \cdots, A_s = u_s^{rk_{tag}}$, $B_s = u_s^{k_{tag}}$ then

$$\left\{u_1^{k_{tag}(r(C_{\epsilon_\mu})_{1,1}-(C_\epsilon)_{1,1})}, u_2^{k_{tag}(r(C_{\epsilon_\mu})_{1,2}-(C_\epsilon)_{1,2})}, \cdots, u_s^{k_{tag}(r(C_{\epsilon_\mu})_{1,s}-(C_\epsilon)_{1,s})}\right\},$$

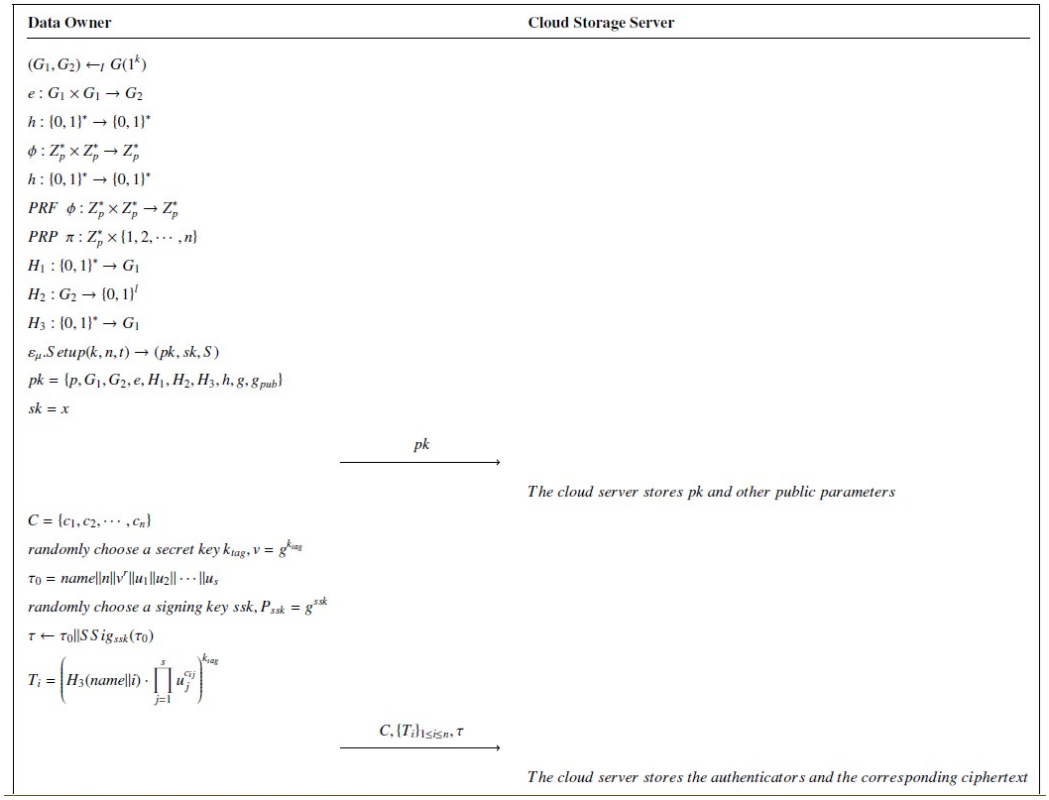

Figure 2. The core data flow between data owner and cloud storage server.

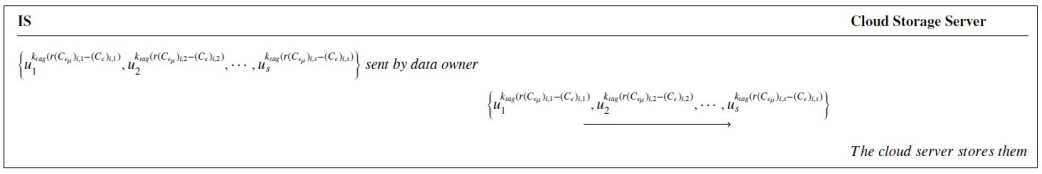

Figure 3. The core data flow between IS and cloud storage server.

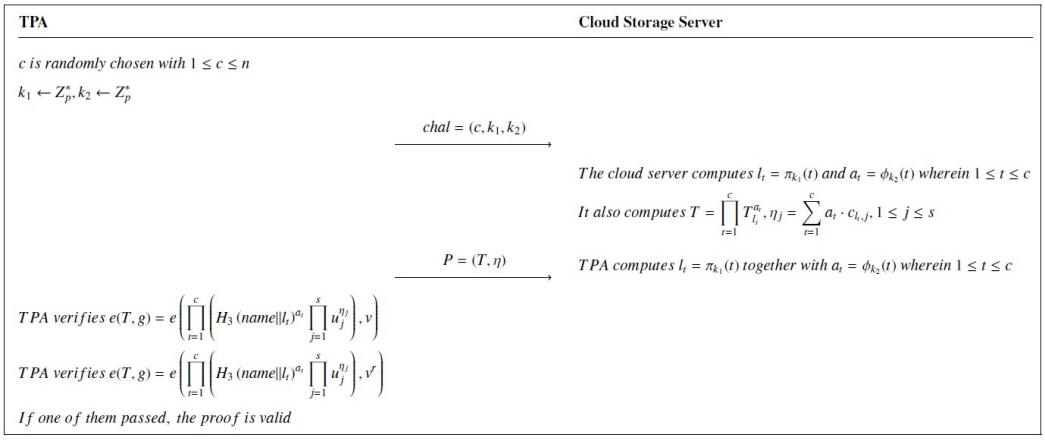

Figure 4. The core data flow between TPA and cloud storage server.

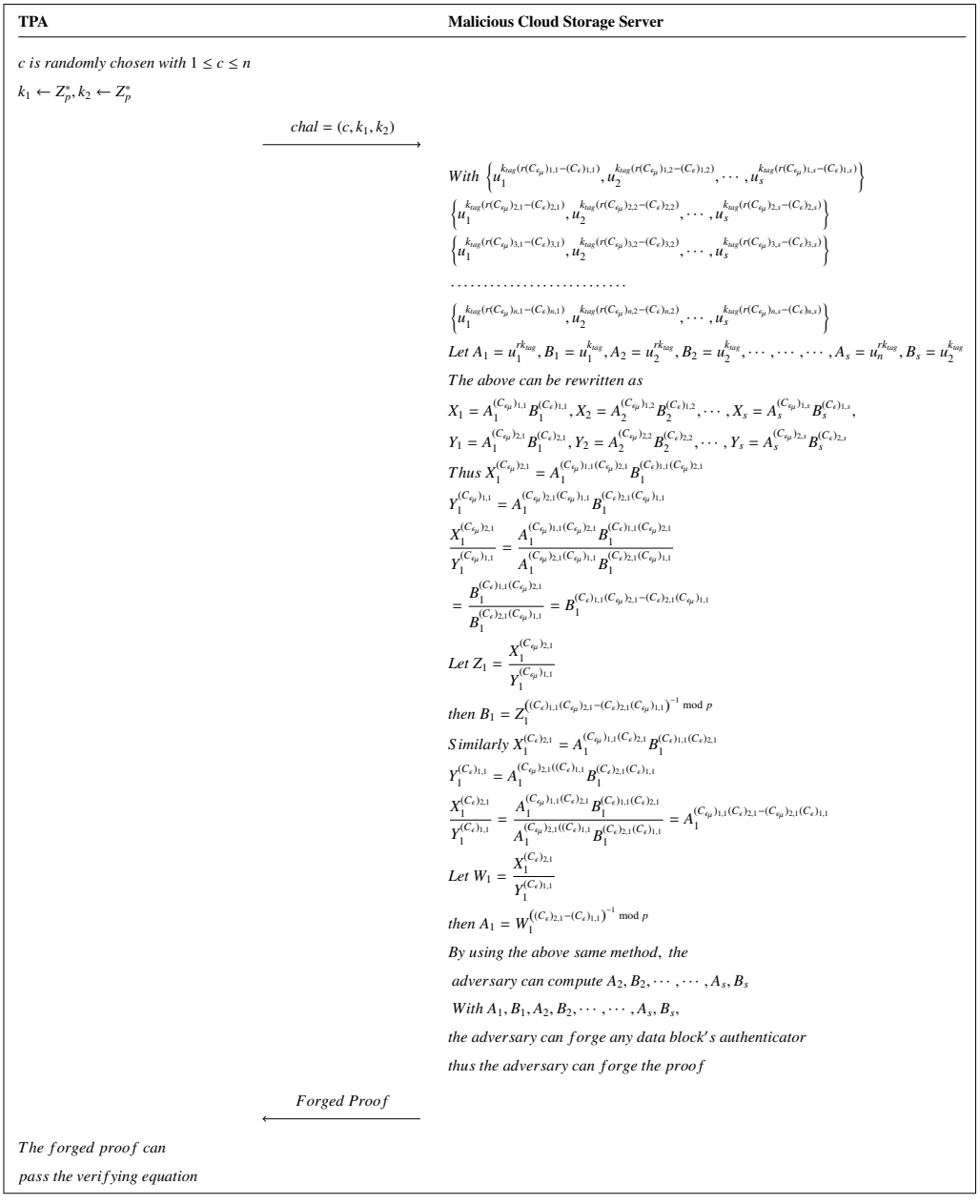

Figure 5. The attack.

$$\left\{u_1^{k_{tag}(r(C_{\epsilon\mu})_{2,1}-(C_\epsilon)_{2,1})}, u_2^{k_{tag}(r(C_{\epsilon\mu})_{2,2}-(C_\epsilon)_{2,2})}, \cdots, u_s^{k_{tag}(r(C_{\epsilon\mu})_{2,s}-(C_\epsilon)_{2,s})}\right\},$$
$$\left\{u_1^{k_{tag}(r(C_{\epsilon\mu})_{3,1}-(C_\epsilon)_{3,1})}, u_2^{k_{tag}(r(C_{\epsilon\mu})_{3,2}-(C_\epsilon)_{3,2})}, \cdots, u_s^{k_{tag}(r(C_{\epsilon\mu})_{3,s}-(C_\epsilon)_{3,s})}\right\},$$
$$\cdots\cdots\cdots,$$
$$\left\{u_1^{k_{tag}(r(C_{\epsilon\mu})_{n,1}-(C_\epsilon)_{n,1})}, u_2^{k_{tag}(r(C_{\epsilon\mu})_{n,2}-(C_\epsilon)_{n,2})}, \cdots, u_s^{k_{tag}(r(C_{\epsilon\mu})_{n,s}-(C_\epsilon)_{n,s})}\right\}$$

can be rewritten as

$$\left\{A_1^{(C_{\epsilon\mu})_{1,1}} B_1^{(C_\epsilon)_{1,1}}, A_2^{(C_{\epsilon\mu})_{1,2}} B_2^{(C_\epsilon)_{1,2}}, \cdots, A_s^{(C_{\epsilon\mu})_{1,s}} B_s^{(C_\epsilon)_{1,s}}\right\},$$
$$\left\{A_1^{(C_{\epsilon\mu})_{2,1}} B_1^{(C_\epsilon)_{2,1}}, A_2^{(C_{\epsilon\mu})_{2,2}} B_2^{(C_\epsilon)_{2,2}}, \cdots, A_s^{(C_{\epsilon\mu})_{2,s}} B_s^{(C_\epsilon)_{2,s}}\right\},$$
$$\left\{A_1^{(C_{\epsilon\mu})_{3,1}} B_1^{(C_\epsilon)_{3,1}}, A_2^{(C_{\epsilon\mu})_{3,2}} B_2^{(C_\epsilon)_{3,2}}, \cdots, A_s^{(C_{\epsilon\mu})_{3,s}} B_s^{(C_\epsilon)_{3,s}}\right\},$$
$$\cdots\cdots\cdots,$$
$$\left\{A_1^{(C_{\epsilon\mu})_{n,1}} B_1^{(C_\epsilon)_{n,1}}, A_2^{(C_{\epsilon\mu})_{n,2}} B_2^{(C_\epsilon)_{n,2}}, \cdots, A_s^{(C_{\epsilon\mu})_{n,s}} B_s^{(C_\epsilon)_{n,s}}\right\}.$$

3. With

$$\left\{A_1^{(C_{\epsilon\mu})_{1,1}} B_1^{(C_\epsilon)_{1,1}}, A_2^{(C_{\epsilon\mu})_{1,2}} B_2^{(C_\epsilon)_{1,2}}, \cdots, A_s^{(C_{\epsilon\mu})_{1,s}} B_s^{(C_\epsilon)_{1,s}}\right\},$$
$$\left\{A_1^{(C_{\epsilon\mu})_{2,1}} B_1^{(C_\epsilon)_{2,1}}, A_2^{(C_{\epsilon\mu})_{2,2}} B_2^{(C_\epsilon)_{2,2}}, \cdots, A_s^{(C_{\epsilon\mu})_{2,s}} B_s^{(C_\epsilon)_{2,s}}\right\},$$

the attacker can compute A_1, B_1, $\cdots, \cdots, \cdots$, A_s, B_s as following. First let $X_1 = A_1^{(C_{\epsilon\mu})_{1,1}} B_1^{(C_\epsilon)_{1,1}}, X_2 = A_2^{(C_{\epsilon\mu})_{1,2}} B_2^{(C_\epsilon)_{1,2}}, \cdots, X_s = A_s^{(C_{\epsilon\mu})_{1,s}} B_s^{(C_\epsilon)_{1,s}}$, $Y_1 = A_1^{(C_{\epsilon\mu})_{2,1}} B_1^{(C_\epsilon)_{2,1}}, Y_2 = A_2^{(C_{\epsilon\mu})_{2,2}} B_2^{(C_\epsilon)_{2,2}}, \cdots, Y_s = A_s^{(C_{\epsilon\mu})_{2,s}} B_s^{(C_\epsilon)_{2,s}}$ then the above can be rewritten as

$$\{X_1, X_2, \cdots, X_s\}, \{Y_1, Y_2, \cdots, Y_s\}.$$

4. With X_1, Y_1, the adversary can compute A_1, B_1 as following:

$$X_1^{(C_{\epsilon\mu})_{2,1}} = A_1^{(C_{\epsilon\mu})_{1,1}(C_{\epsilon\mu})_{2,1}} B_1^{(C_\epsilon)_{1,1}(C_{\epsilon\mu})_{2,1}},$$
$$Y_1^{(C_{\epsilon\mu})_{1,1}} = A_1^{(C_{\epsilon\mu})_{2,1}(C_{\epsilon\mu})_{1,1}} B_1^{(C_\epsilon)_{2,1}(C_{\epsilon\mu})_{1,1}},$$

then

$$\begin{aligned}
&\frac{X_1^{(C_{\epsilon\mu})_{2,1}}}{Y_1^{(C_{\epsilon\mu})_{1,1}}} \\
&= \frac{A_1^{(C_{\epsilon\mu})_{1,1}(C_{\epsilon\mu})_{2,1}} B_1^{(C_\epsilon)_{1,1}(C_{\epsilon\mu})_{2,1}}}{A_1^{(C_{\epsilon\mu})_{2,1}(C_{\epsilon\mu})_{1,1}} B_1^{(C_\epsilon)_{2,1}(C_{\epsilon\mu})_{1,1}}} \\
&= \frac{B_1^{(C_\epsilon)_{1,1}(C_{\epsilon\mu})_{2,1}}}{B_1^{(C_\epsilon)_{2,1}(C_{\epsilon\mu})_{1,1}}} \\
&= B_1^{(C_\epsilon)_{1,1}(C_{\epsilon\mu})_{2,1}-(C_\epsilon)_{2,1}(C_{\epsilon\mu})_{1,1}}.
\end{aligned}$$

5. Due to the group order p is publicly known and thus the following holds. Let $Z_1 = \frac{X_1^{(C_{\epsilon\mu})_{2,1}}}{Y_1^{(C_{\epsilon\mu})_{1,1}}}$ then $B_1 = Z_1^{\left((C_\epsilon)_{1,1}(C_{\epsilon\mu})_{2,1}-(C_\epsilon)_{2,1}(C_{\epsilon\mu})_{1,1}\right)^{-1} \bmod p}$.
6. Similarly

$$\begin{aligned} X_1^{(C_\epsilon)_{2,1}} &= A_1^{(C_{\epsilon\mu})_{1,1}(C_\epsilon)_{2,1}} B_1^{(C_\epsilon)_{1,1}(C_\epsilon)_{2,1}}, \\ Y_1^{(C_\epsilon)_{1,1}} &= A_1^{(C_{\epsilon\mu})_{2,1}((C_\epsilon)_{1,1}} B_1^{(C_\epsilon)_{2,1}(C_\epsilon)_{1,1}}, \end{aligned}$$

then

$$\begin{aligned} &\frac{X_1^{(C_\epsilon)_{2,1}}}{Y_1^{(C_\epsilon)_{1,1}}} \\ &= \frac{A_1^{(C_{\epsilon\mu})_{1,1}(C_\epsilon)_{2,1}} B_1^{(C_\epsilon)_{1,1}(C_\epsilon)_{2,1}}}{A_1^{(C_{\epsilon\mu})_{2,1}((C_\epsilon)_{1,1}} B_1^{(C_\epsilon)_{2,1}(C_\epsilon)_{1,1}}} \\ &= A_1^{(C_{\epsilon\mu})_{1,1}(C_\epsilon)_{2,1}-(C_{\epsilon\mu})_{2,1}(C_\epsilon)_{1,1}}. \end{aligned}$$

7. Due to the group order p is publicly known and thus the following holds. Let

$$W_1 = \frac{X_1^{(C_\epsilon)_{2,1}}}{Y_1^{(C_\epsilon)_{1,1}}},$$

then

$$A_1 = W_1^{\left((C_\epsilon)_{2,1}-(C_\epsilon)_{1,1}\right)^{-1} \bmod p}.$$

8. By using the above same method, the adversary can compute $A_2, B_2, \cdots, \cdots, A_s, B_s$.

With $A_1, B_1, A_2, B_2, \cdots, \cdots, A_s, B_s$, the adversary can forge any data block's authenticator as the following.

1. First the adversary (the IS or the cloud) can obtain

$$T_i = (H_3(name||i) \cdot \prod_{j=1}^{s} u_j^{c_{ij}})^{k_{tag}} (1 \le i \le n).$$

Note here c_{ij} is public known to all.
2. With

$$T_i = (H_3(name||i) \cdot \prod_{j=1}^{s} u_j^{c_{ij}})^{k_{tag}},$$

and $A_1, B_1, A_2, B_2, \cdots, \cdots, A_s, B_s$, the adversary can compute

$$\frac{T_i}{B_1^{c_{i1}} B_2^{c_{i2}} B_3^{c_{i2}} \cdots B_n^{c_{i2}}}$$

$$= \frac{(H_3(name||i) \cdot \prod_{j=1}^{s} u_j^{c_{ij}})^{k_{tag}}}{B_1^{c_{i1}} B_2^{c_{i2}} B_3^{c_{i2}} \cdots B_n^{c_{i2}}}$$
$$= \frac{(H_3(name||i) \cdot \prod_{j=1}^{s} u_j^{c_{ij}})^{k_{tag}}}{(u_1^{k_{tag}})^{c_{i1}} (u_2^{k_{tag}})^{c_{i2}} (u_3^{k_{tag}})^{c_{i2}} \cdots (u_n^{k_{tag}})^{c_{i2}}}$$
$$= (H_3(name||i))^{k_{tag}}.$$

Then it forges any data block's authenticator as following

3. Let $\widehat{c_{ij}}$ be the forged encrypted j-th sector of the i-th data block, then the adversary compute the following:

$$(H_3(name||i))^{k_{tag}} \cdot (B_1^{\widehat{c_{i1}}} B_2^{\widehat{c_{i2}}} B_3^{\widehat{c_{i2}}} \cdots B_n^{\widehat{c_{i2}}})$$
$$= (H_3(name||i) \cdot \prod_{j=1}^{s} u_j^{\widehat{c_{ij}}})^{k_{tag}},$$

which is a valid authenticator for the any forged encrypted sector.

4. This means that the cloud can modify the outsouced encrypted data block and its corresponding authenticator to be any other one, which obviously breaks the security of cloud auditing protocols.

5. The improved cloud auditing scheme with secure generation and updating of the authenticators

First, we will review the core idea for updating the authenticator in [10]. Next, we will analyze why this core idea is not secure. Finally, we will present an improved method.

- *Now we review the core idea in [10]. In the original proposal,* $(H(i) \cdot u^{C_i})^x$ *is the authenticator. Assume* $\sigma_i = (H(i) \cdot u^{C_i})^x$ *is the original authenticator and* $\sigma_i = (H(i) \cdot u^{C'_i})^x$ *is the new corresponding authenticator. Denote* $\triangle\ i = \sigma'_i/\sigma_i = \frac{(H(i) \cdot u^{C'_i})^x}{(H(i) \cdot u^{C_i})^x} = (u^{(C'_i - C_i)})^x = (u^x)^{(C'_i - C_i)}$. *Thus,* $\sigma_i \cdot (u^x)^{(C'_i - C_i)} = \sigma'_i$. *The cloud can compute* σ'_i *given* σ_i *and* $(u^x)^{(C'_i - C_i)}$. *However, the inverse of* $(C'_i - C_i)$ *can be calculated by the adversary and thus it is not secure. For example, for* C^*_i, *the corresponding authenticator* $\sigma^*_i = \sigma_i \cdot (u^x)^{(C'_i - C_i)(C'_i - C_i)^{-1} \cdot (C^*_i - C_i)} = \sigma_i (u^x)^{(C^*_i - C_i)}$ *can be forged. For safety, a blind factor r is introduced.* $(u^x)^{(r \cdot C'_i - C_i)}$ *is first computed and then uploaded by the user to the storage cloud. New authenticator* $\sigma'_i = \sigma_i \cdot (u^x)^{(r \cdot C'_i - C_i)}$ *whenever there are changes regarding the data popularity.*
- *However, the attack above shows that their idea of using* $(u^x)^{(r \cdot C'_i - C_i)}$ *instead of* $(u^x)^{(C'_i - C_i)}$ *is still not secure. The reason is the following: if the cloud knows* $(u^x)^{(r \cdot C'_i - C_i)}$ *for many such* $1 \le i \le n$, *it can compute* $(u^x)^r$ *and* (u^x) *easily. And thus it can forge any authenticator updates* $(u^x)^{(r \cdot C'_{any} - C_{any})}$ *easily. Furthermore, it also can forge authenticator* $\sigma_i = (H(i) \cdot u^{C_{any}})^x$ *easily for any block* C_{any}, *thus their core idea is not secure.*
- *We improve their core idea by modifying* $(u^x)^{(r \cdot C'_i - C_i)}$ *to be* $((u^x)^{(r_i \cdot C'_i - C_i)}, u^{r_i})$ *for many such* $1 \le i \le n$, *in this way, the cloud can not compute* $(u^x)^{r_i} (1 \le i \le n)$ *and* (u^x) *easily. And thus it can not forge authenticators any more.*

Building upon the improved core idea, we have developed an improved cloud auditing scheme which is outlined below:

1. Setup: For the sake of comparison, it is worth noting that this algorithm is identical to the corresponding algorithm presented in [10].
2. Join: This algorithm is the same as the corresponding algorithm in [10].
3. Upload: This algorithm is the same as the corresponding algorithm in [10].
4. AuthGen: With a ciphertext of file $C = \{c_1, c_2, \cdots, c_n\}$ (specially C is C_{ϵ_μ} or C_ϵ) and a secret key $k_{tag} \leftarrow Z_p^*$, the key $v \leftarrow g^{k_{tag}}$ is computed and published by the user. For each ciphertext block $c_i (1 \le i \le n)$, the authenticator T_i is generated by Ui and uploaded to the cloud.
   (a) $u_1, u_2, \cdots, u_s$ are s generators of G_1, which are chosen by Ui, $r_1, r_2, \cdots, r_n \leftarrow Z_p^*$ are also randomly chosen by Ui.
   (b) Let $\tau_0 = name \| n \| v^{r_1} \| v^{r_2} \| \cdots \| v^{r_n} \| u_1 \| u_2 \| \cdots \| u_s$. A signing key $ssk \leftarrow Z_p^*$ and the corresponding verification key $P_{ssk} \leftarrow g^{ssk}$ are randomly generated by the user. The file tag is $\tau \leftarrow \tau_0 \| SSig_{ssk}(\tau_0)$.
   (c) For each data block the authenticator is computed by Ui as

$$T_i = \left(H_3 (name \| i) \cdot \prod_{j=1}^{s} u_j^{c_{ij}} \right)^{k_{tag}}.$$

   (d)

$$\left\{ u_1^{k_{tag}(r_1 (C_{\epsilon_\mu})_{1,1} - (C_\epsilon)_{1,1})}, u_2^{k_{tag}(r_1 (C_{\epsilon_\mu})_{1,2} - (C_\epsilon)_{1,2})}, \cdots, u_s^{k_{tag}(r_1 (C_{\epsilon_\mu})_{1,s} - (C_\epsilon)_{1,s})} \right\},$$

$$\left\{ u_1^{k_{tag}(r_2 (C_{\epsilon_\mu})_{2,1} - (C_\epsilon)_{2,1})}, u_2^{k_{tag}(r_2 (C_{\epsilon_\mu})_{2,2} - (C_\epsilon)_{2,2})}, \cdots, u_s^{k_{tag}(r_2 (C_{\epsilon_\mu})_{2,s} - (C_\epsilon)_{2,s})} \right\},$$

$$\cdots\cdots\cdots\cdots\cdots,$$

$$\left\{ u_1^{k_{tag}(r_n (C_{\epsilon_\mu})_{n,1} - (C_\epsilon)_{n,1})}, u_2^{k_{tag}(r_n (C_{\epsilon_\mu})_{n,2} - (C_\epsilon)_{n,2})}, \cdots, u_s^{k_{tag}(r_n (C_{\epsilon_\mu})_{n,s} - (C_\epsilon)_{n,s})} \right\}$$

   are computed by Ui and sent to IS, we denote them as Upd.
   (e) The file tag and $\{T_i\}_{1 \le i \le n}$ are sent by Ui to the cloud.
5. PopulartityChange: This algorithm is the same as the corresponding algorithm in [10] except

$$T_i' = T_i \cdot \prod_{j=1}^{s} u_j^{k_{tag}(r_i (C_{\epsilon_\mu})_{i,j} - (C_\epsilon)_{i,j})}.$$

   Note here we use r_i instead of r in the exponentiation.
6. ProofGen: With the $\{c_i\}_{1 \le i \le n}$, $\{T_i\}_{1 \le i \le n}$ as the input,
   - the auditing challenge is generated by TPA as the following:
     (a) The file tag gained by the TPA from the cloud and using the key $Pssk$ it checks whether the correctness of signature on τ_0. TPA rejects and halts if the signature is not correct.
     (b) Otherwise, filename $name$, n, $v^{r_1}, v^{r_2}, \cdots, \cdots, v^{r_n}$ and $\{u_1, u_2, \cdots, u_s\}$ are recovered by the TPA. Then $c (1 \le c \le n)$ is chosen by him, which is the number of the challenged blocks.
     (c) $k_1 \leftarrow Z_p^*$, $k_1 \leftarrow Z_p^*$ are randomly picked by the TPA.

(d) The challenge $chal = (c, k_1, k_2)$ is sent by the TPA to the cloud.

- The cloud computes $l_t = \pi_{k_1}(t), a_t = \phi_{k_2}(t)(1 \le t \le c)$ after receiving *chal* from the TPA. And then the proof $T = \prod_{t=1}^{c} T_{l_t}^{a_t}$, $\eta_j = \sum_{t=1}^{c} a_t \cdot c_{l_t,j}, 1 \le j \le s$ is computed.
- ProofVerify: With the proof $P = (T, \eta)$ and the challenge massage $chal = (c, k_1, k_2)$, TPA computes $l_t = \pi_{k_1}(t), a_t = \phi_{k_2}(t)(1 \le t \le c)$. Then the below verification equations are checked

$$e(T, g) = e\left(\prod_{t=1}^{c}\left(H_3(name\|l_t)^{a_t}\prod_{j=1}^{s} u_j^{\eta_j}\right), v\right),$$

$$e(T, g) = \prod_{t=1}^{c} e\left(H_3(name\|l_t)^{a_t}\prod_{j=1}^{s} u_j^{\eta_j}, v^{r_t}\right).$$

If one of them passed, the proof is valid.

The reasons why this improved proposal can resist the attack above is explained as follows: From the Upd,

$$\left\{u_1^{k_{tag}(r_1(C_{\epsilon\mu})_{1,1}-(C_\epsilon)_{1,1})}, u_2^{k_{tag}(r_1(C_{\epsilon\mu})_{1,2}-(C_\epsilon)_{1,2})}, \cdots, u_s^{k_{tag}(r_1(C_{\epsilon\mu})_{1,s}-(C_\epsilon)_{1,s})}\right\},$$

$$\left\{u_1^{k_{tag}(r_2(C_{\epsilon\mu})_{2,1}-(C_\epsilon)_{2,1})}, u_2^{k_{tag}(r_2(C_{\epsilon\mu})_{2,2}-(C_\epsilon)_{2,2})}, \cdots, u_s^{k_{tag}(r_2(C_{\epsilon\mu})_{2,s}-(C_\epsilon)_{2,s})}\right\},$$

$$\cdots\cdots\cdots\cdots\cdots,$$

$$\left\{u_1^{k_{tag}(r_n(C_{\epsilon\mu})_{n,1}-(C_\epsilon)_{n,1})}, u_2^{k_{tag}(r_n(C_{\epsilon\mu})_{n,2}-(C_\epsilon)_{n,2})}, \cdots, u_s^{k_{tag}(r_n(C_{\epsilon\mu})_{n,s}-(C_\epsilon)_{n,s})}\right\},$$

the adversary can not obtain the below values anymore

$$\left\{u_1^{k_{tag}(r(C_{\epsilon\mu})_{1,1}-(C_\epsilon)_{1,1})}, u_2^{k_{tag}(r(C_{\epsilon\mu})_{1,2}-(C_\epsilon)_{1,2})}, \cdots, u_s^{k_{tag}(r(C_{\epsilon\mu})_{1,s}-(C_\epsilon)_{1,s})}\right\},$$
$$\left\{u_1^{k_{tag}(r(C_{\epsilon\mu})_{2,1}-(C_\epsilon)_{2,1})}, u_2^{k_{tag}(r(C_{\epsilon\mu})_{2,2}-(C_\epsilon)_{2,2})}, \cdots, u_s^{k_{tag}(r(C_{\epsilon\mu})_{2,s}-(C_\epsilon)_{2,s})}\right\},$$
$$\left\{u_1^{k_{tag}(r(C_{\epsilon\mu})_{3,1}-(C_\epsilon)_{3,1})}, u_2^{k_{tag}(r(C_{\epsilon\mu})_{3,2}-(C_\epsilon)_{3,2})}, \cdots, u_s^{k_{tag}(r(C_{\epsilon\mu})_{3,s}-(C_\epsilon)_{3,s})}\right\},$$
$$\cdots\cdots\cdots,$$
$$\left\{u_1^{k_{tag}(r(C_{\epsilon\mu})_{n,1}-(C_\epsilon)_{n,1})}, u_2^{k_{tag}(r(C_{\epsilon\mu})_{n,2}-(C_\epsilon)_{n,2})}, \cdots, u_s^{k_{tag}(r(C_{\epsilon\mu})_{n,s}-(C_\epsilon)_{n,s})}\right\},$$

it can only obtain

$$\left\{u_1^{k_{tag}(r_1(C_{\epsilon\mu})_{1,1}-(C_\epsilon)_{1,1})}, u_2^{k_{tag}(r_1(C_{\epsilon\mu})_{1,2}-(C_\epsilon)_{1,2})}, \cdots, u_s^{k_{tag}(r_1(C_{\epsilon\mu})_{1,s}-(C_\epsilon)_{1,s})}\right\},$$
$$\left\{u_1^{k_{tag}(r_2(C_{\epsilon\mu})_{2,1}-(C_\epsilon)_{2,1})}, u_2^{k_{tag}(r_2(C_{\epsilon\mu})_{2,2}-(C_\epsilon)_{2,2})}, \cdots, u_s^{k_{tag}(r_2(C_{\epsilon\mu})_{2,s}-(C_\epsilon)_{2,s})}\right\},$$
$$\left\{u_1^{k_{tag}(r_3(C_{\epsilon\mu})_{3,1}-(C_\epsilon)_{3,1})}, u_2^{k_{tag}(r_3(C_{\epsilon\mu})_{3,2}-(C_\epsilon)_{3,2})}, \cdots, u_s^{k_{tag}(r_3(C_{\epsilon\mu})_{3,s}-(C_\epsilon)_{3,s})}\right\},$$
$$\cdots\cdots\cdots$$

$$\left\{u_1^{k_{tag}(r_n(C_{\epsilon\mu})_{n,1}-(C_\epsilon)_{n,1})}, u_2^{k_{tag}(r_n(C_{\epsilon\mu})_{n,2}-(C_\epsilon)_{n,2})}, \cdots, u_s^{k_{tag}(r_n(C_{\epsilon\mu})_{n,s}-(C_\epsilon)_{n,s})}\right\},$$

from these values, the adversary can not compute $u_1^{r_1 k_{tag}}$, $u_1^{k_{tag}}$, $u_2^{r_2 k_{tag}}$, $u_2^{k_{tag}}$, $\cdots$, $\cdots$, $\cdots$, $u_n^{r_n k_{tag}}$, $B_s = u_2^{k_{tag}}$ anymore. Thus the above attack can not work anymore.

6. Conclusion

In 2019, Hou et al. proposed an auditing scheme. However, in this paper, we demonstrate that their proposal is not secure. The main reason for this is that the core idea of their updated authenticator algorithm is vulnerable. Specifically, if the cloud storage server obtains many values of $(u^x)^{(r\cdot C_i' - C_i)}$ for $1 \le i \le n$, it can easily compute $(u^x)^r$ and (u^x), which allows it to forge an authenticator $\sigma_i = (H(i) \cdot u^{C_{any}})^x$ for any block C_{any}. This attack is a generalization of attacks on many cloud storage auditing protocols based on the discrete logarithm hard problem, as shown in [31,32]. It highlights the need for caution when designing cloud storage auditing protocols using cryptographic techniques, as these schemes have rich algebraic structure that may result in vulnerabilities.

To address these shortcomings, we have developed an improved cloud storage auditing scheme based on Hou et al.'s proposal. Our updated authenticator algorithm now uses $(u^x)^{(r_i\cdot C_i' - C_i)}$ for $1 \le i \le n$, which makes it impossible for the adversary to compute $(u^x)^r$ and (u^x). We have also analyzed why our improved scheme is secure. We hope that our work will help future researchers avoid similar shortcomings in their own cloud storage auditing schemes.

Acknowledgments

This work is supported by the Key Research and Development Program of Xianyang City(No. L2022ZDYFSF061), Scientific Research Funding of Xianyang Vocational & Technical College on "Research on Key Technologies for Secure Outsouced Cloud Storage"(Grant No.2021KJB03).

Conflict of interest

The authors declare there is no conflict of interest.

References

1. J. Nowaková, M. Pokorný, Intelligent controller design by the artificial intelligence methods, *Sensors*, **20** (2020), 4454. https://doi.org/10.3390/s20164454
2. M. Pawlicki, R. Kozik, M. Choras, A survey on neural networks for (cyber-) security and (cyber-) security of neural networks, *Neurocomputing*, **500** (2022), 1075–1087. https://doi.org/10.1016/j.neucom.2022.06.002
3. H. Xu, M. Guo, N. Nedjah, J. Zhang, P. Li, Vehicle and Pedestrian Detection Algorithm Based on Lightweight YOLOv3-Promote and Semi-Precision Acceleration, *IEEE Trans. Intell. Transp. Syst.*, **23** (2022), 19760–19771. https://doi.org/10.1109/TITS.2021.3137253
4. B. Furht, A. Escalante, *Handbook of Cloud Computing*, Springer, 2010. https://doi.org/10.1007/978-1-4419-6524-0

5. G. Fenza, V. Loia, G. Nota, Patterns for visual management in industry 4.0, *Sensors*, **21** (2021), 6440. https://doi.org/10.3390/s21196440
6. M. Hasal, J. Nowaková, K. A. Saghair, H. M. Dahwa Abdulla, Václav Snásel, Lidia Ogiela, Chatbots: Security, privacy, data protection, and social aspects. *Concurr. Comput. Pract. Exp.*, **33** (2021). https://doi.org/10.1002/cpe.6426
7. N. Capuano, G. Fenza, V. Loia, C. Stanzione, Explainable artificial intelligence in cybersecurity: A survey. *IEEE Access*, **10** (2022), 93575–93600. https://doi.org/10.1109/ACCESS.2022.3204171
8. M. Choras, M. Wozniak, The double-edged sword of AI: ethical adversarial attacks to counter artificial intelligence for crime, *AI Ethics*, **3** (2022), 631–634. https://doi.org/10.1007/s43681-021-00113-9
9. V. Snásel, J. Nowaková, F. Xhafa, L. Barolli, Geometrical and topological approaches to big data, *Future Gener. Comput. Syst.*, **67** (2017), 286–296. https://doi.org/10.1016/j.future.2016.06.005
10. H. Hou, J. Yu, R. Hao, Cloud storage auditing with deduplication supporting different security levels according to data popularity, *J. Network Comput. Appl.*, **134** (2019), 26–39. https://doi.org/10.1016/j.jnca.2019.02.015
11. G. Asharov, G. Segev, I. Shahaf, Tight tradeoffs in searchable symmetric encryption, LNCS, Springer, Heidelberg, 2018, 407–436. https://doi.org/10.1007/978-3-319-96884-1_14
12. R. Cheng, J. Yan, C. Guan, F. Zhang, K. Ren, Verifiable searchable symmetric encryption from indistinguishability obfuscation. In Feng Bao, Steven Miller, Jianying Zhou, Gail-Joon Ahn, *ASI-ACCS 15*, ACM Press, 2015, 621–626. https://doi.org/10.1145/2714576.2714623
13. R. Curtmola, J. A. Garay, S. Kamara, R. Ostrovsky, Searchable symmetric encryption: improved definitions and efficient constructions, In Ari Juels, Rebecca N. Wright, Sabrina De Capitani di Vimercati, *ACM CCS 06*, ACM Press, 2006, 79–88. https://doi.org/10.1145/1180405.1180417
14. S. Kamara, C. Papamanthou, T. Roeder, Dynamic searchable symmetric encryption. In Ting Yu, George Danezis, Virgil D. Gligor, *ACM CCS 12*, ACM Press, October 2012, 965–976. https://doi.org/10.1145/2382196.2382298
15. K. Lee, S. G. Choi, D. H. Lee, J. H. Park, M. Yung, Self-updatable encryption: Time constrained access control with hidden attributes and better efficiency, In Kazue Sako, Palash Sarkar, *ASIACRYPT 2013, Part I*, volume 8269 of *LNCS*, Springer, Heidelberg, 2013, 235–254. https://doi.org/10.1007/978-3-642-32009-5_13
16. A. Sahai, H. Seyalioglu, B. Waters, Dynamic credentials and ciphertext delegation for attribute-based encryption, In Reihaneh Safavi-Naini, Ran Canetti, *CRYPTO 2012*, volume 7417 of *LNCS*, Springer, Heidelberg, 2012, 199–217. https://doi.org/10.1007/978-3-642-32009-5_13
17. M. Bellare, S. Keelveedhi, T. Ristenpart, Message-locked encryption and secure deduplication, In Thomas Johansson, Phong Q. Nguyen, *EUROCRYPT 2013*, volume 7881 of *LNCS*, Springer, Heidelberg, 2013, 296–312. https://doi.org/10.1007/978-3-642-38348-9_18
18. M. Bellare, S. Keelveedhi, Interactive message-locked encryption and secure deduplication. In Jonathan Katz, *PKC 2015*, volume 9020 of *LNCS*, Springer, Heidelberg, 2015, 516–538. https://doi.org/10.1007/978-3-662-46447-2_23

19. G. Ateniese, K. Fu, M. Green, S. Hohenberger, Improved proxy re-encryption schemes with applications to secure distributed storage, In *NDSS 2005*, The Internet Society, February 2005.

20. G. Ateniese, R. C. Burns, R. Curtmola, J. Herring, L. Kissner, Z. N. J. Peterson, et al., Provable data possession at untrusted stores, In Peng Ning, Sabrina De Capitani di Vimercati, Paul F. Syverson, *ACM CCS 07*, ACM Press, 2007, 598–609. https://doi.org/10.1145/1315245.1315318

21. C. C. Erway, A. Kupccu, C. Papamanthou, R. Tamassia, Dynamic provable data possession, In Ehab Al-Shaer, Somesh Jha, Angelos D. Keromytis, *ACM CCS 09*, ACM Press, 2009, 213–222. https://doi.org/10.1145/1653662.1653688

22. A. Juels, B. S. Kaliski Jr., Pors: Proofs of retrievability for large files. In Peng Ning, Sabrina De Capitani di Vimercati, Paul F. Syverson, *ACM CCS 07*, ACM Press, 2007, 584–597. https://doi.org/10.1145/1315245.1315317

23. E. Shi, E. Stefanov, C. Papamanthou, Practical dynamic proofs of retrievability, In Ahmad-Reza Sadeghi, Virgil D. Gligor, Moti Yung, *ACM CCS 13*, ACM Press, 2013, 325–336. https://doi.org/10.1145/2508859.2516669

24. H. Shacham, B. Waters, Compact proofs of retrievability, In Josef Pieprzyk, *ASIACRYPT 2008*, volume 5350 of *LNCS*, Springer, Heidelberg, 2008, 90–107. https://doi.org/10.1007/978-3-540-89255-7_7

25. Q. Wang, C. Wang, J. Li, K. Ren, W. Lou, Enabling public verifiability and data dynamics for storage security in cloud computing, In *ESORICS*, volume 5789 of *Lecture Notes in Computer Science*, Springer, 2009, 355–370. https://doi.org/10.1007/978-3-642-04444-1_22

26. Q. Wang, C. Wang, K. Ren, W. Lou, J. Li, Enabling public auditability and data dynamics for storage security in cloud computing. *IEEE Trans. Parallel Distrib. Syst.*, 22(5):847–859, 2011. https://doi.org/10.1109/TPDS.2010.183

27. Y. Yu, L. Xue, M. H. Au, W. Susilo, J. Ni, Y. Zhang, et al., Cloud data integrity checking with an identity-based auditing mechanism from RSA, *Future Generation Comp. Syst.*, **62** (2016), 85–91. https://doi.org/10.1016/j.future.2016.02.003

28. Y. Yu, M. H. Au, G. Ateniese, X. Huang, W. Susilo, Y. Dai, et al., Identity-based remote data integrity checking with perfect data privacy preserving for cloud storage, *IEEE Trans. Inf. Forens. Secur.*, **12** (2017), 767–778. 2017. https://doi.org/10.1109/TIFS.2016.2615853

29. Y. Yu, Y. Li, B. Yang, W. Susilo, G. Yang, J. Bai, Attribute-based cloud data integrity auditing for secure outsourced storage, *IEEE Trans. Emerg. Top. Comput.*, **8** (2020), 377–390. https://doi.org/10.1109/TETC.2017.2759329

30. Y. Huang, Y. Yu, H. Li, Y. Li, A. Tian, Blockchain-based continuous data integrity checking protocol with zero-knowledge privacy protection, *Digit. Commun. Networks*, **8** (2022), 604–613. https://doi.org/10.1016/j.dcan.2022.04.017

31. J. Zhang, B. Wang, X. A. Wang, H. Wang, S. Xiao, New group user based privacy preserving cloud auditing protocol. *Future Gener. Comput. Syst.*, **106** (2020), 585–594. https://doi.org/10.1016/j.future.2020.01.029

32. J. Zhang, B. Wang, M. R. Ogiela, X. A. Wang, A. K. Sangaiah, New public auditing protocol

based on homomorphic tags for secure cloud storage, *Concurr. Comput. Pract. Exp.*, **32** (2020). https://doi.org/10.1002/cpe.5600

AIMS Press