Pers Ubiquit Comput (2017) 21:949–962
DOI 10.1007/s00779-017-1048-7

ORIGINAL ARTICLE

Secure public data auditing scheme for cloud storage in smart city

Libing Wu[1] · Jing Wang[1] · Neeraj Kumar[2] · Debiao He[1,3]

Received: 5 September 2016 / Accepted: 2 April 2017 / Published online: 1 July 2017

Abstract In the smart city construction, massive data collected from various fields need to be outsourced to the cloud for convenience and resource saving. However, integrity and confidentiality of the data in cloud remains a challenge issue due to the loss of data possession. As a solution, some public data auditing schemes have been proposed in last several years. Most recently, Li et al. proposed an efficient public auditing scheme and claimed that it could reduce the cost of clients on generating verification metadata. In this paper, we analyze the security of Li et al.'s scheme and point out two weaknesses in it. We demonstrate that it cannot achieve the confidentiality for outsourced data and it is vulnerable to the proof forgery attack. To address these weaknesses, we propose an improved public auditing scheme, which can not only preserve the data privacy but also resist the proof forgery attack. Security analysis shows that our scheme is provably secure in a robust security model. Performance analysis shows that the proposed scheme can overcome the weaknesses in Li et al.'s scheme at the cost of increasing computation overhead slightly.

Keywords Smart city · Cloud storage · Outsourced data · Public integrity auditing · Privacy preserving

✉ Debiao He
hedebiao@163.com

Libing Wu
whuwlb@126.com

Jing Wang
cswjing@whu.edu.cn

Neeraj Kumar
neeraj.kumar@thapar.edu

[1] State Key Lab of Software Engineering, Computer School, Wuhan University, Wuhan, China

[2] Department of Computer Science and Engineering, Thapar University, Patiala, India

[3] Guangxi Key Laboratory of Cryptography and Information Security, Guilin University Of Electronic Technology, Guilin, China

1 Introduction

Smart city, as the most creative city urban morphology, has become a global strategic choice of urban development. In essence, the smart city mainly uses information and communication technologies (ICT) to achieve the intelligent management of various fields for cities, and then create a better life for urban citizens [1]. As shown in Fig. 1, a smart city involves all sorts of smart domains, such as smart transportation, smart care, smart education, and smart economy. And these smart systems form the backbone of a city's livability, efficiency and sustainability [2]. Take the smart care as an example, people living in the smart city can make a reservation registration on their mobile phone at home instead of registering in line at hospital, which brings much convenience for both patients and hospitals. Moreover, electronic medical records (EMR) enable the authorized doctors to diagnose patients at anytime and anywhere.

To provide accurate smart city services, all kinds of data should be collected and analyzed with advanced data processing technologies [3], as shown in Fig. 2. In other words, data can be considered as the core of a smart city. Therefore, there should be an efficient infrastructure with powerful ability in storing and sharing data for archival, online, and offline analytics. Just like EMR, the medical institution

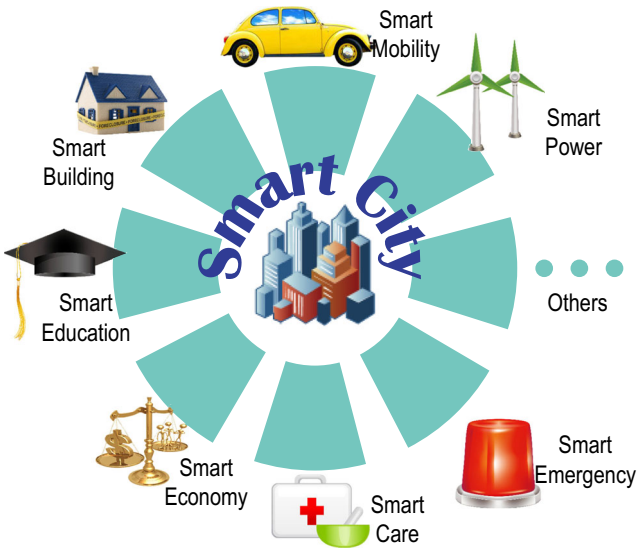

Fig. 1 The structure of smart city

must collect and store many medical recorders for a large number of patients. Taking the reliability, scalability, and cost minimization of the data into account, it is a good choice to store these data in open source cloud-based data storage servers [4].

As described above, cloud storage plays an increasingly important role in smart city construction while facing with the explosive growth of the data amount in Big Data Age. To enjoy the smart city services better, organizations and individuals have to move their massive data into the cloud servers since they lack powerful storage and management abilities. By delegating the data to a cloud server, organizations and individuals not only liberate themselves from anguishing about the management of complex hardware system, but also do not worry about their data backup. What's more, they can get access to their data in the cloud at anytime/anywhere.

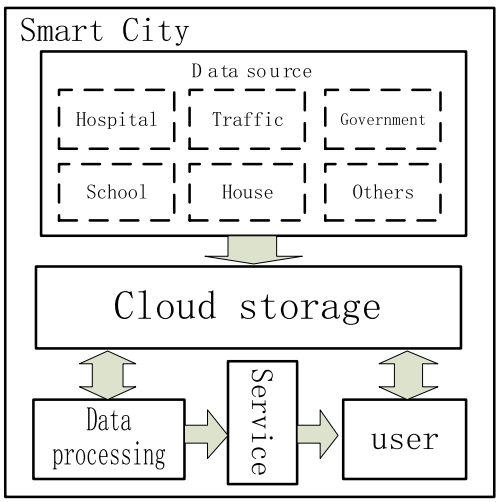

Fig. 2 The cloud storage in smart city

Along with the unprecedented convenience brought by cloud storage, some significant security and privacy challenges come into being. Once people living in smart city outsource their data to the cloud server, they have to give up the physical possession of their outsourced data and authorize the cloud service providers(CSPs) to implement some fundamental operation on data. It means that users cannot perform data modification, insertion and deletion operations on data in their local system, but the cloud service provider can perform these even without informing the data owners. For example, CSPs can delete the outsourced data that seems to be expired or infrequently accessed for economies. Additionally, even if the CSPs are honest in terms of storing users' data, they still cannot avoid some inevitable software/hardware breakdowns [5, 6], which leads to some data corruptions.

When the above problems occur, CSPs will try to hide these corruptions out of some benefits, i.e., reputation, economy, and then driving the data owners to believe that their outsourced data are stored correctly in the cloud storage server. As a consequence, checking data integrity is necessary for ensuring the data security [7, 8]. However, it is not efficient for some traditional cryptographic checking methods to check data integrity, because the users do not physically possess their data. Since the data are the core of a smart city, ensuring the data security is the basic premise to achieve the sustainable development of smart city. Hence, an efficient and secure data auditing mechanism without downloading the whole data is required to guarantee the data integrity at cloud server.

In recent years, researchers have proposed many different schemes including private auditing [9] and public auditing [10–12] to audit the integrity of remote data in cloud. Private auditing is necessary in some cases, such as the conditions where the data owners must be strictly limited to access the Internet [13]. As for public auditing schemes, they allow both the data owners and other authorized third parties as auditors to verify the data integrity in cloud computing environment when compared with private auditing. Additionally, it is convenient for users to delegate the auditing assignment to a trusted TPA when they cannot audit in person, and users do not need to afford any fee for their auditing request. Hence, public auditing mechanism seemed more practical in terms of verifying the data integrity of cloud storage [11].

1.1 Related work

Cryptographic schemes are very suitable for implementing secure communication; a lot of encryption schemes [14–16], authentication schemes [17–19], and digital signature schemes [20–22] have been proposed for practical applications. To ensure the integrity of the outsourced data in smart

city, a number of schemes [10–13, 23–27] have been proposed. Ateniese et al.[23] was the first to design a provable data possession (PDP) model for public auditing, and it was the primitive security model to verify storage correctness on untrusted cloud. They utilized RSA-based homomorphic authentication and enabled the cloud users to verify the integrity of outsourced data without downloading the whole data file. However, it did not satisfy dynamic storage files. In 2007, Juels and Kaliski first proposed Proofs of Retrievability(POR) [24] which was another method to audit the data integrity in semi-trusted cloud servers. But it could only deal with limited number of queries.

Based on the [23] and [24], there are a series of improved public auditing schemes proposed by other researchers. For example, compact proofs of retrievability(CPOR) [25] was a public auditing scheme based on BLS signatures [28]. Yuan et al. [10] proposed another public auditing scheme whose communication cost is constant. Wang et al. [29] designed a public auditing scheme with group dynamic which is efficiently solved by outsourcing signature updating operations to the cloud via a secure proxy re-signature scheme. In [30], Wang et al. came up with a certificateless public auditing scheme to enhance the security in certificate management.

Recently, Wang et al. proposed the scheme Panda [31], which is an auditing method with efficient user revocation and secure public verification for outsourced data in cloud. But Yang et al. [32] pointed out that Panda is vulnerable to data recovery attack and it is not secure enough in resisting proof forgery attack. In fact,the variants of Panda, i.e. [33, 34] proposed by Wang et al. suffer from the two vulnerabilities as well. Then in [32], Yang et al. proposed an improved public auditing scheme based on Panda by utilizing random masking technology. However, their method incurred some extra communication and computation overhead, especially there were some complicated inverse operations during the verification process of TPA.

Most recently, Li et al. proposed another provable data integrity scheme [35], which mainly reduced the cost in the process of generating verification metadata at the client to achieve high efficiency. Unfortunately, we find that their scheme also encounters with the security threats about resisting data recovery attack and proof forgery attack. Thus, we propose a new scheme to enhance the security of [35] with the random masking technology, and we eliminate the complicated inverse operations compared with [32]. Besides, the improvements can also be applied to some other public auditing schemes, i.e. [36], which is an identity-based public auditing mechanism from RSA.

1.2 Our contribution

In this paper, we mainly analyze the scheme proposed by Li et al. [35] and propose an improved data auditing scheme for checking the storage correctness of outsourced data, which inherits all advantages of [35] and further refines it by improving its security. To be specific, the three major contributions are presented as follows:

- Firstly, we give a security analysis to the scheme in [35], and prove that there are two typical security drawbacks. One is it cannot resist data recovery attack, i.e., an external attacker can impersonate a public auditor to extract the entire outsourced data. The other one is it is not secure enough to resist proof forgery attack, i.e., a malicious cloud server can generate a valid auditing proof without correctly storing the outsourced data.
- Secondly, we propose an improved public data auditing scheme, and the security analysis demonstrates that it strongly satisfies the security and privacy requirements of public data auditing. In particular, our schemes avoid the vulnerabilities in resisting the data recovery attack and proof forgery attack. What is more, the improvements and analysis can also be applied to other public auditing schemes.
- Finally, we present a detailed analysis in terms of the communication cost and computation cost to show that the proposed scheme can greatly improve the security at the expense of increasing communication and computation cost slightly.

1.3 Organization of this paper

The rest of this paper is organized as follows. Section 2 introduces the mathematical background and system models related to this paper. Section 3 makes a brief review to Li et al.'s scheme [35]. Section 4 presents the details about how to implement the two attacks including data privacy recovery attack and proof forgery attack. Section 5 describes the improved public auditing scheme on the basis of [35]. Then, Section 6 analyzes the security of the proposed scheme, and Section 7 evaluates the communication cost and computation cost of the proposed auditing scheme. Section 8 makes some concluding remarks at last.

2 Preliminaries

The necessary cryptographic primitives, system model, and the security requirements are described in this section as follows.

2.1 Mathematical background

1) **Bilinear Maps.** In order to enhance the readability of this article, some mathematical properties of the bilinear maps are given below.

Let G, G_T be two cyclic groups with the same large prime order p. And let $e : G \times G \longrightarrow G_T$ denote a bilinear map with the following properties:

- ***Bilinearity*** For all $P, Q \in G$ and $a, b \in Z_p$, the equation $e(P^a, Q^b) = e(P, Q)^{ab}$ holds.
- ***Nondegeneracy*** For at least one element $P \in G$, the inequality $e(P, P) \neq 1_{G_T}$ holds.
- ***Computability*** For any two elements $P, Q \in G$, we can always find an efficient algorithm to compute the map $e(P, Q)$.

2) **Security Assumptions.** The security of our auditing scheme is supported by the two mathematical assumptions as below.

- ***Computation Diffie-Hellman (CDH) Assumption*** Given $g, g^a, g^b \in G$, where a, b are two unknown elements in Z_p, there is no polynomial algorithm can calculate g^{ab} with a non-negligible probability.
- ***Discrete Logarithm (DL) Assumption*** Given an element $g^a \in G$, it is computationally infeasible to get the value of a.

2.2 System model

As shown in Fig. 3, the system model in this paper consists of three entities: the user, the cloud server, and the public auditor. Each entity is specifically defined as follows.

The user The user is someone who owns a mass of data and outsources his/her data to the cloud after generating a signature for each data block. Then with the help of various interfaces provided by the cloud server, he/she can access the outsourced data at anytime.

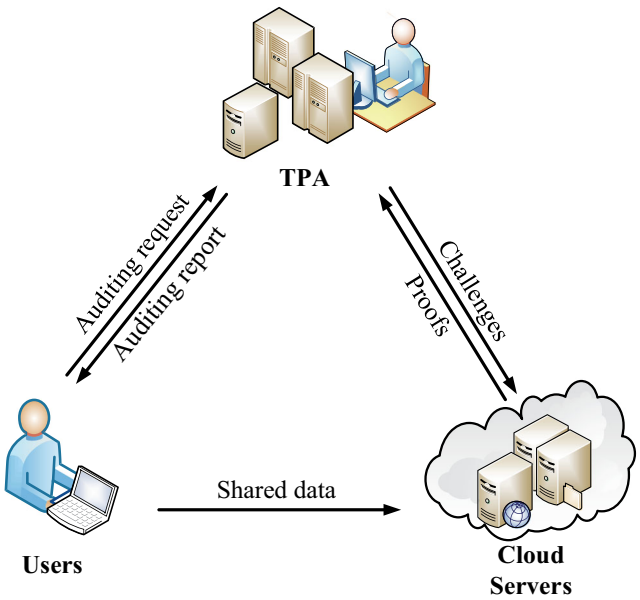

Fig. 3 System model

The cloud server The cloud server is a distributed storage system which has advantages in providing data storing, complex calculating, and data sharing for users.

The public auditor The public auditor is the data owners or other third-party verifiers, who execute the data integrity verification process to check the storage correctness of the outsourced data.

In this system model, the user can upload his/her data to the cloud server and delete the original data from the local memory, which greatly reduces the storage burden for users. Then the user can delegate the auditing task to a trusted TPA or act as a verifier himself/herself to verify the validity of outsourced data. Each data block gets stored with a signature signed by the private key of data owners.

2.3 Threat models

- ***Integrity threat*** Under the security model proposed by Shacham and Waters for integrity checking in [25], the user or another third-party verifier can challenge the cloud server to examine whether the outsourced data are stored correctly in cloud. Generally, the integrity of outsourced data in cloud may encounter two types of threats. The first one is that an attacker can generate a set of forged signatures for all data blocks, and then make use of the signature set to return a forged proof; The other one is that the malicious attacker can forge the aggregated information of proof about the sampled data blocks to pass the verification of public auditors.
- ***Privacy threat*** For each auditing mechanism, it is important to ensure there is no one who can get access to the outsourced data in cloud without the data owner's authorization, which means the public auditor is only allowed to verify the integrity with no data leakage. If someone can extract the details of outsourced data from the auditing information, the security model suffers from data privacy attack.

2.4 Design goals

To achieve an efficient and secure data integrity checking for public cloud storage referring to the aforementioned schemes, the method of Li et al.'s and our improved scheme should satisfy the following security and performance goals.

- ***Public auditability*** The public auditor is allowed to verify the integrity of outsourced data stored in public cloud.
- ***Storage correctness*** No malicious cloud server is able to pass the public verifier's auditing but for storing the outsourced data correctly.

- ***Dynamic operation*** In the auditing scheme, users are allowed to execute block-level operations on the outsourced data, and provide correctness guarantee of the changed file.
- ***Privacy preserving*** It is crucial for all auditing schemes that the public verifier cannot reveal the whole outsourced data even some data blocks via the auditing information collected during the integrity checking process. What we can know from later security analysis is that the auditing method in Li et al.'s scheme suffers from this weakness, and this goal can be realized in our advanced scheme.

3 Review of Li et al.'s scheme

In this section, we will review Li et al.'s scheme [35] on achieving provable data integrity in the public cloud storage, which consists of four algorithms: $KeyGen(1^{\lambda}) \longrightarrow (pk, sk)$, $SignGen(pk, sk, F) \longrightarrow \Phi$, $GenProof(F, \Phi, chal) \longrightarrow P$, $CheckProof(pk, chal, P) \longrightarrow \{'0', '1'\}$. The main process of verification is showed in Fig. 4, and the scheme details are described as below.

Let p be a large prime, which is the order of two multiplicative groups G and G_T, and let g be a generator of group G. $G \times G \longrightarrow G_T$ is defined as a bilinear map, and $H : \{0,1\}^* \longrightarrow G$ denotes a hash function, which maps strings uniformly to group G.

Firstly, the user executes $KeyGen$ to generate a key pair (sk, pk), where sk is a random element $x \in Z_p$ chosen by the user, and $pk = g^x$. Then, based on the scheme of BLS signatures proposed in [25, 26], let data file F be divided into n blocks, and each block is further split into s sectors. Thus, F can be denoted as below:

$$F = \begin{pmatrix} \overrightarrow{m_1} \\ \vdots \\ \overrightarrow{m_n} \end{pmatrix} = \begin{pmatrix} m_{11} & \dots & m_{1s} \\ \vdots & \ddots & \vdots \\ m_{n1} & \dots & m_{ns} \end{pmatrix} \in Z_p^{n \times s}$$

Now, the user execute $SignGen$ to generate a signature for each data block. First, the user randomly chooses s elements $\alpha_1, \dots, \alpha_s \in Z_p$ to compute $w_j = g^{\alpha_j}$, then for each data block $\overrightarrow{m_i}$, the user can get the corresponding signature:

$$\sigma_i = (H(id_i) \cdot g^{\sum_{j=1}^{s} \alpha_j m_{ij}})^{sk}, 1 \le i \le s \quad (1)$$

where $id_i = filename \,||\, i$ denotes the identifier of data block m_i, $filename$ is the name of outsourced data F. $\Phi = \{\sigma_i\}_{1 \le i \le n}$ denotes the signature set of outsourced data blocks.Then, the user delivers his file F and corresponding signature Φ set to the public cloud and removes the data copy from his local memory. Meanwhile, the user should send the message $\{filename, n, \{w_j\}_{1 \le j \le s}\}$ to the TPA.

While the TPA intends to check the data integrity in the cloud, it first randomly chooses c-element subset $I = \{l_1, \dots, l_c\}$ from the set $[1, n]$ as the samples to be checked. Then, the TPA continues to choose c random elements

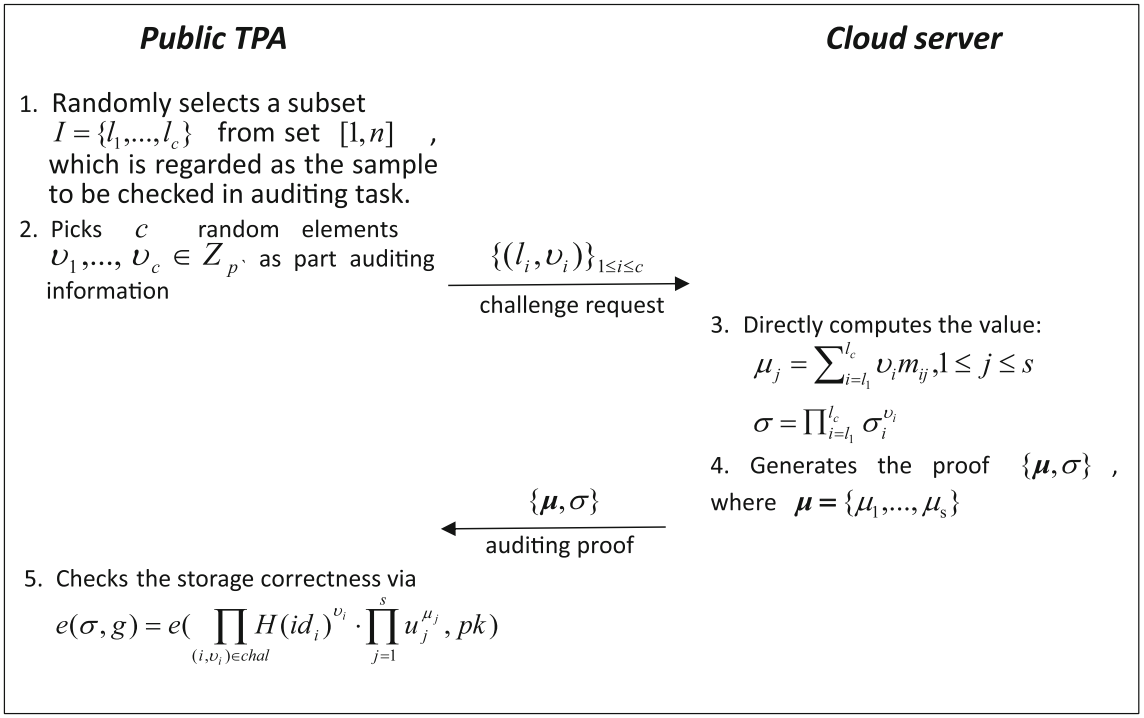

Fig. 4 The auditing process of Li et al.'s scheme

from Z_p, which is $\upsilon_{l_1}, \ldots, \upsilon_{l_c}$. Finally, the TPA sends the challenge request $chal = \{(i, \upsilon_i)_{l_i \le i \le l_c}\}$ to the cloud server.

Once the cloud server obtains the auditing request $chal = \{(i, \upsilon_i)_{l_i \le i \le l_c}\}$, it executes the algorithm $ProofGen$ to generate a corresponding auditing proof. The proof consists of two parts, which are

$$\mu_j = \sum_{i=l_1}^{l_c} \upsilon_i m_{ij}, 1 \le j \le s \tag{2}$$

$$\sigma = \prod_{i=l_1}^{l_c} \sigma_i^{\upsilon_i} \tag{3}$$

Let the set $\{\mu, \sigma\}$, where $\mu = \{\mu_1, \ldots, \mu_s\}$, be the response proof and be sent to the TPA. Then, the TPA can verify the storage correctness of outsourced data by running $CheckProof$ via the following equation:

$$e(\sigma, g) \stackrel{?}{=} e(\prod_{(i,\upsilon_i)\in chal} H(id_i)^{\upsilon_i} \cdot \prod_{j=1}^{s} w_j^{\mu_j}, pk) \tag{4}$$

4 Weaknesses of Li et al.'s scheme

As described in [35], any client can check the correctness of the data in cloud by acting as a public auditor. Since the public auditor is not fully trusted, it is possible for him/her to achieve his/her own purpose such as recovering the outsourced data by collecting enough auditing information. What is more, the cloud server is semi-trusted, so it may cheat the verifier by giving a forged proof for its benefits and reputation when it deletes some data blocks or hides some data corruptions.

The scheme [35] claimed that the scheme could meet these design requirements as follows: public auditing, storage correctness, dynamic operation, batch auditing, and lightweight. However, we find that the scheme suffers from data privacy attack and proof forgery attack, which are mainly caused by an external attacker (curious TPA) and the malicious cloud server, respectively. The detailed attack process is shown as follows.

4.1 Data privacy attack

We assume that the curious public verifier is an attacker, who intends to obtain the outsourced data without the owner's authorization. Then, we prove that it can achieve this target by collecting enough public auditing information pairs from the cloud server.

Let the c-element subset $I = \{l_1, \ldots, l_c\}$ of set $[1, n]$ chosen by the attacker denote as follows:

$$F' = \begin{pmatrix} \overrightarrow{m_{l_1}} \\ \vdots \\ \overrightarrow{m_{l_c}} \end{pmatrix} = \begin{pmatrix} m_{l_1 1} & \ldots & m_{l_1 s} \\ \vdots & \ddots & \vdots \\ m_{l_c 1} & \ldots & m_{l_c s} \end{pmatrix} = (\mathbf{F_1} \ \ldots \ \mathbf{F_s})$$

For the sake of simplicity, let us take the data fragmentation $\mathbf{F_1}$ as a target that the attacker intends to reveal for example, where

$$\mathbf{F_1}^{\mathrm{T}} = \{m_{l_1 1}, m_{l_2 1}, \ldots, m_{l_c 1}\}$$

After the user accomplishes the process of $KeyGen$, $SignGen$ and outsources their data into cloud, the attacker can disguise as an auditor to perform challenge-response phase with the cloud server for at least c times. And the c-challenges it sends to the cloud server can be presented as

$$\begin{cases} chal_1 = \{(l_1, \upsilon_{1l_1}), (l_2, \upsilon_{1l_2}), \cdots, (l_c, \upsilon_{1l_c})\} \\ chal_2 = \{(l_1, \upsilon_{2l_1}), (l_2, \upsilon_{2l_2}), \cdots, (l_c, \upsilon_{2l_c})\} \\ \quad \ldots\ldots \\ chal_c = \{(l_1, \upsilon_{cl_1}), (l_2, \upsilon_{cl_2}), \cdots, (l_c, \upsilon_{cl_c})\} \end{cases}$$

In return, it can receive c times corresponding proofs: $pf_1 = (\mu_{\mathbf{1}}, \sigma_1)$, $pf_2 = (\mu_{\mathbf{2}}, \sigma_2), \ldots, pf_c = (\mu_{\mathbf{c}}, \sigma_c)$.

Because the attacker only targets at the data fragment F_1 firstly, there is merely one element in $\mu_{\mathbf{i}}$ as $\mu_{\mathbf{i}} = \{\mu_{1i}\}$, where $\mu_{1i} = \sum_{j=1}^{c} \upsilon_{il_j} m_{l_j 1}$, $1 \leqslant i \leqslant c$. Assume that $\upsilon_{\mathbf{1}} = (\upsilon_{1l_1}, \upsilon_{1l_2}, \ldots, \upsilon_{1l_c})$, $\upsilon_{\mathbf{2}} = (\upsilon_{2l_1}, \upsilon_{2l_2}, \ldots, \upsilon_{2l_c}), \ldots, \upsilon_{\mathbf{c}} = (\upsilon_{cl_1}, \upsilon_{cl_2}, \ldots, \upsilon_{cl_c})$ and the construction of matrix υ is

$$\mathbf{V} = \begin{pmatrix} \upsilon_{1l_1} & \upsilon_{1l_2} & \ldots & \upsilon_{1l_c} \\ \upsilon_{2l_1} & \upsilon_{2l_2} & \ldots & \upsilon_{2l_c} \\ \vdots & \vdots & \ddots & \vdots \\ \upsilon_{cl_1} & \upsilon_{cl_2} & \ldots & \upsilon_{cl_c} \end{pmatrix}$$

Let vectors $\upsilon_{\mathbf{1}}, \upsilon_{\mathbf{2}}, \ldots, \upsilon_{\mathbf{c}}$ be linearly independent, so there exists a matrix $\mathbf{R}$ that meets the requirement of $\mathbf{RV} = \mathbf{E}$, where $\mathbf{E}$ is a unit matrix.

Assume that $\mu = \{\mu_{11}, \mu_{12}, \ldots, \mu_{1c}\}$, since $\mu = \mathbf{VF_1}$, the external attacker can reveal $\mathbf{F_1}$ by computing $\mathbf{F_1} = \mathbf{R}\mu$. In the same way, the attacker can derive other data fragments such as $\mathbf{F_2}$, …, $\mathbf{F_s}$. What's more, it can furtherly derive the data blocks m_i for $i \notin I$ by implementing other corresponding challenge-response operations.

As a matter of fact, even though the attacker cannot play the role of an auditor in some limited conditions, it is still able to recover the matrix of $\mathbf{m_i}$, provided that it can eavesdrop or intercept the message of challenge-response. So, Li et al.'s scheme fails to resist the attack in data privacy preserving.

4.2 Proof forgery attack

We assume that the malicious cloud is an internal attacker who deletes the outsourced data owned by users or incorrectly stores the original data out of some interests. Then, we demonstrate that a malicious cloud can generate a forged auditing proof corresponding to the public auditor's challenges and pass the verification successfully even without correct data storage.

Similar to the data recovery attack, let the c-element $I = \{l_1, l_2, \ldots, l_c\}$ from set $[1, n]$ chosen by attacker present as

$$F' = \begin{pmatrix} \overrightarrow{m_{l_1}} \\ \vdots \\ \overrightarrow{m_{l_c}} \end{pmatrix} = \begin{pmatrix} m_{l_1 1} & \ldots & m_{l_1 s} \\ \vdots & \ddots & \vdots \\ m_{l_c 1} & \ldots & m_{l_c s} \end{pmatrix} = (\mathbf{F_1} \ \ldots \ \mathbf{F_s})$$

For the sake of simplicity, we still take the fragmentation $\mathbf{F_1}$ as a target that needs to be audited by a auditor, where $\mathbf{F_1}^{\mathrm{T}} = \{m_{l_1 1}, m_{l_2 1}, \ldots, m_{l_c 1}\}$. Suppose that the auditor has performed c times challenge-response phase with the cloud server and the corresponding c-challenges are

$$\begin{cases} chal_1 = \{(l_1, \upsilon_{1l_1}), (l_2, \upsilon_{1l_2}), \cdots, (l_c, \upsilon_{1l_c})\} \\ chal_2 = \{(l_1, \upsilon_{2l_1}), (l_2, \upsilon_{2l_2}), \cdots, (l_c, \upsilon_{2l_c})\} \\ \quad \ldots\ldots \\ chal_c = \{(l_1, \upsilon_{cl_1}), (l_2, \upsilon_{cl_2}), \cdots, (l_c, \upsilon_{cl_c})\} \end{cases}$$

Then, the malicious cloud outputs corresponding auditing proofs as $pf_1 = (\mu_{\mathbf{1}}, \sigma_1')$, $pf_2 = (\mu_{\mathbf{2}}, \sigma_2')$, …, $pf_c = (\mu_{\mathbf{c}}, \sigma_c')$.

In the same way, because the malicious cloud server firstly targets at the data fragment F_1, there is merely one element in $\mu_{\mathbf{i}}$ as $\mu_{\mathbf{i}} = \{\mu_{1i}\}$. And the value of μ_{1i} and σ_j' are respectively denoted as

$$\begin{aligned} \mu_{1i} &= \sum_{k=1}^{c} \upsilon_{il_k} m_{l_k 1}, 1 \leqslant i \leqslant c \\ \sigma_j' &= \prod_{i=1}^{c} \sigma_{l_i}^{\upsilon_{jl_i}}, 1 \leqslant j \leqslant c \end{aligned} \tag{5}$$

Assume that $\upsilon_{\mathbf{1}} = (\upsilon_{1l_1}, \upsilon_{1l_2}, \ldots, \upsilon_{1l_c})$, $\upsilon_{\mathbf{2}} = (\upsilon_{2l_1}, \upsilon_{2l_2}, \ldots, \upsilon_{2l_c})$, …,$\upsilon_{\mathbf{c}} = (\upsilon_{cl_1}, \upsilon_{cl_2}, \ldots, \upsilon_{cl_c})$ and the construction of matrix υ is

$$\mathbf{V} = \begin{pmatrix} \upsilon_{1l_1} & \upsilon_{1l_2} & \ldots & \upsilon_{1l_c} \\ \upsilon_{2l_1} & \upsilon_{2l_2} & \ldots & \upsilon_{2l_c} \\ \vdots & \vdots & \ddots & \vdots \\ \upsilon_{cl_1} & \upsilon_{cl_2} & \ldots & \upsilon_{cl_c} \end{pmatrix}$$

If $det(\mathbf{V}) \neq 0$, vectors $\upsilon_{\mathbf{1}}, \upsilon_{\mathbf{2}}, \ldots, \upsilon_{\mathbf{c}}$ are linearly independent, we can prove that the malicious cloud can generate valid auditing proofs by utilizing these kinds of c pairs of auditing requests and responses after it deletes or destroys the user's data.

Given that the malicious cloud server has stored c pairs of auditing messages $(chal_i, pf_i)$ for $1 \leqslant i \leqslant c$. Now it receives a new auditing challenge set $chal^* = \{(l_1, \upsilon_{l_1}^*), (l_2, \upsilon_{l_2}^*), \ldots, (l_c, \upsilon_{l_c}^*)\}$ to the data block set $\mathbf{F}'$, then it forges a valid auditing proof as following steps:

(1) Since $det(\mathbf{V}) \neq 0$ and $\upsilon^* = (\upsilon_{l_1}^*, \upsilon_{l_2}^*, \ldots, \upsilon_{l_c}^*)$, the malicious cloud can find a data set $\{a_1, a_2, \ldots, a_c\}$ that satisfies $\upsilon^* = a_1 \upsilon_{\mathbf{l_1}} + a_2 \upsilon_{\mathbf{l_2}} + \cdots + a_c \upsilon_{\mathbf{l_c}}$, that is, $\upsilon_{l_i}^* = \sum_{j=1}^{c} a_j \upsilon_{jl_i}$.

(2) The malicious cloud computes $\mu_1^* = \sum_{i=1}^{c} a_i \mu_{1i}$, $\sigma^* = \prod_{i=1}^{c} \sigma_{l_i}^{\upsilon_{l_i}^*}$. In the same way, it can compute μ_2^*, …, μ_s^*, where $\mu_j^* = \sum_{i=1}^{c} a_i \mu_{ji}$. At last, it outputs $\{\mu^*, \sigma^*\}$ as the proof.

The public verifier checks the correctness of auditing proofs by verifying the equation $e(\sigma, g) = e(\prod_{(l_i, \upsilon_i) \in chal} H(id_i)^{\upsilon_i} \cdot \prod_{j=1}^{s} w_j^{\mu_j}, pk)$. Therefore, the proof $\{\mu^*, \sigma^*\}$ can pass the checking due to the computing process below:

$$\begin{aligned} e(\sigma^*, g) &= e(\prod_{i=1}^{c} \sigma_{l_i}^{\upsilon_{l_i}^*}, g) = e(\prod_{i=1}^{c} \sigma_{l_i}^{\sum_{j=1}^{c} a_j \upsilon_{jl_i}}, g) \\ &= e(\prod_{i=l_1}^{l_c} (H(id_i) \cdot g^{\sum_{j=1}^{s} \alpha_j m_{ij}})^{sk \cdot \sum_{j=1}^{c} a_i \upsilon_{ji}}, g) \\ &= e(\prod_{i=l_1}^{l_c} H(id_i)^{\upsilon_i^*} \cdot \prod_{i=l_1}^{l_c} g^{\sum_{j=1}^{s} \alpha_j m_{ij} \sum_{k=1}^{c} a_k \upsilon_{il_k}}, g^{sk}) \\ &= e(\prod_{i=l_1}^{l_c} H(id_i)^{\upsilon_i^*} \cdot \prod_{i=l_1}^{l_c} (\prod_{j=1}^{s} w_j^{m_{ij}})^{\sum_{k=1}^{c} a_k \upsilon_{il_k}}, pk) \\ &= e(\prod_{i=l_1}^{l_c} H(id_i)^{\upsilon_i^*} \cdot \prod_{j=1}^{s} w_j^{\sum_{i=l_1}^{l_c} m_{ij} \sum_{k=1}^{c} a_k \upsilon_{il_k}}, pk) \\ &= e(\prod_{(i, \upsilon_i) \in chal} H(id_i)^{\upsilon_i^*} \cdot \prod_{j=1}^{s} w_j^{\mu_j^*}, pk) \end{aligned} \tag{6}$$

Since the auditing challenges from public verifiers are simply arranged for data blocks, it is very simple to segregate the auditing challenges according to the identifier of each data block and match them with corresponding auditing proofs in the similar manner. Thus, if the malicious cloud server firstly collects enough pairs of auditing messages, then deletes some data blocks for its own benefits, it can generate a valid auditing proof (μ^*, σ^*) by inducing the partial auditing proofs and challenges from its collections.

What is worse, an external attacker, who does not possess the outsourced data initially, can also generate an auditing proof of forgery corresponding to any challenges

once he/she eavesdrops on enough valid pairs of challenge-proofs. In other words, any external attacker can pretend to be the cloud server when he/she obtains enough auditing messages by some way. Obviously, both users and cloud servers may suffer from unexpected risks caused by this serious security flaw.

5 Our proposed scheme

As described above, the flaw of Li et al.'s scheme is that the public auditing method cannot guarantee the data confidentiality when an exploitative verifier manages the auditing process; even that it cannot guarantee the correctness of the auditing proof when the malicious cloud attempts to give a counterfeited proof due to some considerations over the economy and reputation problems. Therefore, data privacy preserving as another important requirement should be added into our improved scheme to make sure the public auditor is incapable to extract the original data blocks or files no matter how many operations he/she does during the integrity auditing phase, and we further enhance the security over the auditing proof forgery attack in the new proposed scheme.

According to the two attacks specifically analyzed in Section 4, the main cause of the security vulnerabilities is the linear operation for those specified blocks at the process of proof generating. Hence, the original data file can be easily derived by any external attacker and malicious cloud server if he/she collects enough linear combinations of the selected blocks. To remedy the flaws left by Li et al.'s scheme, we make use of random masking technique to eliminate the linear relationship between the data blocks and integrity proofs in this paper, and the details are illustrated in Fig. 5.

Our new proposed scheme also consists of four parts: (1) The $KeyGen(1^{\lambda})$ algorithm is executed by the client to get a public key and a private key with the security parameter λ. (2) In the $SignGen(pk, sk, F)$ algorithm, the user generates a signature set Φ for corresponding data file F. (3) The cloud server implements the $ProofGen(F, \Phi, chal)$ algorithm after receiving the auditor's request $chal$, and then takes the result as integrity proof sending to the auditor. (4) Algorithm $ProofCheck(pk, chal, pf)$ is executed by the auditor to verify the correctness of that proof derived from the public cloud, and it returns accept or reject. The first two algorithms are the same with Li et al.'s scheme, and we only give an appropriate revision on the last two algorithms to

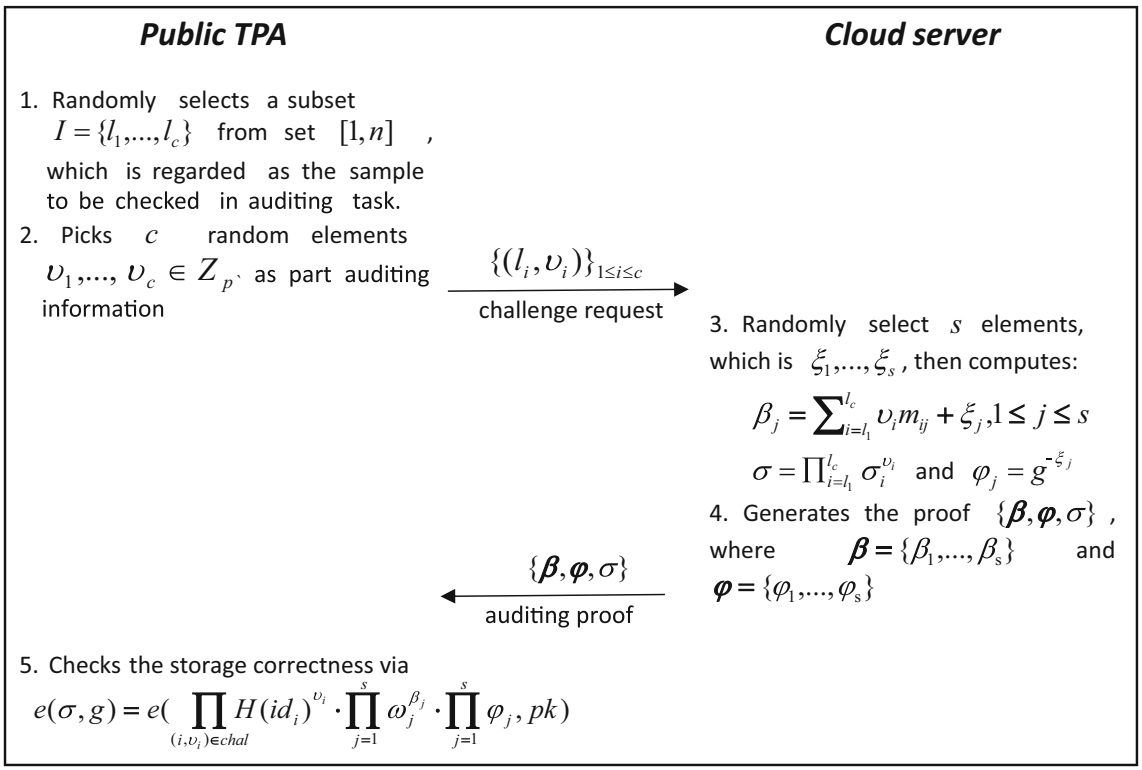

Fig. 5 The auditing process of the new scheme

achieve the data privacy preserving and get better security performance in the new proposed scheme.

5.1 $KeyGen(1^{\lambda}) \longrightarrow (sk, pk)$

This is a *setup* phase, where the user randomly selects a key $x \in Z_p$ as private key, and regards g^x as public key.

5.2 $SignGen(pk, sk, F) \longrightarrow \Phi$

It is first for the user to divide the data file F into n blocks, namely $F = \{m_i\}_{1 \le i \le n}$. In order to reduce the computation and storage cost while generating a signature set [26], the user furtherly divide each block into $s \geq 1$ sectors, where $m_i = \{m_{ij} \in Z_p^*\}_{1 \le j \le s}$. Then, the user randomly chooses $s \geq 1$ numbers, $\alpha_1, ..., \alpha_s \in Z_p$, to compute $\omega_j = g^{\alpha_j}$. To cut down the computation cost, the user translates $\prod_{j=1}^{s} \omega_j^{m_{ij}}$ into $g^{\sum_{j=1}^{s} \alpha_j \cdot m_{ij}}$, and generates the signature for each file block m_i, $1 \le i \le n$, which is shown as follows:

$$\sigma_i = (H(id_i) \cdot g^{\sum_{j=1}^{s} \alpha_j \cdot m_{ij}})^{sk}, 1 \le i \le n \tag{7}$$

Finally, the file F and the set of signature $\Phi = \{\sigma_i\}_{1 \le i \le n}$ are sent to the cloud without storage in local memory by the data owner. Besides, it is necessary to deliver the message $\{filename, n, \{\omega_i\}_{1 \le i \le n}\}$ to the TPA for later auditing process.

5.3 $ProofGen(F, \Phi, chal) \longrightarrow pf$

Before issuing the verification, an auditing challenge should be generated by TPA as following steps:

- Randomly chooses a c-elements subset L of set $[1, n]$, where $L = \{l_1, \ldots, l_c\}$.
- Randomly picks a c-element vector $\upsilon = \{\upsilon_1, \ldots, \upsilon_c\}_{\upsilon_i \in Z_p}$ corresponding to $\{l_i\}_{1 \le i \le c}$.
- Sends the challenge request $chal = \{(l_i, \upsilon_i)\}_{l_i \in L}$ to the public cloud server.

After receiving the challenge request $chal$ from TPA, the cloud generates a relevant proof to prove the integrity of outsourced file. To avoid the two attacks shown in Section 4, choose s random elements $\xi_1, ..., \xi_s \in Z_p$ to compute the proof as follows:

$$\begin{aligned} \beta_j &= \sum_{i=1}^{c} \upsilon_i m_{l_i j} + \xi_j, 1 \le j \le s \\ \varphi_j &= g^{-\xi_j}, 1 \le j \le s \\ \sigma &= \prod_{i=1}^{c} \sigma_{l_i}^{\upsilon_i} \end{aligned} \tag{8}$$

So, the cloud server returns $\{\sigma, \beta, \varphi\}$ as its auditing proof to TPA, where $\beta = \{\beta_1, \beta_2, \ldots, \beta_s\}$ and $\varphi = \{\varphi_1, \varphi_2, \ldots, \varphi_s\}$.

5.4 $ProofCheck(pk, chal, pf) \longrightarrow (true, false)$

Once the public auditor receives the proof $\{\sigma, \beta, \varphi\}$, he/she checks its correctness on the basis of specified request message $chal$ and the public key pk by validating the following equation:

$$e(\sigma, g) \stackrel{?}{=} e(\prod_{(l_i, \upsilon_i) \in chal} H(id_i)^{\upsilon_i} \cdot \prod_{j=1}^{s} \omega_j^{\beta_j} \cdot \prod_{j=1}^{s} \varphi_j, pk) \tag{9}$$

If this equation is tenable, the result of $ProofCheck(pk, chal, pf)$ will be 1, that is, the public auditor believes that the data file F maintains its integrity in public cloud storage. Otherwise, the auditor can assert that one or more data blocks in outsourced file get destroyed in cloud when the output is 0.

Assume that the cloud is honest and it executes the procedures proposed in our scheme with the client in a right manner, it must be able to pass the verifier's auditing. Then, the validity of this proof is verified as below:

$$\begin{aligned} e(\sigma, g) &= e(\prod_{i=l_1}^{l_c} \sigma_i^{\upsilon_i}, g) \\ &= e(\prod_{i=l_1}^{l_c} (H(id_i) \cdot g^{\sum_{j=1}^{s} \alpha_j m_{ij}})^{sk \cdot \upsilon_i}, g) \\ &= e(\prod_{i=l_1}^{l_c} H(id_i)^{\upsilon_i} \cdot \prod_{i=l_1}^{l_c} g^{\upsilon_i \cdot \sum_{j=1}^{s} \alpha_j m_{ij}}, g^{sk}) \\ &= e(\prod_{i=l_1}^{l_c} H(id_i)^{\upsilon_i} \cdot \prod_{j=1}^{s} (g^{\alpha_j \beta_j} \cdot g^{-\xi_j}), pk) \\ &= e(\prod_{(l_i, \upsilon_i) \in chal} H(id_i)^{\upsilon_i} \cdot \prod_{j=1}^{s} \omega_j^{\beta_j} \cdot \prod_{j=1}^{s} \varphi_j, pk) \end{aligned} \tag{10}$$

6 Security analysis and comparisons

In this section, we make an analysis to the security of proposed provable data possession scheme for cloud storage. We first prove that the scheme can resist the data privacy attack and proof forgery attack. Then, we compare the security of our scheme and other most recent schemes.

6.1 Security analysis

On the basis of the system model and the abilities of adversaries, we demonstrate that the signatures of outsourced

data blocks cannot be forged. Under the premise, we further prove that the proposed scheme is secure against proof forgery attack. Then, we demonstrate that the proposed scheme can resist the data privacy attack; any external attacker cannot reveal the content of outsourced data from auditing message.

Theorem 1 It is impossible for a dishonest cloud or an external attacker to forge a signature of the data block as long as the CDH problem is computationally infeasible to solve.

Proof If the cloud server or other external attacker can create a correct signature for the data block with no possession of the owner's private key and entire file, it is highly possible to tamper the original data and pass the auditor's verification. Here, we prove that our scheme is secure in signature forgery attack. □

We first give an assumption that an adversary $\mathscr{A}$ which can choose message and identity adaptively is competent to break this scheme with the probability of ε in time t after executing at most q_H hash queries and q_s sign queries and requesting q_k key queries. Then, there exists a simulator $\mathscr{B}$ that can solve the CDH problem in G with

$$t' \leqslant t + q_H(T_G + T_p) + q_s T_G$$
$$\varepsilon' \geqslant \frac{\varepsilon}{q_H q_k} \tag{11}$$

where T_G denotes the time each exponentiation takes on G, and T_p is the time one incorporation takes on Z_p.

Given (g, g^a, g^b) as a CDH problem example, the simulator $\mathscr{B}$ implement a signature forgery attack for adversary as follows:

Setup. As A asks for the creation of system users, $\mathscr{B}$ sets the security parameter (G, g, p) and guess which one $\mathscr{A}$ will plan to forge. Without loss of generality, we set $pk_t = g^a$ as the target public key. For all other public keys, we randomly choose a element $\tau_i \in Z_p$ to set the public key pk_i as g^{τ_i} for each $i \neq t$.

Queries. There only exist two kinds of oracle queries Q_{hash}, Q_{sign} that $\mathscr{A}$ can execute and $\mathscr{B}$ is supposed to give the corresponding valid answers.

- Q_{hash} : $\mathscr{B}$ maintains a list L_H to look up the Q_{hash} records. For each input (id_i, m_i), $\mathscr{B}$ checks whether the entry is in L_H, if so, returns the corresponding value to A. Otherwise, guesses if the queries (id_i, m_i) is the target id^* or the target block m^* that A used in its forgery. If $id_i = id^*$, $m_i = m^*$, let $H(id_i)g^{\sum_{j=1}^{s} \alpha_j m_{ij}} = g^b$. If not, randomly chooses a $\gamma_i \in Z_p$, and returns g^{γ_i} to A, then inserts this record into list L_H.
- Q_{sign} : $\mathscr{B}$ maintains a four-tuple $(pk_j, id_i, m_i, \sigma_i)$ consisting of the list L_{sign}. For each input (pk_j, id_i, m_i), if $j \neq t, id_i = id^*, m_i = m^*$, $\mathscr{B}$ aborts. Otherwise, if $j \neq t$, $\mathscr{B}$ returns $\sigma_i = (H(id_i)g^{\sum_{j=1}^{s} \alpha_j m_{ij}})^{\tau_j}$ via hash queries (assuming that A has executed the hash query for simplicity); else if $j = t$ but $id_i \neq id^*$ or $m_i \neq m^*$, $\mathscr{B}$ returns $\sigma_i = (g^{\gamma_i})^a = (g^a)^{\gamma_i}$.
- $Forgery$: Finally the adversary $\mathscr{A}$ generates a forgery $(pk_j, id^*, m^*, \sigma^*)$ after adaptive queries. If $j \neq t$, $\mathscr{B}$ fails to guess the target user, the game aborts. If $ProofCheck(pk_j, m^*, \sigma^*) \neq 1$, or $ProofCheck(pk_j, m^*, \sigma^*) = 1$ but a result of Q_{sign}, it aborts, because $\mathscr{B}$ cannot forge a valid signature by itself. Otherwise, A succeeds to create a signature of forgery with no information from above oracle queries, which also means that $\mathscr{B}$ is able to find a $\sigma = (H(id^*)g^{\sum_{j=1}^{s} \alpha_j m_j^*})^a = (g^b)^a = g^{ab}$ as a solution to the proposed CDH problem.

Since the public key queries time is q_k, $\mathscr{B}$ can guess the target user with a probability of $1/q_k$, and the probability that $\mathscr{B}$ accurately guess the id^* and target block m^* is $1/q_H$. Thus, $\mathscr{B}$ will solve the CDH problem with the probability of $\varepsilon/q_k q_H$ if $\mathscr{A}$ can forge a valid signature with probability ε. Throughout the process, each hash query requests an exponentiation on G and an extra incorporations on Z_p, and each sign query requires an exponentiation on G, thus the entire performing time is $t + q_H(T_G + T_p) + q_s T_G$.

According to the analysis above, the simulator $\mathscr{B}$ can solve the CDH problem with a non-negligible probability in polynomial time which is contradictory to the CDH assumption. Therefore, hardly can the signature of forgery be generated in this scheme.

Theorem 2 For the cloud server, the auditing proof is unforgeable under our improved scheme as long as the DL problem is computationally infeasible to solve.

Proof By reference to the security game defined in [31] and [35], we can demonstrate that if the cloud server succeeds to win the game, named Game1, via generating a fake auditing proof without correct outsourced data, then it means that we can solve the DL problem on group G with non-negligible probability in a polynomial time. □

Game1: The data owner or the other public verifier sends an auditing request $chal = \{(l_i, \upsilon_i)\}_{l_i \in L}$ to the cloud server

and asks for the auditing proof $\{\sigma, \beta, \varphi\}$ of the correct outsourced data, which is sure to make the verification equation (9) hold. But the cloud server may return a proof of forgery $\{\sigma, \beta', \varphi\}$ when it loses or modifies the original data. where $\beta' = \{\beta'_1, ..., \beta'_s\}$ and $\beta'_i = \sum_{i=1}^{c} \upsilon_i m_{l_i j} + \xi_j, i \in [1, s]$. Let $\Delta\beta_i = \beta_i - \beta'_i$, it is obvious that at least one element in $\Delta\beta_i$ for $i \in [1, s]$ is nonzero; otherwise, the outsourced data is stored correctly. Only does the fake proof pass the verification executed by the public auditor that the cloud server can win $Game1$, or it fails.

Because the auditing proof $\{\sigma, \beta, \varphi\}$ is correct, we get

$$e(\sigma, g) = e(\prod_{(l_i, \upsilon_i) \in chal} H(id_i)^{\upsilon_i} \cdot \prod_{j=1}^{s} \omega_j^{\beta_j} \cdot \prod_{j=1}^{s} \varphi_j, pk) \quad (12)$$

Now, we assume that the incorrect proof generated by cloud succeeds to pass the verification; then, in the same way, we get

$$e(\sigma, g) = e(\prod_{(l_i, \upsilon_i) \in chal} H(id_i)^{\upsilon_i} \cdot \prod_{j=1}^{s} \omega_j^{\beta'_j} \cdot \prod_{j=1}^{s} \varphi_j, pk) \quad (13)$$

According to the properties of bilinear maps, we can learn from (12) and equation (13) that

$$\prod_{j=1}^{s} \omega_j^{\beta_j} = \prod_{j=1}^{s} \omega_j^{\beta'_j} \Rightarrow \prod_{j=1}^{s} \omega_j^{\Delta\beta_j} = 1 \quad (14)$$

Since $\omega_j = g^{\alpha_j}$, where g is a generator of the cyclical group G and $\alpha_j \in Z_p$, there exists $b \in Z_p$ that satisfies $y = x^b$, where x and y are the other two generators of the cyclical group G. Then, we can select two random elements $r_j, k_j \in Z_p$ to meet the equation $\omega_j = x^{r_j} y^{k_j}$. So we have

$$1 = \prod_{j=1}^{s} (x^{r_j} y^{k_j})^{\Delta\beta_j} = x^{\sum_{j=1}^{s} r_j \Delta\beta_j} \cdot y^{\sum_{j=1}^{s} k_j \Delta\beta_j} \quad (15)$$

Given $x, y = x^b \in G$, it is clear that we obtain the value of b which is a solution of the DL problem as long as $\sum_{j=1}^{s} k_j \Delta\beta_j$ is not equal to zero, showed as below:

$$y = x^{-\frac{\sum_{j=1}^{s} r_j \Delta\beta_j}{\sum_{j=1}^{s} k_j \Delta\beta_j}} \Rightarrow b = -\frac{\sum_{j=1}^{s} r_j \Delta\beta_j}{\sum_{j=1}^{s} k_j \Delta\beta_j} \quad (16)$$

As we have described in Game 1, at least one element in $\{\Delta\beta_j\}_{1 \leqslant j \leqslant s}$ is not equal to zero and the random element $k_j \in Z_p$. Therefore, the probability that $\sum_{j=1}^{s} k_j \Delta\beta_j = 0$ is at most $1/p$, then it denotes that we can solve the DL problem with a probability of not less than $1 - 1/p$. Because the element p is a large prime, the value of $1/p$ is negligible, as a result, the probability of $1 - 1/p$ is non-negligible, which is contradictory to the DL assumption. Therefore, this scheme is secure in terms of proof forgery attack.

Theorem 3 Our improved scheme is secure in data privacy preserving if the DL assumption is correct. Simply speaking, it is very difficult for the external attacker to recovery the data file via any auditing proof from the cloud server under the complexity of DL problem.

Proof As for data privacy preserving, we show that none of the elements in auditing proof $\{\beta, \varphi, \sigma\}$ can be used to recover the outsourced data by any internal or external attacker. □

First, we prove that the element β can avoid the data recovery attack showed in Section 4.2 and preserves the privacy of $\sum_{i=1}^{j} \upsilon_i m_{l_i j}$. The reason is that we break the linear features in proof generating phase by embedding an element ξ_j as $\beta_j = \sum_{i=1}^{j} \upsilon_i m_{l_i j} + \xi_j$, where ξ_j is randomly chosen by cloud and is blinded to the public auditor.

Second, even though the element φ is transparent to all people, it is still very hard to get the value of ξ_j due to the computational complexity of the DL problem and CDH problem.

At last, we prove that the element σ can guarantee the data privacy. Following the definition that

$$\begin{aligned}
\sigma &= \prod_{i=l_1}^{l_c} (H(id_i) g^{\sum_{j=1}^{s} \alpha_j m_{ij}})^{sk\upsilon_i} \\
&= \prod_{i=l_1}^{l_c} H(id_i)^{\upsilon_i sk} \cdot \prod_{i=l_1}^{l_c} (g^{\upsilon_i \sum_{j=1}^{s} \alpha_j m_{ij}})^{sk} \\
&= \prod_{i=l_1}^{l_c} H(id_i)^{\upsilon_i sk} \cdot g^{sk \sum_{j=1}^{s} (\alpha_j \sum_{i=l_1}^{l_c} \upsilon_i m_{ij})} \\
&= \prod_{i=l_1}^{l_c} H(id_i)^{\upsilon_i sk} \cdot pk^{\sum_{j=1}^{s} (\alpha_j \sum_{i=l_1}^{l_c} \upsilon_i m_{ij})}
\end{aligned} \quad (17)$$

From the equation above, we can learn that $pk^{\sum_{j=1}^{s} (\alpha_j \sum_{i=l_1}^{l_c} \upsilon_i m_{ij})}$ is masked by $\prod_{i=l_1}^{l_c} H(id_i)^{\upsilon_i sk}$, where the value is composed of $H(id_i)$, υ_i and sk. Although the value $H(id_i)$, υ_i and g^{sk} are public to the auditing verifier, it is still impossible to calculate their product due to the hardness of CDH problem. What is more, hardly can we get the information about $\sum_{i=1}^{j} \upsilon_i m_{l_i j}$ from $pk^{\sum_{j=1}^{s} (\alpha_j \sum_{i=l_1}^{l_c} \upsilon_i m_{ij})}$ owing to the intractability of DL problem. Thus, our scheme is secure in terms of keeping data privacy.

6.2 Security comparisons

We compare the security of the improved integrity auditing scheme with Li et al.'s scheme and another two recent schemes [31] and [36] for public cloud storage. Let SR_1,

Table 1 Security comparisons of our proposed scheme and related schemes

	Wang's scheme [31]	Li's scheme [35]	Yu's scheme [36]	Our proposed scheme
SR_1	✓	✓	✓	✓
SR_2	×	×	✓	✓
SR_3	×	×	×	✓

✓: The requirement is satisfied.

×: The requirement is not satisfied.

SR_2, SR_3 denote public verification, unforgeability of auditing proof and preservation of data privacy. The comparison details of these schemes are shown in Table 1.

According to Table 1, we can see that none of the three schemes (i.e.,Wang's scheme [31], Li's scheme [35], Yu's scheme [36]) can meet all the three basic security requirements (SR_1 to SR_3). Especially for Wang's scheme [31] and Li's scheme [35], neither can they ensure the preservation of data privacy nor satisfy the unforgeability of auditing proofs. Yu's scheme is secure against proof forgery attack, but it suffers from data privacy attack. It is only our proposed scheme can meet these three basic security requirements for public data auditing.

7 Performance analysis

In this section, we analyze the performance of our proposed scheme via evaluating the computation cost and communication cost. The result shows that the improved scheme can preserve the data privacy and provide a more secure public auditing mechanism by increasing a little computation and communication costs compared with Li et al.'s scheme.

Communication cost analysis For the improved scheme, it does not add extra communication overhead to the data owner, because the method of giving a signature to each data block is the same as in [35]. Therefore, we only analyze the communication overhead increased by the auditing challenges and corresponding auditing proofs, shown in Table 2. To make the matter simple, assume that c blocks are chosen to be verified during the challenge-response process. Then, the size of an auditing challenge $\{(l_i, \upsilon_i)\}_{1\le i\le c}$ is $c(|p|+|n|)$ bits, where $|p|$ denotes the length of element υ_i in Z_p, $|n|$ denotes the length of each data block index. For the corresponding auditing proof $\{\mu, \varphi, \sigma\}$, there are s elements in μ and φ, respectively, so the size of auditing proof is about $(2s+1)|p|$. Compared with Li et al.'s scheme, the extra communication overhead mainly comes from the transmission of vector φ, whose size is only $s|p|$. Inheriting the advantage of Li et al.'s scheme, the new proposed scheme also adopts the BLS short signatures, so the communication complexity is constant and asymptotically with $O(1)$.

Computation cost analysis we compare the computation cost between our proposed scheme and Li et al.'s scheme [35] in Table 3. For convenience, the notations used to present the execution time are defined below:

- *MG:* The time cost of a multiplication operation on group G.
- *Exp:* The time cost of an exponent operation in group G.
- *MZ:* The time cost of a multiplication operation in the filed Z_p.
- *Add:* The time cost of an addition operation in the filed Z_p.
- *H:* The time cost of one hash operation on group G.
- *BP:* The time cost of a pairing operation on group G.

As we can see, the initialization phase and algorithms $KeyGen$, $SignGen$ are completely same between [35] and our proposed scheme. Therefore, the entire computation overhead of these three phases is also the same as that in [35], which is $nMG + nH + 2nExp + nsMZ + nsAdd$. Since the difference between Li et al.'s scheme and our new proposed scheme is the $ProofGen$ and $ProofCheck$ algorithms, we mainly discuss the computation cost during the two processes.

According to the equation presented in Fig. 5, the computation cost for generating an auditing proof is around

Table 2 Comparisons of communication cost between our proposed scheme and Li's schemes [35]

Scheme	Auditing challenges	Auditing proofs	Complexity
Li's scheme [35]	$c(\|p\|+\|n\|)$ bits	$(s+1)\|p\|$ bits	$O(1)$
Our proposed scheme	$c(\|p\|+\|n\|)$ bits	$(2s+1)\|p\|$ bits	$O(1)$
Deviation	none	$s\|p\|$ bits	none

Table 3 Comparisons of computation cost between our proposed scheme and Li's scheme [35]

Scheme	Li's scheme [35]	Our proposed scheme
Data user	$nsMZ + nMG + nsAdd + 2nExp + nH$	$nsMZ + nMG + nsAdd + 2nExp + nH$
Cloud server	$csMZ + cMG + csAdd + cExp$	$csMZ + cMG + s(c+1)Add + (c+s)Exp$
Public auditor	$2BP + (c+s)Exp + (c+s+1)MG$	$2BP + (c+s)Exp + (c+2s+2)MG$

$csMZ + s(c+1)Add + (c+s)Exp + cMG$ for the cloud server; In the same way, the computation cost for the TPA to verify the validity of auditing proofs is about $2P + (c+s)Exp + (c+2s+2)MG$. Referring to Table 3, the extra computation cost of the new proposal is only $sAdd$ for cloud server and $(s+1)MG$ for the public auditor, but we enforce the security in a large extent when it is compared with [35].

Auditing performance analysis Next, we analyze the auditing performance of our proposed scheme. We carried out the experiments on a personal computer Dell (with an I5-4460S 2.90GHz processor, 4GB memory, and the Window 8 operating system) using the MIRACL library. If the probability of data corruption is 1%, the TPA should randomly choose only 460 sampled blocks to detect this misbehavior with probability greater that 99%. As the same with Li et al.'s scheme [35], we implemented the auditing operations with 300 blocks (detection probability of about 95%) and 460 blocks, respectively. Specifically, we assume that there are 100 sections in each block. Then, we repeated the experiments for 100 trials and compared with [35], which is shown in Fig. 6. Obviously, the auditing time increases with the increase of value c.

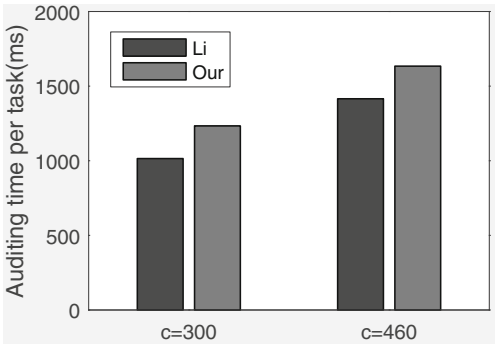

Fig. 6 The comparison of the proposed scheme and Li et al.'s scheme

8 Conclusion

To achieve the blueprint of smart city in big data age, it is crucial to ensure the security of outsourced data in the public cloud server. In the last several years, many public integrity auditing protocols have been proposed, but the openness of these audit models might bring some security and privacy risks. In this paper, we analyze two weaknesses of Li et al.'s scheme. we demonstrate that Li et al.'s scheme is vulnerable to the data privacy attack and proof forgery attack. Aimed at the two weaknesses, we propose a new provable data possession scheme. The improved scheme inherits the properties and advantages of Li et al.' scheme [35], such as batch auditing. And the detailed security analysis and performance evaluation demonstrate that it can achieve a desire public auditing and a novel data privacy preserving especially while incurring a little extra communication and computation cost. Note that the cryptanalysis and proposed method can solve the security weaknesses in [29, 33, 34, 36, 37] or other variants as well.For the future work, we will be devoted to design a more secure and practical auditing scheme with high efficiency.

Acknowledgments We thank the anonymous reviewers for the constructive comments which help improve the quality and presentation of this paper. The work of L. Wu was supported by the National Natural Science Foundation of China (Nos.61272112, 61472287). The work of D. He was supported in part by the National Natural Science Foundation of China (Nos. 61572379, 61501333, U1536204), in part by the National High-Tech Research and Development Program of China (863 Program) (No. 2015AA016004), in part by the open fund of Guangxi Key Laboratory of Cryptography and Information Security, and in part by the Natural Science Foundation of Hubei Province of China (No. 2015CFB257).

References

1. Neirotti P, De Marco A, Cagliano AC, Mangano G, Scorrano F (2014) Current trends in smart city initiatives Somestylised facts. Cities 38:25–36

2. Li Y, Dai W, Ming Z, Qiu M (2016) Privacy protection for preventing data over-collection in smart city. IEEE Trans Comput 65(5):1339–1350
3. Yamamoto S, Matsumoto S, Nakamura M (2012) Using cloud technologies for large-scale house data in smart city. In: 2012 IEEE 4th international conference on cloud computing technology and science (CloudCom). IEEE, pp 141–148
4. Dey S, Chakraborty A, Naskar S, Misra P (2012) Smart city surveillance: Leveraging benefits of cloud data stores. In: 2012 IEEE 37th conference on local computer networks workshops (LCN Workshops). IEEE, pp 868–876
5. Ren K, Wang C, Wang Q (2012) Security challenges for the public cloud. IEEE Internet Comput
6. Song D, Shi E, Fischer I, Shankar U (2012) Cloud data protection for the masses. Computer
7. Behl A, Behl K (2012) An analysis of cloud computing security issues. In: 2012 world congress on information and communication technologies (WICT). IEEE, pp 109–114
8. Chen D, Zhao H (2012) Data security and privacy protection issues in cloud computing. In: 2012 data international conference on computer science and electronics engineering (ICCSEE), volume 1. IEEE, pp 647–651
9. Tate SR, Vishwanathan R, Everhart L (2013) Multi-user dynamic proofs of data possession using trusted hardware. In: Proceedings of the 3rd ACM conference on data and application security and privacy. ACM, pp 353–364
10. Yuan J, Yu S (2013) Proofs of retrievability with public verifiability and constant communication cost in cloud. In: Proceedings of the 2013 international workshop on security in cloud computing. ACM, pp 19–26
11. Wang Q, Wang C, Ren K, Lou W, Li J (2011) Enabling public auditability and data dynamics for storage security in cloud computing. IEEE Trans Parallel Distrib Syst 22(5):847–859
12. Shuang T, Lin T, Li X, Yan J (2014) An efficient method for checking the integrity of data in the cloud. Commun China 11(9):68–81
13. Ren Y, Shen J, Wang J, Han J, Lee S (2015) Mutual verifiable provable data auditing in public cloud storage. J Internet Technol 16(2):317–323
14. Fu Z, Sun X, Qi L, Zhou L, Shu J (2015) Achieving efficient cloud search services: multi-keyword ranked search over encrypted cloud data supporting parallel computing. IEICE Trans Commun 98(1):190–200
15. Xia Z, Wang X, Sun X, Wang Q (2016) A secure and dynamic multi-keyword ranked search scheme over encrypted cloud data. IEEE IEEE Trans Parallel Distrib Syst 27(2):340–352
16. Fu Z, Ren K, Shu J, Sun X, Huang F (2016) Enabling personalized search over encrypted outsourced data with efficiency improvement. IEEE Trans Parallel Distrib Syst 27(9):2546–2559
17. Guo P, Wang J, Geng X, Chang SK, Kim J-U (2014) A variable threshold-value authentication architecture for wireless mesh networks. J Internet Technol 15(6):929–935
18. Shen J, Tan H, Wang J, Wang J, Lee S (2015) A novel routing protocol providing good transmission reliability in underwater sensor networks. J Internet Technol 16(1):171–178
19. He D, Zeadally S, Kumar N, Lee JH (2016) Anonymous authentication for wireless body area networks with provable security. IEEE Syst J. doi:10.1109/JSYST.2016.2544805
20. He D, Huang B, Chen J (2013) New certificateless short signature scheme. IET Inf Secur 7(7):113–117
21. Hwang JY, Chen L, Cho HS, Nyang DH (2015) Short dynamic group signature scheme supporting controllable linkability. IEEE Trans Inf Forensics Secur 10(6):1109–1124
22. He D, Kumar N, Choo K-KR, Wu W (2016) Efficient hierarchical identity-based signature with batch verification for automatic dependent surveillance-broadcast system. IEEE Trans Inf Forensics Secur. doi:10.1109/TIFS.2016.2622682
23. Ateniese G, Burns R, Curtmola R, Herring J, Kissner L, Peterson Z, Song D (2007) Provable data possession at untrusted stores. In: Proceedings of the 14th ACM conference on computer and communications security. ACM, pp 598–609
24. Juels A, Kaliski Jr B S (2007) Pors: Proofs of retrievability for large files. In: Proceedings of the 14th ACM conference on computer and communications security. ACM, pp 584–597
25. Shacham H (2008) Compact proofs of retrievability. Trans ASIACRYPT (2008)
26. Shacham H, Waters B (2013) Compact proofs of retrievability. J Cryptol 26(3):442–483
27. He D, Zeadally S, Wu L (2015) Certificateless public auditing scheme for cloud-assisted wireless body area networks. IEEE Syst J. doi:10.1109/JSYST.2015.2428620
28. Boneh D, Lynn B, Shacham H (2001) Short signatures from the weil pairing. In: Advances in cryptology—ASIACRYPT 2001. Springer, pp 514–532
29. Wang B, Li H, Li M (2013) Privacy-preserving public auditing for shared cloud data supporting group dynamics. In: 2013 IEEE international conference on communications (ICC). IEEE, pp 1946–1950
30. Wang B, Li B, Li H, Li F (2013) Certificateless public auditing for data integrity in the cloud. In: 2013 IEEE conference on communications and network security (CNS). IEEE, pp 136–144
31. Wang B, Li B, Li H (2015) Panda: public auditing for shared data with efficient user revocation in the cloud. IEEE Trans Serv Comput 8(1):92–106
32. Yang T, Yu B, Wang H, Li J, Lv Z (2015) Cryptanalysis and improvement of panda-public auditing for shared data in cloud and internet of things. Multimedia Tools and Applications
33. Wang B, Chow SSM, Li M, Li H (2013) Storing shared data on the cloud via security-mediator. In: 2013 IEEE 33rd international conference on distributed computing systems (ICDCS). IEEE, pp 124–133
34. Wang B, Li B, Li H (2013) Public auditing for shared data with efficient user revocation in the cloud. In: IEEE INFOCOM. IEEE, pp 2904–2912
35. Li A, Tan S, Jia Y (2016) A method for achieving provable data integrity in cloud computing. J Supercomput
36. Yu Y, Xue L, Au MH, Susilo W, Ni J, Zhang Y, Vasilakos AV, Shen J (2016) Cloud data integrity checking with an identity-based auditing mechanism from rsa. Futur Gener Comput Syst 62:85–91
37. Tang C-M, Zhang X-J (2015) A new publicly verifiable data possession on remote storage. J Supercomput:1–15