Hindawi Publishing Corporation
International Journal of Distributed Sensor Networks
Volume 2015, Article ID 593759, 10 pages
http://dx.doi.org/10.1155/2015/593759

Research Article

Efficient Pairing-Free Privacy-Preserving Auditing Scheme for Cloud Storage in Distributed Sensor Networks

Xinpeng Zhang, Chunxiang Xu, and Xiaojun Zhang

School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu 611731, China

Correspondence should be addressed to Xinpeng Zhang; carriage1029@163.com

Received 21 November 2014; Accepted 19 January 2015

Academic Editor: Lu-An Tang

With the rapid growth of the distributed sensor networks, the distributed sensor network data security problems begin to attract the attention of people. The previous research of distributed sensor network security has focused on secure information in communication; however the research of secure data storage has been overlooked. As we know, cloud data storage and retrieval have become popular for efficient data management in distributed sensor networks; thus they can enjoy the on-demand high-quality cloud storage service. Meanwhile, it also introduces new security challenges. To tackle with these security challenges, many classic auditing schemes of cloud storage have been proposed. However, these schemes all need very expensive pairing computation, which is not suitable for sensor networks. In this paper, we propose an efficient pairing-free auditing scheme for data storage of distributed sensor networks. We exploit homomorphic message authentication codes (MACs) to reduce the space used to store the verification information. We also employ the random masking technique to make sure the TPA cannot recover the primitive data blocks of the sensor networks data manager. Experimental results show that our auditing scheme is more light-weight than previous auditing schemes and more practical in applied distributed sensor networks environments.

1. Introduction

Nowadays, distributed sensor networks have been rapidly applied in many practical environments in our social life [1, 2]. With distributed sensor networks being applied widely, the sensor network data managers often need to collect massive data and choose to be stored in the cloud server, while the security and privacy of sensor networks storage data become increasingly important [3, 4]. As we know, cloud computing is an alternative to conventional computing model since it can provide a flexible, resilient, and cost-effective infrastructure [5]. So it is suitable option to store the massive sensor network data on cloud server [6]. While cloud storage is an important service of cloud computing, which allows cloud users to move data from their local computing systems to the cloud, by data outsourcing, the cloud users can be relieved from the burden of local data storage and maintenance. Thus the cloud servers can concentrate on their core business issues and operate other business applications through the Internet, rather than incurring substantial hardware, software, and personnel costs involved in deploying and maintaining applications in-house.

Although the cloud storage service makes these advantages more appealing than ever before, it also introduces new security challenges towards user's outsourced data [7–9]. Firstly, the cloud users would worry their data could be misused or accessed by unauthorized users. Many researches have been done on this security issue of data hosting [10–12]. Secondly, the cloud users would worry their data could be lost in the cloud. This is because data loss could happen in any infrastructure, no matter what high degree of reliable measures the cloud service providers would take [13, 14]. Sometimes, the cloud service providers may be dishonest and they may discard the data which have not been accessed or rarely accessed to save the storage space or keep fewer replicas than promised. Moreover, the cloud service providers may choose to hide data loss and claim that the data are still correctly stored in the cloud. Consequently, the cloud users need to be convinced that their data are correctly stored in the cloud.

As the cloud users no longer physically possess the storage of their data, traditional cryptographic primitives for the purpose of data security protection can not be

directly adopted. Thus how to efficiently verify the integrity of outsourced cloud data without the local copy of data files becomes a big challenge for data storage security in cloud computing. Checking on retrieval is a common method for checking the data integrity, which means cloud users check the data integrity when accessing their data. This method has been used in peer-to-peer storage systems [15], network file systems [16, 17], web-service object stores [18], and database systems [19]. However, checking on retrieval is not sufficient to check the integrity for all the data stored in the cloud. There is usually a large amount of data stored in the cloud; the ability to audit the correctness of the data in a cloud environment can be formidable and expensive for the cloud users [20, 21]. Therefore, in order to save the communication resources as well as the online burden potentially brought by the periodic storage correctness verification, cloud users can delegate a third party (TPA) to perform security auditing tasks as it is not economically feasible for them to handle it by themselves. Meanwhile, the cloud users also hope to keep their data private from the TPA and the cloud server.

1.1. Related Work. Until now, a number of auditing schemes have been proposed in the context of ensuring remotely stored data integrity without the knowledge of the entire data with different requirements [20, 22–24]. However, these schemes need the expensive pairing computation; it is a burden for the sensor network. And most of these schemes [20, 22, 24] do not consider the privacy protection of user's data. Indeed, the user's data may be revealed to some curious adversaries. This shortcoming will greatly affect the security of these schemes in cloud computing. In the view of protecting the data privacy, the users can rely on the TPA for the storage security of their data, and they also do not want this auditing process to introduce new vulnerabilities of unauthorized information leakage toward their data security [25]. The unauthorized data leakage still remains possible due to the potential exposure of decryption keys. In 2013, Wang et al.'s [26] has presented a privacy-preserving public auditing scheme for cloud storage; it resorts to the homomorphic authenticator technique and random masking technique to achieve privacy-preserving public auditing and utilizes the technique of bilinear aggregate signature to realize batch auditing. However it also acquires very expensive pairing computation, which is time-consuming. Therefore, how to design an efficient privacy-preserving auditing scheme for cloud storage in distributed sensor networks, especially without needing the expensive pairing computation, is the important work we are going to do in this paper.

1.2. Our Contribution. Motivated by the above, in this paper, we propose an efficient pairing-free privacy-preserving auditing scheme for cloud storage in distributed sensor networks. In particular, we utilize the modified Schnorr signature to construct homomorphic authenticator so that the TPA can verify the integrity of the data without retrieving the entire data. Additionally, we exploit homomorphic MACs [27] to reduce the space used to store the verification information. As a necessary tradeoff, we allow the TPA to share a private key pair with the DSN data manager, which we refer to as authorized auditing. Due to the function of the random masking, even if the authorized TPA possesses the private key pair, the TPA cannot recover the primitive data blocks of the DSN data manager. As the individual auditing of these growing auditing tasks can be tedious, we extend our basic scheme to support batch auditing for multiuser, which can thus enable the TPA to efficiently perform multiple auditing tasks in a batch manner simultaneously. Furthermore, compared with the previous classic auditing scheme [26], our experimental results show that our auditing scheme is more light-weight, and this is mainly because our auditing scheme does not need the expensive pairing operations, which can satisfy the requirement of the sensor network.

1.3. Organization. The rest of this paper is organized as follows. We introduce the preliminaries of our work in Section 2. We give the formal pairing-free privacy-preserving auditing scheme for cloud storage with distributed sensor networks in Section 3. We give the analysis of the proposed auditing scheme in Section 4. We make a performance comparison in Section 5. We make a conclusion in Section 6.

2. Preliminaries

2.1. The Cloud Data Storage Model in Sensor Network. We exemplify the security needs in data storage with a distributed sensor networks application scenario. Here, for simplicity, after collecting the data by the sink node, we assume that we assume that a DSN (distributed sensor network) data storage manager processes and transfers sensor networks data to the cloud sever. Since DSN data storage manager does not own additional computing resources, it only takes advantage of the limited computing capacity of the sink node to finish the secure DSN data storage. For the part of DSN, it can be considered as a data storage manager, while, for the part of the CSP (cloud service provider), it can be considered as a special cloud user. In our distributed sensor networks application scenario, we suppose Pob is a DSN data owner; his business is that to collect sensor network data which are processed to supply various service to clients. Since he does not have enough money to buy devices and hire professionals, he wishes to turn to CSP and outsource his data to CSP. However, he will worry about the following questions: (1) he cannot physically control the data, and CSP may repudiate that the data are lost, which makes him verify the integrity of data at any time; (2) CSP is honest and curious, Pob wants to guarantee his data confidentiality, and he must assure his storage mode has the function of privacy-preserving; (3) Pob's main work is responsible for sensor network, and he needs an efficient audit scheme to complete this task.

As illustrated in Figure 1, sensor nodes collect data from the target setting and send them to sink node; Pob is a distributed sensor networks data owner; he can assign a DSN data storage manager to sign and encrypt data, then outsource data and tags to CSP, and delete local data simultaneously. If the DSN data storage manager wants to verify data integrity stored in the cloud server, he makes a request for TPA; TPA verifies tags after it receives the requests.

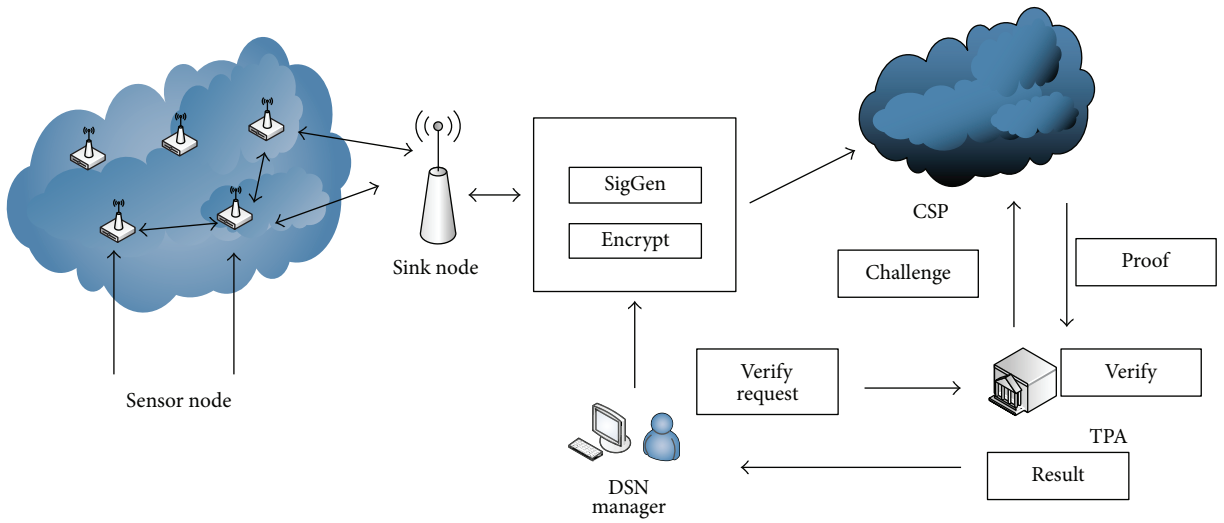

Figure 1: DSN data storage in the CSP.

If it is true, it generates a challenge message. After receiving the challenge message request, CSP supplies the response proof to TPA; TPA verifies the response proof message and returns the verified result to the DSN data storage manager. Finally, the DSN data storage manager submits the auditing result to Pob.

The sensor network (DSN) data manager can rely on the cloud server for cloud data storage and maintenance. They may also dynamically interact with the cloud server to access and update their stored data for various application purposes. The DSN data manager may resort to the TPA for ensuring the storage security of their outsourced data, while hoping to keep their data private from the TPA. We consider that a semitrusted cloud server exists. Namely, in most of time it behaves properly and does not deviate from the prescribed protocol execution. However, during providing the cloud data storage based services, for the benefits the cloud server might neglect to keep or deliberately delete rarely accessed data files which belong to the DSN data manager. Moreover, the cloud server may decide to hide the data corruptions caused by server hacks or failures to maintain reputation. We assume that the TPA, who is in the business of auditing, is reliable and independent and thus has no incentive to collude with either the cloud server or the DSN data manager during the auditing process. The TPA should be able to efficiently audit the cloud data storage without local copy of data and without bringing in additional online burden to the DSN data manager. However, any possible leakage of DSN manager's outsourced data towards the TPA through the auditing protocol should be prohibited.

2.2. Design Goals. To enable privacy-preserving auditing for cloud data storage under the aforementioned model, our auditing scheme should achieve the following security and performance guarantee:

(i) public auditability: to allow the TPA to verify the correctness of the cloud data on demand without retrieving a copy of the whole data or introducing additional online burden to the DSN data manager,

(ii) storage correctness: to ensure that there is not a cheating cloud server that can pass the auditing from the TPA without indeed storing DSN data manager data intact,

(iii) privacy-preserving: to ensure that there exists no way for the TPA to derive DSN data managers' data content from the information collected during the auditing process,

(iv) batch auditing: to enable the TPA with secure and efficient auditing capability to cope with multiple auditing delegations from possibly large number of different DSN data managers simultaneously,

(v) lightweight: to allow the TPA to perform auditing with minimum communication and computation overhead.

2.3. Cryptographic Definition

Definition 1. Discrete Logarithm problem states that, given a multiplicative cyclic group of order of p and $g, d \in G$ as input, compute $\eta \in Z_p$ such that $d = g^{\eta}$.

The Discrete Logarithm assumption holds in G if no polynomial time algorithm has a nonnegligible probability in solving the Discrete Logarithm problem, which means it is computationally infeasible to solve the Discrete Logarithm problem in G.

Now we introduce homomorphic MAC, described in [27].

Table 1: The privacy-preserving auditing scheme.

TPA		The cloud server
(1) Retrieve file *tag*, verify its signature, and quit if it failed.		
(2) Generate a challenge message $chal = \{(j, \nu_j)\}_{j\in\mathcal{J}}$.	$\overrightarrow{chal = \{(j, \nu_j)\}_{j\in\mathcal{J}}}$	(3) Compute $r = \prod_{j\in\mathcal{J}} r_j^{\nu_j r_j} \bmod p$ and compute $s = \sum_{j\in\mathcal{J}} \nu_j s_j \bmod q$, $\mu'_\ell = \sum_{j\in\mathcal{J}} \nu_j m'_{j,\ell} \in Z_p$, where $\ell \in \{1, \ldots, k\}$.
		(4) Choose a random element $\eta_\ell \leftarrow Z_q$ and calculate $W_\ell = y^{\eta_\ell}$.
		(5) Compute $\mu = (\mu_1, \ldots, \mu_k)$, where $\mu_\ell = \mu'_\ell + \eta_\ell h(W_\ell)$ and $W = (W_1, \ldots, W_k)$.
	$\overleftarrow{\{\mu, r, s, W, \{\mathrm{id}_j\}_{j\in\mathcal{J}}\}}$	
(6) Generate $\rho = (\rho_1, \ldots, \rho_k) \leftarrow \mathrm{PRG}(\mathrm{sk}_{\mathrm{prg}}) \in Z_q^k$ and $\rho_j \leftarrow \mathrm{PRF}(\mathrm{sk}_{\mathrm{prf}}, \mathrm{id}_j) \in Z_q$.		
(7) Compute $\lambda_1 = \sum_{\ell=1}^k \rho_\ell \mu_\ell + \sum_{j\in\mathcal{J}} \nu_j \omega_j \in Z_q$, $\lambda_2 = \sum_{\ell=1}^k \sum_{j\in\mathcal{J}} \rho_\ell \nu_j f_\tau(\ell, \mathrm{id}_j) \in Z_q$, and $h(W_\ell)$, where $\ell \in \{1, \ldots, k\}$, and then verify $\{\mu, r, s, W, \{\mathrm{id}_j\}_{j\in\mathcal{J}}\}$ via the verification equation.		

Definition 2 (Homomorphic MAC). Given a data block $m_j = (m_{j,1}, \ldots, m_{j,k}) \in Z_q^k$, the homomorphic MAC of this data block can be computed as $t_j = \sum_{\ell=1}^k \rho_\ell m_{j,\ell} + \omega_j \in Z_q$, where $\rho = (\rho_1, \ldots, \rho_k)$ is generated by a pseudorandom generator and a secret key $\mathrm{sk}_{\mathrm{prg}}$ and ω_j is calculated by a pseudorandom function and a secret key $\mathrm{sk}_{\mathrm{prf}}$.

We know that, given t_1 and t_2, an intermediate node can compute a valid MAC of a new date block $m' = m_1 + m_2$ by calculating $t' = t_1 + t_2$ without knowing the secret key pair $(\mathrm{sk}_{\mathrm{prg}}, \mathrm{sk}_{\mathrm{prf}})$.

3. Pairing-Free Privacy-Preserving Auditing Scheme for Cloud Storage in Distributed Sensor Networks

In this section, we propose our privacy-preserving authorized auditing scheme for cloud storage in distributed sensor networks, and our scheme does not need pairing computation and thus can reduce much computation cost. The privacy-preserving auditing scheme is illustrated in Table 1. Here, we need to define a semitrusted TPA, who is only responsible for auditing the integrity of data blocks honestly; however, it is curious and may try to reveal the DSN managers' primitive data blocks based on verification information. Our scheme consists of the following four algorithms. They are Setup, SigGen, ProofGen, and ProofVerify, respectively.

Setup. The initial system chooses two large prime numbers p and q, satisfying that q is a prime factor of $p - 1$. Choose an integer g, such that $g^q \equiv 1 \bmod p$; g is a generator of multiplicative cyclic group of order q; denote it by G. Data file M is divided into n blocks, and each data block is further divided into k elements of Z_q. Therefore, M can be presented as $M = (m_1, m_2, \ldots, m_n) \in Z_q^{n\times k}$; each $m_j = (m_{j,1}, m_{j,2}, \ldots, m_{j,k}) \in Z_q^k$, $1 \le j \le n$. The system sets a pseudorandom generator $\mathrm{PRG} : \mathcal{K}_{\mathrm{prg}} \to Z_q^k$ and a pseudorandom function $\mathrm{PRF} : \mathcal{K}_{\mathrm{prf}} \times \mathcal{J} \to Z_q$, where $\mathcal{K}_{\mathrm{prg}}$ and $\mathcal{K}_{\mathrm{prf}}$ denote the set of secret keys for PRG and PRF, respectively, and $\mathcal{J}$ denotes the set of all identities of each data block in data file M. Then, the DSN data manager selects $x \leftarrow Z_q$ randomly, and $x \neq 0$ computes $y = g^x \bmod p$. Meanwhile, the DSN data manager also randomly computes a secret key pair $\mathrm{skp} = (\mathrm{sk}_{\mathrm{prg}}, \mathrm{sk}_{\mathrm{prf}})$, where $\mathrm{sk}_{\mathrm{prg}} \in \mathcal{K}_{\mathrm{prg}}$ and $\mathrm{sk}_{\mathrm{prf}} \in \mathcal{K}_{\mathrm{prf}}$. The system sets a lightweight symmetry encryption algorithm f, with its private key being τ. The system also sets a secure hash function $h : G \to Z_q$. In particular, to generate the data block tag, the DSN data manager chooses a random signing key pair (spk, ssk). Thus the public parameters are $\mathrm{pk} = \{G, g, y, \mathrm{spk}\}$, and the private parameters are $\mathrm{sk} = \{x, \tau, \mathrm{ssk}\}$.

SigGen. Given a data block $m_j = (m_{j,1}, \ldots, m_{j,k})$, this data block's identifier $\mathrm{id}_j \in \mathcal{J}$. To ensure the integrity of unique data block identity, the DSN data manager computes $\mathrm{tag}_j = \mathrm{id}_j \parallel \mathrm{SSig}_{\mathrm{ssk}}(\mathrm{id}_j)$ as the data block tag for m_j. The DSN data manager computes $\rho = (\rho_1, \ldots, \rho_k) \leftarrow \mathrm{PRG}(\mathrm{sk}_{\mathrm{prg}}) \in Z_q^k$ and $\omega_j \leftarrow \mathrm{PRF}(\mathrm{sk}_{\mathrm{prf}}, \mathrm{id}_j) \in Z_q$. Then the DSN data manager calculates the homomorphic MAC of data block $m_j = (m_{j,1}, \ldots, m_{j,k})$ as $t_j = \sum_{\ell=1}^k \rho_\ell m_{j,\ell} + \omega_j \in Z_q$. The DSN data manager begins to compute the signature of t_j as follows:

(1) choose $k_j \leftarrow Z_q$ and compute $r_j \equiv g^{k_j} \bmod p$ and $r'_j \equiv r_j \bmod q$;

(2) $s_j = (r'_j k_j + t_j x) \bmod q$;

(3) output $\sigma_j = (r_j, s_j)$ as the signature of t_j.

Denote the set of signatures by $\Phi = \{\sigma_j\}_{1\le j\le n}$. Meanwhile, to guarantee the confidentiality of the data file, the DSN data manager employs the lightweight symmetry encryption

algorithm f to encrypt each data block $m_j = (m_{j,1}, \ldots, m_{j,k})$ as $m'_j = (m_{j,1} + f_\tau(1, \mathrm{id}_j), \ldots, m_{j,k} + f_\tau(k, \mathrm{id}_j))$ under the symmetry private key τ. Thus, the data file $M = (m_1, \ldots, m_n)$ is encrypted to be $M' = (m'_1, \ldots, m'_n)$. Finally, the DSN data manager sends $\{M', \mathrm{tag}_{1\le j\le n}, \Phi\}$ to the cloud server and deletes them from local storage.

ProofGen. In this phase, for each data block m_j, the TPA first retrieves the data block tag, verifies the signature $\mathrm{SSig}_{\mathrm{ssk}}(\mathrm{id}_j)$ with spk, and aborts if the verification fails. Otherwise, the DSN data manager recovers id_j.

Now it comes to the important part of the auditing process. To audit the integrity of data file, a DSN data manager first sends an auditing request to the TPA. After receiving an auditing request, the TPA generates an auditing challenge message as follows.

(1) Randomly choose a c-element subset $\mathcal{J}$ of set $\{1, \ldots, n\}$ to locate the c selected data blocks in this auditing task.

(2) For each $j \in \mathcal{J}$, the TPA also chooses a random value ν_j.

(3) Output an auditing challenge message $\mathrm{chal} = \{(j, \nu_j)\}_{j\in\mathcal{J}}$ and send it to the cloud server; the chal message specifies the positions of the data blocks required to be checked.

After receiving an auditing challenge message chal, the cloud server generates a response proof of possession of selected data blocks storage correctness as follows.

(1) Compute $r = \prod_{j\in\mathcal{J}} r_j^{\nu_j r_j} \bmod p$.

(2) Compute $s = \sum_{j\in\mathcal{J}} \nu_j s_j \bmod q$.

(3) Compute μ'_ℓ as the linear combination of sampled blocks: $\mu'_\ell = \sum_{j\in\mathcal{J}} \nu_j m'_{j,\ell} \in Z_q$, where $\ell \in \{1, \ldots, k\}$.

To blind μ'_ℓ, the cloud server chooses a random element $\eta_\ell \leftarrow Z_q$, and then it calculates $W_\ell = y^{\eta_\ell}$ and $\mu_\ell = \mu'_\ell + \eta_\ell h(W_\ell)$. Finally, the cloud server sends $\{\mu, r, s, W, \{\mathrm{id}_j\}_{j\in\mathcal{J}}\}$ to the TPA for auditing, where $\mu = (\mu_1, \ldots, \mu_k)$ and $W = (W_1, \ldots, W_k)$.

ProofVerify. Given an auditing response proof $\{\mu, r, s, W, \{\mathrm{id}_j\}_{j\in\mathcal{J}}\}$, an auditing message $\mathrm{chal} = \{(j, \nu_j)\}_{j\in\mathcal{J}}$. The TPA verifies the correctness of this proof as follows.

(1) Generate $\rho = (\rho_1, \ldots, \rho_k) \leftarrow \mathrm{PRG}(\mathrm{sk}_{\mathrm{prg}}) \in Z_q^k$ and $\omega_j \leftarrow \mathrm{PRF}(\mathrm{sk}_{\mathrm{prf}}, \mathrm{id}_j) \in Z_q$, $j \in \mathcal{J}$.

(2) Compute $\lambda_1 = \sum_{\ell=1}^k \rho_\ell \mu_\ell + \sum_{j\in\mathcal{J}} \nu_j \omega_j \in Z_q$, $\lambda_2 = \sum_{\ell=1}^k \sum_{j\in\mathcal{J}} \rho_\ell \nu_j f_\tau(\ell, \mathrm{id}_j) \in Z_q$, and $h(W_\ell)$, where $\ell \in \{1, \ldots, k\}$.

(3) Verify the response proof by checking whether the verification equation $g^s = r y^{\lambda_1 - \lambda_2} \prod_{\ell=1}^k W_\ell^{-\rho_\ell h(W_\ell)} \bmod p$ holds or not.

If the verification equation $g^s = r y^{\lambda_1 - \lambda_2} \prod_{\ell=1}^k W_\ell^{-\rho_\ell h(W_\ell)} \bmod p$ holds, the DSN data manager can believe that the integrity of the data file stored in the cloud server is correct, it is not modified by others, and, with the random masking codes $W_{\{1\le\ell\le k\}}$, the TPA can never recover the primitive data blocks from the DSN manager's data file.

4. Analysis of the Proposed Auditing Scheme

In this section, we begin to analyze the proposed auditing scheme, including its correctness, unforgeability, and privacy-preserving. Considering the scalability of the auditing scheme, we also extend it to support batch auditing.

4.1. Correctness. According to the *ProofVerify* phase of the auditing scheme, the correctness of the verification equation is elaborated as follows:

$$\begin{aligned}
g^s &= g^{\sum_{j\in\mathcal{J}} \nu_j s_j} = g^{\sum_{j\in\mathcal{J}} \nu_j (r'_j k_j + t_j x (\bmod q))} \bmod p \\
&= g^{\sum_{j\in\mathcal{J}} \nu_j r'_j k_j} g^{\sum_{j\in\mathcal{J}} \nu_j t_j x} \bmod p \\
&= \prod_{j\in\mathcal{J}} r_j^{\nu_j r_j} y^{\sum_{j\in\mathcal{J}} \nu_j t_j} \bmod p \\
&= r y^{\sum_{j\in\mathcal{J}} (\sum_{\ell=1}^k \rho_\ell m_{j,\ell} + \omega_j) \nu_j} \bmod p \\
&= r y^{\sum_{j\in\mathcal{J}} \sum_{\ell=1}^k \rho_\ell \nu_j m_{j,\ell} + \sum_{j\in\mathcal{J}} \omega_j \nu_j} \bmod p \\
&= r y^{\sum_{\ell=1}^k \rho_\ell \sum_{j\in\mathcal{J}} \nu_j m_{j,\ell} + \sum_{j\in\mathcal{J}} \omega_j \nu_j} \bmod p \\
&= r y^{\sum_{\ell=1}^k \rho_\ell (\mu_\ell - \sum_{j\in\mathcal{J}} \nu_j f_\tau(\ell, \mathrm{id}_j) - \eta_\ell h(W_\ell)) + \sum_{j\in\mathcal{J}} \nu_j \omega_j} \bmod p \\
&= r y^{\sum_{\ell=1}^k \rho_\ell \mu_\ell + \sum_{j\in\mathcal{J}} \nu_j \omega_j} \\
&\quad \cdot y^{-\sum_{\ell=1}^k \sum_{j\in\mathcal{J}} \rho_\ell \nu_j f_\tau(\ell, \mathrm{id}_j)} y^{-\sum_{\ell=1}^k \rho_\ell \eta_\ell h(W_\ell)} \bmod p \\
&= r y^{\lambda_1 - \lambda_2} \prod_{\ell=1}^k W_\ell^{-\rho_\ell h(W_\ell)} \bmod p.
\end{aligned} \tag{1}$$

Thus the verification equation $g^s = r y^{\lambda_1 - \lambda_2} \prod_{\ell=1}^k W_\ell^{-\rho_\ell h(W_\ell)} \bmod p$ holds.

4.2. Unforgeability

Theorem 3. *With the from DSN data manager's data file M' and the corresponding signatures stored in the cloud server, a malicious cloud server is computationally infeasible to generate an invalid auditing response proof that can pass the verification equation.*

In the proposed auditing scheme, we make use of homomorphic MACs to compress each data block to efficiently decrease the amount of storage space needed to store verification information. According to the discussions and proofs in [27], we know that the probability for an adversary to break one homomorphic MAC on a data block is $1/q$, which is negligible.

Besides generating a forgery of a homomorphic MAC, if the malicious cloud server can win Game 1, it can generate an invalid auditing response proof for the challenged data blocks and enable this invalid auditing response proof to successfully pass the verification. Now we describe Game 1 as follows.

Game 1. After receiving an auditing message from the DSN data manager, the TPA sends an auditing challenge message $chal = \{(j, v_j)\}_{j\in\mathcal{J}}$ to the cloud server, and the correct auditing response proof should be $\{\mu, r, s, W, \{\mathrm{id}_j\}_{j\in\mathcal{J}}\}$, where $\mu = (\mu_1, \ldots, \mu_k)$, $W = (W_1, \ldots, W_k)$. The response proof can pass the verification equation. Now, instead of generating the correct auditing response proof, the malicious cloud server generates an invalid auditing proof as $\{\mu^*, r, s, W, \{\mathrm{id}_j\}_{j\in\mathcal{J}}\}$ based on the corrupted data file M'^*, where $\mu^* = (\mu_1^*, \ldots, \mu_k^*)$, $\mu_\ell^* = \mu_\ell'^* + \eta_\ell h(W_\ell)$, and $\mu_\ell'^* = \sum_{j\in\mathcal{J}} v_j m_{j,\ell}'^* \in Z_q$. Define $\Delta\mu_\ell = \mu_\ell^* - \mu_\ell$ for $1 \le \ell \le k$, since $M' \neq M'^*$, and thus there is at least one element of $\{\Delta\mu_\ell\}_{1\le\ell\le k}$ which is nonzero. If this invalid response proof can still pass the verification, the malicious cloud wins Game 1. Otherwise, it fails.

Now we begin to show that if the malicious cloud can win the above Game 1, we can find a solution to the Discrete Logarithm problem. We first assume that the malicious cloud wins Game 1. Then, according to the verification equation, we have $g^s = r y^{\lambda_1^* - \lambda_2} \prod_{\ell=1}^k W_\ell^{-\rho_\ell h(W_\ell)} \bmod p$, where $\lambda_1^* = \sum_{\ell=1}^k \rho_\ell \mu_\ell^* + \sum_{j\in\mathcal{J}} v_j \omega_j \in Z_q$. Since $\{\mu, r, s, W, \{\mathrm{id}_j\}_{j\in\mathcal{J}}\}$ is the correct auditing response proof, we also have $g^s = r y^{\lambda_1 - \lambda_2} \prod_{\ell=1}^k W_\ell^{-\rho_\ell h(W_\ell)} \bmod p$. Then, according to the two verification equations, we learn that $y^{\lambda_1^*} = y^{\lambda_1}$. Thus

$$\begin{aligned} y^{\sum_{\ell=1}^k \rho_\ell \mu_\ell^* + \sum_{j\in\mathcal{J}} v_j \omega_j} &= y^{\sum_{\ell=1}^k \rho_\ell \mu_\ell + \sum_{j\in\mathcal{J}} v_j \omega_j}, \\ y^{\sum_{\ell=1}^k \rho_\ell \mu_\ell^*} &= y^{\sum_{\ell=1}^k \rho_\ell \mu_\ell}, \\ y^{\sum_{\ell=1}^k \rho_\ell \Delta\mu_\ell} &= \prod_{\ell=1}^k \left(y^{\rho_\ell}\right)^{\Delta\mu_\ell} = 1. \end{aligned} \tag{2}$$

Because G is a multiplicative cyclic group of order q, for two random elements $\alpha, \beta \in G$, there exists $\eta \in Z_q$ such that $\beta = \alpha^\eta$. Without loss of generality, given $\alpha, \beta \in G$, each y^{ρ_ℓ} is able to randomly and correctly be generated by computing $y^{\rho_\ell} = \alpha^{\xi_\ell} \beta^{\gamma_\ell}$, where ξ_ℓ and γ_ℓ are random values in Z_q. Then we get

$$\begin{aligned} 1 &= \prod_{\ell=1}^k \left(y^{\rho_\ell}\right)^{\Delta\mu_\ell} \\ &= \prod_{\ell=1}^k \left(\alpha^{\xi_\ell} \beta^{\gamma_\ell}\right)^{\Delta\mu_\ell} \\ &= \alpha^{\sum_{\ell=1}^k \xi_\ell \Delta\mu_\ell} \cdot \beta^{\sum_{\ell=1}^k \gamma_\ell \Delta\mu_\ell}. \end{aligned} \tag{3}$$

Obviously, we can find a solution to the Discrete Logarithm problem. Particularly, given α, $\beta = \alpha^\eta \in G$, we can output $\beta = \alpha^\eta = \alpha^{-\sum_{\ell=1}^k \xi_\ell \Delta\mu_\ell / \sum_{\ell=1}^k \gamma_\ell \Delta\mu_\ell}$; thus $\eta = -\sum_{\ell=1}^k \xi_\ell \Delta\mu_\ell / \sum_{\ell=1}^k \gamma_\ell \Delta\mu_\ell$, unless the denominator is zero. However, as we defined in Game 1, there is at least one element of $\{\Delta\mu_\ell\}$ which is nonzero, and γ_ℓ is a random element of Z_q. Therefore, the denominator is zero with probability of $1/q$, which is negligible. It means that once the malicious cloud wins Game 1, we can find a solution to the Discrete Logarithm problem with a nonnegligible probability of $1 - 1/q$, which contradicts to the assumption that Discrete Logarithm problem is computationally infeasible in G.

Moreover, if the malicious cloud server tries to forge the aggregate signature, that means the cloud server generates an invalid response proof as $\{\mu, r', s', W, \{\mathrm{id}_j\}_{j\in J}\}$, this invalid response proof can still pass the verification equation $g^{s'} = r' y^{\lambda_1 - \lambda_2} \prod_{\ell=1}^k W_\ell^{-\rho_\ell h(W_\ell)} \bmod p$, and the malicious cloud server can succeed. As we know that the correct auditing response proof should be $\{\mu, r, s, W, \{\mathrm{id}\}_{j\in J}\}$, which can pass the verification equation $g^s = r y^{\lambda_1 - \lambda_2} \prod_{\ell=1}^k W_\ell^{-\rho_\ell h(W_\ell)} \bmod p$, according to the two verification equations, we get that $g^{s'-s} = r' r^{-1} \bmod p$; thus we get $s = s'$ and $r = r'$, or we can find a solution of the Discrete Logarithm problem between g and d (here we set $d = r' r^{-1}$), and these two results both contradict to our assumption.

Therefore, it is computationally infeasible for the malicious cloud to generate an invalid auditing proof, which can pass the verification equation.

4.3. Privacy-Preserving

Theorem 4. *Given an auditing response proof message proof $= \{\mu, r, s, W, \{id_j\}_{j\in\mathcal{J}}\}$ from the cloud server, it is computationally infeasible for the curious TPA to reveal any private data block from the data file of the DSN data manager.*

Proof. If the combined message $\mu_\ell' = \sum_{j\in\mathcal{J}} v_j m_{j,\ell}' \in Z_q$, which is a linear combination of elements in data blocks, is directly sent to the TPA, the curious TPA can learn the content of data blocks by solving linear equations after collecting a sufficient number of linear combinations. To preserve private data blocks from the TPA, the combined message is computed with random masking as $\mu_\ell = \mu_\ell' + \eta_\ell h(W_\ell)$. In order to still solve linear equations, the TPA must know the value of η_ℓ. However, given y, $W_\ell = y^{\eta_\ell} \in G$, computing η_ℓ is as hard as solving the Discrete Logarithm problem in G, which is computationally infeasible. Therefore, given the auditing response proof message, the TPA cannot directly obtain any linear combination of elements in data blocks and cannot further reveal any private data block from the data file by solving linear equations. □

4.4. Support for Batch Auditing. With the usage of privacy-preserving auditing scheme in the cloud storage, the TPA may receive amount of multiple auditing requests from different DSN data managers in a short time. Unfortunately, allowing the TPA to execute the separate auditing task can be tedious and very inefficient. Therefore, we further extend our scheme to support batch auditing. Batch auditing not only allows the TPA to execute the multiple auditing tasks simultaneously,

but also dramatically decreases the computation cost on the TPA side. This is because aggregating L verification equations into one helps save a considerable amount of auditing time. The details are described as follows.

Setup Phase. The DSN data managers just perform setup independently. Suppose there are L DSN data managers in the auditing system, and each DSN data manager θ has a data file $M_\theta = \{m_{\theta,1}, \ldots, m_{\theta,n}\}$ to be outsourced to the cloud server, where $m_{\theta,j} = (m_{\theta,j,1}, \ldots, m_{\theta,j,k})$, $j = 1, 2, \ldots, n$. For simplicity, we assume each data file M_θ has the same number of n data blocks. Particularly, for a DSN data manager θ, denote his private parameters by $(x_\theta, \mathrm{ssk}_\theta, \mathrm{sk}_{\mathrm{prg}_\theta}, \mathrm{sk}_{\mathrm{prf}_\theta})$ and the corresponding public parameters by $(G, g, y_\theta, \mathrm{spk}_\theta)$, where $y_\theta = g^{x_\theta}$. As it is similar to the single DSN data manager case, each DSN data manager θ has already randomly chosen a different identity $\mathrm{id}_{\theta,j}$ for the data block $m_{\theta,j}$ and has correctly generated the corresponding data block $\mathrm{tag}_{\theta,j} = \mathrm{id}_{\theta,j} \parallel \mathrm{SSig}_{\mathrm{ssk}_\theta}(\mathrm{id}_{\theta,j})$.

Then each DSN data manager θ computes $\rho_\theta = (\rho_{\theta,1}, \ldots, \rho_{\theta,k}) \leftarrow \mathrm{PRG}(\mathrm{sk}_{\mathrm{prg}_\theta}) \in Z_q^k$ and $\omega_{\theta,j} \leftarrow \mathrm{PRF}(\mathrm{sk}_{\mathrm{prf}_\theta}, \mathrm{id}_{\theta,j}) \in Z_q$. Then the DSN data manager calculates the homomorphic MAC of data block $m_{\theta,j} = (m_{\theta,j,1}, \ldots, m_{\theta,j,k})$ as $t_{\theta,j} = \sum_{\ell=1}^{k} \rho_{\theta,\ell} m_{\theta,j,\ell} + \omega_{\theta,j} \in Z_q$. The DSN data manager begins to compute the signature of $t_{\theta,j}$ as follows.

(1) Choose $k_{\theta,j} \leftarrow Z_q$ and compute $r_{\theta,j} \equiv g^{k_{\theta,j}} \bmod p$ and $r'_{\theta,j} \equiv r_{\theta,j} \bmod q$;

(2) $s_{\theta,j} = (r'_{\theta,j} k_{\theta,j} + t_{\theta,j} x) \bmod q$,

(3) output $\sigma_{\theta,j} = (r_{\theta,j}, s_{\theta,j})$ as the signature of $t_{\theta,j}$.

Denote the set of signatures by $\Phi_\theta = \{\sigma_{\theta,j}\}_{1 \le j \le n}$. Meanwhile, to guarantee the confidentiality of the data file, the DSN data manager employs the lightweight symmetry encryption algorithm f to encrypt each data block $m_{\theta,j} = (m_{\theta,j,1}, \ldots, m_{\theta,j,k})$ as $m'_{\theta,j} = (m_{\theta,j,1} + f_{\tau_\theta}(1, \mathrm{id}_{\theta,j}), \ldots, m_{\theta,j,\mathrm{k}} + f_{\tau_\theta}(k, \mathrm{id}_{\theta,j}))$ under the symmetry private key τ_θ. Thus, the data file $M_\theta = (m_{\theta,1}, \ldots, m_{\theta,n})$ is encrypted to be $M'_\theta = (m'_{\theta,1}, \ldots, m'_{\theta,n})$. Finally, the DSN data manager θ sends $\{M'_\theta, \{\mathrm{tag}_{\theta,j}\}_{1 \le j \le n}, \Phi_\theta\}$ to the cloud server and deletes them from local storage.

Audit Phase. The TPA first retrieves and verifies the data block $\mathrm{tag}_{\theta,j}$ for each DSN data manager θ for later auditing. If the verification fails, the TPA aborts. Otherwise, the TPA recovers $\mathrm{id}_{j,\theta}$ and sends the auditing challenge message $\mathrm{chal} = \{(j, \nu_j)\}_{j \in \mathcal{J}}$ to the cloud server. Meanwhile, for each DSN data manager θ, the cloud server chooses $\eta_{\theta,\ell} \in Z_q$ randomly as before and computes $W_{\theta,\ell} = y_\theta^{\eta_{\theta,\ell}}$ and $\mu_{\theta,\ell} = \sum_{j \in \mathcal{J}} \nu_j m'_{\theta,j,\ell} + \eta_{\theta,\ell} h(W_{\theta,\ell})$; thus, the cloud server can compute $\mu_\theta = (\mu_{\theta,1}, \ldots, \mu_{\theta,\ell}, \ldots, \mu_{\theta,k})$. Then the cloud server makes the aggregation as $r = \Pi_{\theta=1}^{L} \Pi_{j \in \mathcal{J}} r_{\theta,j}^{\nu_j r_{\theta,j}} \bmod p$ and $s = \sum_{\theta=1}^{L} \sum_{j \in \mathcal{J}} \nu_j s_{\theta,j} \bmod q$. Finally, the cloud server responses with $(\{\mu_\theta\}_{1 \le \theta \le L}, r, s, \{W_\theta\}_{1 \le \theta \le L}, \{\mathrm{id}_{\theta,j}\}_{j \in \mathcal{J}, 1 \le \theta \le L})$, where $W_\theta = (W_{\theta,1}, \ldots, W_{\theta,\ell}, \ldots, W_{\theta,k})$.

To verify the response, the TPA first does as follows.

(1) Generate $\rho_\theta = (\rho_{\theta,1}, \ldots, \rho_{\theta,k}) \leftarrow \mathrm{PRG}(\mathrm{sk}_{\mathrm{prg}_\theta}) \in Z_q^k$ and $\omega_{\theta,j} \leftarrow \mathrm{PRF}(\mathrm{sk}_{\mathrm{prf}_\theta}, \mathrm{id}_{\theta,j}) \in Z_q$, $j \in \mathcal{J}$.

(2) Compute $\lambda_{\theta,1} = \sum_{\ell=1}^{k} \rho_{\theta,\ell} \mu_{\theta,\ell} + \sum_{j \in \mathcal{J}} \nu_j \omega_{\theta,j} \in Z_q$, $\lambda_{\theta,2} = \sum_{\ell=1}^{k} \sum_{j \in \mathcal{J}} \rho_{\theta,\ell} \nu_j f_{\tau_\theta}(\ell, id_{\theta,j}) \in Z_q$, and $h(W_{\theta,\ell})$, where $1 \le \ell \le k$ and $1 \le \theta \le L$.

Then the TPA checks if the following verification equation holds: $g^s = r \prod_{\theta=1}^{L} y_\theta^{\lambda_{\theta,1} - \lambda_{\theta,2}} (\prod_{\ell=1}^{k} W_{\theta,\ell}^{-\rho_{\theta,\ell} h(W_{\theta,\ell})}) \bmod p$. The correctness of the verification equation can be shown as follows:

$$
\begin{aligned}
g^s &= g^{\sum_{\theta=1}^{L} \sum_{j \in \mathcal{J}} \nu_j s_{\theta,j}} \bmod p \\
&= \prod_{\theta=1}^{L} g^{\sum_{j \in \mathcal{J}} \nu_j s_{\theta,j}} \bmod p \\
&= \prod_{\theta=1}^{L} g^{\sum_{j \in \mathcal{J}} \nu_j (r'_{\theta,j} k_{\theta,j} + t_{\theta,j} x_\theta (\bmod q))} \bmod p \\
&= \prod_{\theta=1}^{L} g^{\sum_{j \in \mathcal{J}} \nu_j r'_{\theta,j} k_{\theta,j}} g^{\sum_{j \in \mathcal{J}} \nu_j t_{\theta,j} x_\theta} \bmod p \\
&= \prod_{\theta=1}^{L} \prod_{j \in \mathcal{J}} r_{\theta,j}^{\nu_j r_{\theta,j}} y_\theta^{\sum_{j \in \mathcal{J}} \nu_j t_{\theta,j}} \bmod p \\
&= r \prod_{\theta=1}^{L} y_\theta^{\sum_{j \in \mathcal{J}} \nu_j (\sum_{\ell=1}^{k} \rho_{\theta,\ell} m_{\theta,j,\ell} + \omega_{\theta,j})} \bmod p
\end{aligned}
$$

$$
\begin{aligned}
&= r\prod_{\theta=1}^{L} y_\theta^{\sum_{\ell=1}^{k}\rho_{\theta,\ell}\sum_{j\in\mathcal{J}}\nu_j m_{\theta,j,\ell}+\sum_{j\in\mathcal{J}}\nu_j\omega_{\theta,j}} \bmod p \\
&= r\prod_{\theta=1}^{L} y_\theta^{\sum_{\ell=1}^{k}\rho_{\theta,\ell}(\mu_{\theta,\ell}-\sum_{j\in\mathcal{J}}\nu_j f_{\tau_\theta}(\ell,\mathrm{id}_{\theta,j})-\eta_{\theta,\ell}h(W_{\theta,\ell}))+\sum_{j\in\mathcal{J}}\nu_j\omega_{\theta,j}} \bmod p \\
&= r\prod_{\theta=1}^{L} y_\theta^{\sum_{\ell=1}^{k}\rho_{\theta,\ell}\mu_{\theta,\ell}+\sum_{j\in\mathcal{J}}\nu_j\omega_{\theta,j}-\sum_{\ell=1}^{k}\sum_{j\in\mathcal{J}}\rho_{\theta,\ell}\nu_j f_{\tau_\theta}(\ell,\mathrm{id}_{\theta,j})-\sum_{\ell=1}^{k}\rho_{\theta,\ell}\eta_{\theta,\ell}h(W_{\theta,\ell})} \bmod p \\
&= r\prod_{\theta=1}^{L} y_\theta^{\lambda_{\theta,1}-\lambda_{\theta,2}}\, y_\theta^{-\sum_{\ell=1}^{k}\rho_{\theta,\ell}\eta_{\theta,\ell}h(W_{\theta,\ell})} \bmod p \\
&= r\prod_{\theta=1}^{L} y_\theta^{\lambda_{\theta,1}-\lambda_{\theta,2}}\left(\prod_{\ell=1}^{k} W_{\theta,\ell}^{-\rho_{\theta,\ell}h(W_{\theta,\ell})}\right) \bmod p.
\end{aligned}
\tag{4}
$$

Thus the verification equation $g^s = r\prod_{\theta=1}^{L} y_\theta^{\lambda_{\theta,1}-\lambda_{\theta,2}} (\prod_{\ell=1}^{k} W_{\theta,\ell}^{-\rho_{\theta,\ell}h(W_{\theta,\ell})}) \bmod p$ holds.

5. Performance Comparison

In this section, we begin to compare the performance of our privacy-preserving auditing scheme for cloud storage with the auditing scheme in [26]. We first focus on discussing the computation cost and the communication cost. Then we evaluate the performance comparison between the two schemes in experiments to show our auditing scheme advantages.

5.1. Computation Cost. We first give the computation cost of our pairing-free auditing scheme for cloud storage with the auditing scheme in [26]. The main cryptographic operations used in our scheme include multiplications, additions, and hash operations. For simplicity, we omit the computation cost of the pseudorandom number generator PRG and pseudorandom function PRF because they are much easier to be computed than the three types of operations mentioned above. Here, we denote Mult_G, Add_G, and Exp_G by multiplication, addition, and modular exponentiation operation in group G, respectively; we also denote Hash_G by hash operation into the group G, and we denote pair_{G_1,G_2} by pairing operation.

During the auditing process, the TPA first generates some random values to construct the auditing message, which only introduces a small cost in computation. Then, after receiving the auditing message, the cloud server needs to compute a proof $\{\mu, r, s, W, \{\mathrm{id}_j\}_{j\in\mathcal{J}}\}$ to the TPA for auditing, where $\mu = (\mu_1, \ldots, \mu_k)$ and $W = (W_1, \ldots, W_k)$. The computation cost of a proof is about $(kc+2c+k)\mathrm{Mult}_{Z_p} + c\mathrm{Mult}_{Z_q} + kc\mathrm{Add}_{Z_p} + (c-1)\mathrm{Add}_{Z_q} + k\mathrm{Exp}_{Z_p} + k\mathrm{Hash}_{Z_p}$, while the computation cost of a proof in [26] is about $(c-1)\mathrm{Mult}_{G_1} + (c+1)\mathrm{Mult}_{Z_p} + c\mathrm{Exp}_{G_1} + \mathrm{Exp}_{G_T} + c\mathrm{Add}_{Z_p} + \mathrm{Hash}_{Z_p}$.

To check the correctness of the proof, the TPA verifies it based on verification equation and the computation cost of verifying the auditing proof is $(2k+c+2ck)\mathrm{Mult}_{Z_q} + k\mathrm{Mult}_{Z_p} + (k+2)\mathrm{Exp}_{Z_p} + (ck+c+k-2)\mathrm{Add}_{Z_q} + 2k\mathrm{Hash}_{Z_q} + c\mathrm{Enc}_\varepsilon$, while the computation cost of verifying the auditing proof in [26] is $(c+1)\mathrm{Mult}_{G_1} + \mathrm{Mult}_{G_T} + (c+3)\mathrm{Exp}_{G_1} + 2\mathrm{Pair}_{G_1,G_2} + \mathrm{Hash}_{Z_p} + c\mathrm{Hash}_{G_1}$.

5.2. Communication Cost. The communication cost of our scheme is mainly introduced by two factors: the auditing message and the auditing proof. For the auditing message $\mathrm{chal} = \{(j, \nu_j)\}_{j\in\mathcal{J}}$, the auditing proof information generated by the cloud server is $\{\mu, r, s, W, \{\mathrm{id}_j\}_{j\in\mathcal{J}}\}$, where $\mu = (\mu_1, \ldots, \mu_k)$ and $W = (W_1, \ldots, W_k)$; thus the total communication cost of our auditing scheme is $(c + k + 1)|q| + c|n| + (k + 1)|p|$, while the total communication cost of the auditing scheme in [26] is $c(|p| + |n|) + |p| + |G_1| + |G_T| + |\mathrm{id}|$, where $|n|$ is the length of an index and $|G_T|$ is the length of an element of G_T. Moreover, the communication overhead of $|G_1|$ and $|G_T|$ in [26] is much larger than others; therefore our auditing scheme is more light-weight than [26] in communication cost.

5.3. Experimental Results. We now compare the cloud server computation cost and the TPA auditing computational cost of our auditing scheme with the work of [26] in experiments. Since the random mask needs one exponentiation operation, one multiplication operation, one hash and one addition operation, so the sum of the extra cost that resulted from the random mask only needs a constant, $\mathrm{Exp}_{G_T} + \mathrm{Mult}_{Z_p} + \mathrm{Hash}_{Z_p} + \mathrm{Add}_{Z_p}$, which has nothing to do with the number of sampled blocks c. When c is set to be 400 to 600 for high assurance of auditing, the extra cost on the cloud server side for privacy-preserving guarantee would be negligible against the total server computation for response generation. Therefore, the main computation cost of the cloud server in [26] is $(c-1)\mathrm{Mult}_{G_1} + c\mathrm{Mult}_{Z_p} + c\mathrm{Exp}_{G_1} + (c-1)\mathrm{Add}_{Z_p}$ in our experiments. However, in our auditing scheme, the extra cost resulting from the random masking is only a small constant: $k(\mathrm{Exp}_{Z_p} + \mathrm{Mult}_{Z_p} + \mathrm{Hash}_{Z_q} + \mathrm{Add}_{Z_q})$, where k is much less than

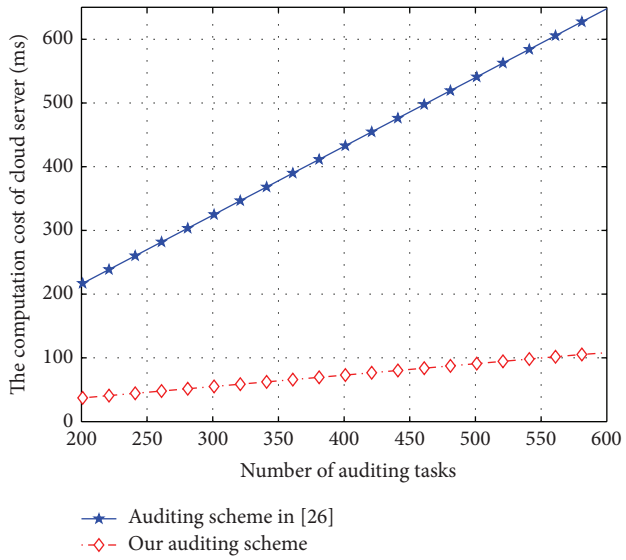

Figure 2: Comparison on the computation cost of cloud server.

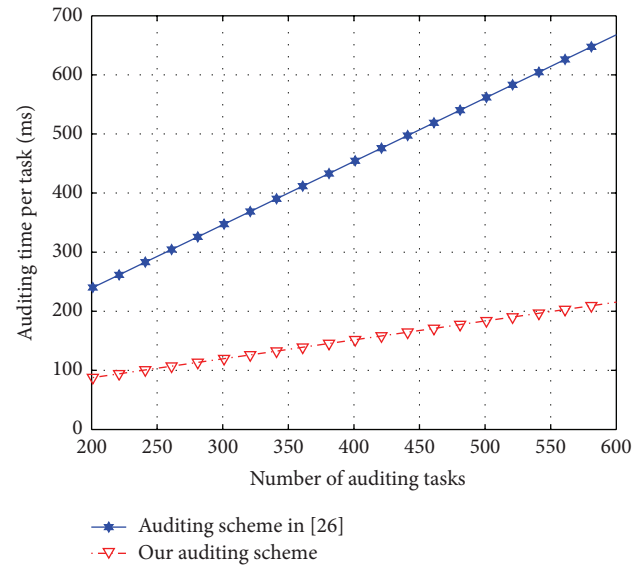

Figure 3: Comparison on the auditing time between our scheme and the scheme in [26].

the practical challenge number of the data blocks. Here we can omit the computation cost $k(\text{Mult}_{Z_p} + \text{Hash}_{Z_q} + \text{Add}_{Z_q})$. Therefore, in our experiments we set the main cloud server computation cost to be $(kc + 2c)\text{Mult}_{Z_p} + c\text{Mult}_{Z_q} + (kc - k)\text{Add}_{Z_p} + (c-1)\text{Add}_{Z_q} + k\text{Exp}_{Z_p}$.

As also discussed in [26], the extra cost resulting from the random masking is only a constant: $\text{Mult}_{G_T} + 2\text{Exp}_{G_1} + \text{Hash}_{Z_p}$, which has nothing to do with the number of sampled blocks c. As considering the relatively expensive pairing operations, the extra cost for privacy-preserving guarantee would be also negligible against the overall cost of response validation. Therefore, here we set the main auditing computation cost of the TPA to be $c\text{Mult}_{G_1} + (c+1)\text{Exp}_{G_1} + 2\text{Pair}_{G_1,G_2} + c\text{Hash}_{G_1}$ in our experiments. However, in our auditing scheme, the extra cost resulting from the random masking is $k(\text{Mult}_{Z_p} + \text{Mult}_{Z_q} + \text{Hash}_{Z_q} + \text{Exp}_{Z_p})$, where k is much less than the practical challenge number of the data blocks. Since the modular exponentiation operation is much larger than others, here we can omit the computation cost $k(\text{Mult}_{Z_p} + \text{Mult}_{Z_q} + \text{Hash}_{Z_q})$. For consistence, we also set the main auditing computation cost of the TPA to be $(k + c + 2ck)\text{Mult}_{Z_q} + (k+2)\text{Exp}_{Z_p} + (ck + c + k - 2)\text{Add}_{Z_q} + k\text{Hash}_{Z_q} + c\text{Enc}_\varepsilon$.

Our experiments are implemented on a Windows 7 system with an Intel Core 2 i5 CPU running at 2.53 GHz, 2 GB DDR 3 of RAM (1.74 GB available). All algorithms are implemented by C language, and our code uses the MIRACL library version 5.6.1. The elliptic curve we use is a MNT curve, the base field size is 159 bits, and the embedding degree is 6. The security level is chosen to be 80 bit, and $|p| = |q| = 160$. For simplicity, we also set $k = 20$. All the results of experiments are represented as the average of 30 trials.

As described in Figures 2 and 3, the experimental results show that, compared with the auditing scheme in [26], the computation cost of the cloud server and the TPA auditing time of our auditing scheme are much more light-weight than [26]. More specifically, with the increasing of the number of challenge data blocks, our auditing scheme is more advantageous than [26] in computation cost. This is mainly because the auditing scheme in [26] needs very expensive pairing computation which is much more time-consuming.

6. Conclusions

Data outsourcing, one of the fundamental components of cloud computing, centralizes DSN data manager's data to the cloud server and enables the DSN data managers to enjoy high quality service. However, the DSN data managers do not have physical possession on their own data; hence it is indispensable to create schemes on how to protect the security of the data, unlike the previous auditing schemes [26] which need expensive pairing operations. In this paper, we propose a pairing-free privacy-preserving auditing scheme for data storage security in distributed sensor networks. We employ the homomorphic linear authenticator and random masking to guarantee that the TPA would not only eliminate the burden of the DSN data managers from the tedious and possible expensive auditing task, but also alleviate the DSN data managers' fear of their outsourced data leakage. We also utilize homomorphic MACs to effectively reduce the amount of storage space needed to store verification information. Moreover, we further extend our auditing scheme to support batch auditing for multiple DSN data managers, where the TPA can perform multiple auditing tasks simultaneously. Extensive security and performance compared analysis shows that the proposed auditing scheme is more light-weight and more practical in distributed sensor networks environments.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (no. 61370203) and the Science and Technology on Communication Security Laboratory Foundation (Grant no. 9140C110301110C1103).

References

[1] F. Ye, H. Luo, J. Cheng, S. Lu, and L. Zhang, "A two-tier data dissemination model for large-scale wireless sensor networks," in *Proceedings of the 8th ACMe on Mobile Computing and Networking (MOBICOM '02)*, pp. 148–159, September 2002.

[2] G. Wang, G. Cao, T. la Porta, and W. Zhang, "Sensor relocation in mobile sensor networks," in *Proceedings of the IEEE INFOCOM*, pp. 2302–2312, March 2005.

[3] E. Mykletun, J. Girao, and D. Westhoff, "Public key based cryptoschemes for data concealment in wireless sensor networks," in *Proceedings of the IEEE International Conference on Communications (ICC '06)*, vol. 5, pp. 2288–2295, July 2006.

[4] J. Girao, D. Westhoff, E. Mykletun, and T. Araki, "*TinyPEDS*: tiny persistent encrypted data storage in asynchronous wireless sensor networks," *Ad Hoc Networks*, vol. 5, no. 7, pp. 1073–1089, 2007.

[5] P. Mell and T. Grance, "The NIST definition of cloud computing," *National Institute of Standards and Technology*, vol. 53, no. 6, p. 50, 2009.

[6] N. Subramanian, C. Yang, and W. Zhang, "Securing distributed data storage and retrieval in sensor networks," *Pervasive and Mobile Computing*, vol. 3, no. 6, pp. 659–676, 2007.

[7] J. Kincaid, "MediaMax/Thelinkup Close Its Doors," 2009, http://techcrunch.com/2008/07/10/mediamaxthelinkup-closes-its-doors/.

[8] Amazon.com, *Amazon s3 Availability Events: July 20, 2008*, 2008, http://status.aws.amazon.com/s3-20080720.html.

[9] Cloud Security Alliance, *Top Threats to Cloud Computing*, 2010, http://www.cloudsecurityalliance.org.

[10] T. Schwarz and E. L. Miller, "Store, forget, and check: using algebraic signatures to check remotely administered storage," in *Proceedings of the 26th IEEE International Conference on Distributed Computing Systems (ICDCS '06)*, July 2006.

[11] S. Yu, C. Wang, K. Ren, and W. Lou, "Achieving secure, scalable, and fine-grained data access control in cloud computing," in *Proceedings of the 29th IEEE Conference on Information Communications (INFOCOM '10)*, pp. 534–542, March 2010.

[12] M. Li, S. Yu, K. Ren, and W. Lou, "Secure personal health records in cloud computing:patient-centric and fine-grained data access control in multi-owner settings," in *Security an Privacy in Communication Networks*, pp. 89–106, Springer, Berlin, Germany, 2010.

[13] V. Kher and Y. Kim, "Securing distributed storage: challenges, techniques, and systems," in *Proceedings of the ACM Workshop on Storage Security and Survivability (StorageSS '05)*, pp. 9–25, November 2005.

[14] B. Schroeder and G. A. Gibson, "Disk failures in the real world: what does an MTTF of 1,000,000 hours mean to you?" in *Proceedings of the 5th USENIX Conference on File and Storage Technologies (FAST '07)*, pp. 1–16, ACM, New York, NY, USA, 2007.

[15] A. Muthitacharoen, R. Morris, T. M. Gil, and B. Chen, "Ivy: a read/write peer to peer file system," in *Proceeding of the 5th Symosium on Operation Systems Design and Implementation (OSDI '02)*, pp. 31–44, ACM, 2002.

[16] M. Kallahalla, E. Riedel, R. Swaminathan, Q. Wang, and K. Fu, "Plutus: scalable secure file sharing on untrusted storage," in *Proceedings of the 2nd USENIX Conference on File and Storage Technologies*, pp. 29–42, USENIX Association, San Francisco, Calif, USA, 2003.

[17] J. Li, M. Krohn, D. Mazieres, and D. Shasha, "Secure untrusted data repository (sundr)," in *Proceedings of the 6th Conference on Symposium on Operating Systems Design and Implementation*, p. 9, USENIX Association, Berkeley, Calif, USA, 2004.

[18] A. R. Yumerefendi and J. S. Chase, "Strong accountability for network storage," *ACM Transactions on Storage*, vol. 3, no. 3, article 11, 2007.

[19] U. Maheshwari, R. Vingralek, and W. Shapiro, "How to build a trusted database system on untrusted storage," in *Proceedings of the 4th Conference on Symposium on Operating System Design and Implementation (OSDI '00)*, USENIX Association, San Diego, Calif, USA, 2000.

[20] Q. Wang, C. Wang, L. Li, K. Ren, and W. Lou, "Enabling public verifiability and data dynamics for storage security in cloud computing," in *Proceedings of the 14th European Symposium Research in Computer Security (ESORICS '09)*, pp. 355–370, Saint Malo, France, 2009.

[21] Cloud Security Alliance, "Security guidance for critical areas of focus in cloud computing," 2009, http://cloudsecurityalliance.org/.

[22] G. Ateniese, R. Burns, R. Curtmola et al., "Provable data possession at untrusted stores," in *Proceedings of the 14th ACM Conference on Computer and Communications Security (CCS '07)*, pp. 598–609, November 2007.

[23] A. Juels, J. Burton, and S. Kaliski, "Pors: proofs of retrievability for large files," in *Proceedings of the 14th ACM Conference on Computer and Communications Security (CCS '07)*, pp. 584–597, Alexandia, Va, USA, October 2007.

[24] H. Shacham and B. Waters, "Compact proofs of retrievability," in *Proceedings of the 14th International Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT '08), Melbourne, Australia, December 2008*, vol. 5350 of *Lecture Notes in Computer Science*, pp. 90–107, Springer, 2008.

[25] M. A. Shah, M. M. Baker, J. C. Mogul, and R. Swaminathan, "Auditing to keep online storage services honest," in *Proceedings of the 11th USENIX Workshop on Hot Topics in Operating Systems (HOTOS '07)*, pp. 1–6, USENIX Association, Berkeley, Calif, USA, 2007.

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

[27] S. Agrawal and D. Boneh, "Homomorphic MACs: MAC-based integrity for network coding," in *Proceedings of the 7th International Conference on Applied Cryptography and Network Security (ACNS '09), Paris-Rocquencourt, France, June 2009*, pp. 292–305, Springer, 2009.