Hindawi
Security and Communication Networks
Volume 2022, Article ID 7882662, 17 pages
https://doi.org/10.1155/2022/7882662

Research Article

Security-Enhanced Certificate-Based Remote Data Integrity Batch Auditing for Cloud-IoT

Wenhao Wang,[1] **Yinxia Sun**,[1] **and Yumei Li**[2]

[1]*School of Computer Technology, Nanjing Normal University, Nanjing 210097, China*
[2]*School of Computer Science, Hubei University of Technology, Wuhan 430068, China*

Correspondence should be addressed to Yinxia Sun; 73003@njnu.edu.cn

Received 1 April 2022; Accepted 21 June 2022; Published 31 July 2022

Academic Editor: Jingyu Feng

The Internet of Things (IoT) plays a crucial role in the generation of new, intelligent information technologies. Generally, the IoT facilities are composed of lightweight devices, and they expand computing and storage resources primarily through the cloud. Massive data collected by intelligent devices will be stored in cloud servers, but the vulnerabilities of cloud servers will directly threaten the security and reliability of the IoT. To ensure the integrity of data in the cloud, data owners need to audit the integrity of their outsourced data. Recently, several remote data integrity batch auditing protocols have been proposed to reduce transmission loss and time cost in the auditing process. However, most of them cannot resist collusion attacks. Meanwhile, certificate management problem exists in their system, which brings an enormous burden on the system. In this paper, we construct a certificate-based remote data integrity batch auditing protocol which can issue batch auditing and resist the highest level of collusion attacks—the fully chosen-key attacks for cloud-IoT . Our protocol makes use of a certificate-based cryptosystem which gets rid of the certificate management problem and key escrow problem, with no need for secret channels. Our protocol is proved to be secure in the random oracle model and implemented to show its efficiency. The simulation results illustrate that, in the case of enhanced security, our batch auditing protocol still has computational efficiency and practicability.

1. Introduction

1.1. Background and Motivation. With the introduction and application of new concepts and technologies such as smart cities, virtual sports, and the Metaverse, the number of devices connected to the Internet is increasing, requiring more powerful storage and processing resources. Fortunately, cloud services can provide data outsourcing storage services [1] for data owners (DOs), which are the devices in the IoT. By outsourcing data to a cloud server (CS), a DO is free from the burden of complex data management and huge storage. Meanwhile, a DO can access outsourced data in a network environment anytime and anywhere.

While cloud storage and the IoT are convenient to people's life, they bring some security concerns to the outsourced data [2]. One of the main security problems of cloud storage is the integrity of the outsourced data [3]. On the one hand, the CS is vulnerable to external attacks, resulting in the destruction of the outsourced data's integrity. On the other hand, the IoT devices store data in the storage resources provided by a cloud service provider (CSP) who will maliciously tamper with the outsourced data in order to gain greater profits. If the data stored in the CS is tampered with or damaged, DOs may suffer considerable loss. Therefore, DOs are forced to audit the integrity of their outsourced data.

The local data uploaded to the cloud will be deleted locally at the same time. In this case, the DO audits their own data integrity, which means that they need to download the data stored in the CS to the local server [4], and then locally audit the integrity of the data. It undoubtedly brings more trouble. Remote data integrity auditing (RDIA) can enable the DO to audit the integrity of cloud data and protect its interests without local backup. In practice, DOs prefer to hire a third-party auditor (TPA) to audit the integrity of their outsourced data.

Because of the linearity of the linear homomorphic signature (LHS) [5], any linear combination of valid signatures can form a new valid signature. Thus, the LHS can be used to audit data integrity by random sampling. A TPA can sample random data blocks to audit data integrity without accessing the entire file. A basic RDIA process is shown in Figure 1. Firstly, a TPA gives a challenge to the CSP. Then, the CSP honestly generates a proof which is based on the LHS according to the challenge and returns it to the TPA. Finally, the TPA validates the proof and returns "accept" or "reject" to the DO. Note that the process of a TPA validating the proof is actually the process of verifying the linear homomorphic signature.

To date, many RDIA protocols have been proposed and the vast majority of them only support auditing data from a single DO in a single auditing process. Let us imagine a scenario. If a TPA needs to audit the data of multiple DOs, it needs to interact with the CSP many times, which is undoubtedly inefficient and highly risky in the complex network environment. Batch auditing [6] is an efficient auditing technique, which enables the TPA to complete multiple data auditing tasks with only one interaction with the CSP. When a TPA needs to audit the data integrity of multiple DOs, it will first prepare a set of challenges and send them to the CSP. After receiving these challenges, the CSP will generate and send a batch proof to the TPA. Passing the inspection of the TPA means that the integrity of the data associated with the set of challenges has not been compromised. In this way, the interaction between the TPA and the CSP is only once.

The initial remote data integrity batch auditing (RDIBA) protocols [6, 7] are based on a traditional public key cryptosystem. Although this cryptosystem has been broadly applied in practice, it still faces the problem of public key certificate management, which brings heavy burden to the system. In order to avoid this weakness, an identity-based cryptosystem (IBC) [8] is used to build RDIBA protocols. However, key escrow is an intrinsic defect of the IBC. Therefore, the identity-based RDIBA protocols have certain limitations which are only applied to small, closed systems rather than to RDIA. A certificateless cryptosystem (CLC) [9] is beneficial to address the weaknesses of the public key certificate management and key escrow, simultaneously. Nevertheless, it is necessary for the CLC to construct a costly, secure, and secret channel for each DO to send the partial private key, which is tricky. In order to overcome the shortcomings, we use a certificate-based cryptosystem (CBC) first introduced in [10] to form our protocol. The CBC requires no certificate management, key escrow, secure channels, and a fully trusted authorization, and hence it can be easily deployed in the public environment of cloud-IoT. In recent years, a number of RDIBA protocols have been proposed. Unfortunately, the vast majority of them cannot prevent collusion attacks. It means that the CSP can be tricked by multiple DOs such that the DOs can claim compensations from the CSP.

1.2. Our Contributions. In this paper, a security-enhanced certificate-based remote data integrity batch auditing protocol that can resist the highest level of collusion attacks is proposed. The main contributions of our paper can be summarized as follows:

(1) We propose the formal definition and security model of the certificate-based remote data integrity batch auditing (CBRDIBA) protocol according to cloud-IoT.
(2) We give a secure protocol that can prevent the current highest level of collusion attacks. And we offer four games to analyse the security of our protocol in the random oracle model.
(3) We compare batch auditing with single auditing in terms of the communication cost and the computation cost in our protocol in theory, and simulate their performance through experiments by Java pairing-based cryptography library. The simulation results illustrate that, although our protocol is security-enhanced, our batch auditing protocol remains computationally efficient and practical.

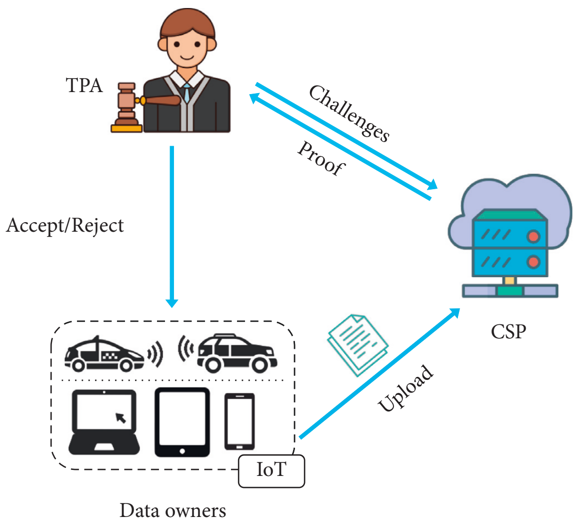

Figure 1: A data integrity auditing model for cloud-IoT.

1.3. The Organization of the Rest Paper. The rest of our paper is structured as follows. In Section 2, we review some previous work associated with the CBRDIBA protocol. Next, in Section 3, we give some preliminaries used as the basis for the CBRDIBA protocol and present the problem formulations including the system model, an overview of the CBRDIBA protocol, and the security model. Then, we demonstrate the concrete construction of the CBRDIBA protocol in section 4, and analyze the properties of the CBRDIBA protocol in Section 5. In Section 6, we demonstrate the superiority of the CBRDIBA protocol through theoretical analysis and experiments. Finally, we draw conclusions in Section 7.

2. Related Work

In the cloud-IoT environment, RDIA provides a fundamental solution to audit the integrity of data according to homomorphic verifiable tags (HVTs), which are some homomorphic signatures of the data blocks. The concept of homomorphic signature was first proposed by Rivest in 2000 [11]. It can be divided into linear homomorphic signature, polynomial function homomorphic signature, and fully homomorphic signature. In 2007, Zhao et al. [12] proposed the first linear homomorphic signature (LHS) scheme that allows the arbitrary linear combination of the signature, which can be used to easily audit the integrity of the received data. However, their scheme has been proven to be unsafe and impractical. In the same year, Ateniese et al. [13] first presented a provable data possession model and initially introduced the technique of probabilistic integrity checking for the remote data.

In 2009, Boneh et al. [5] proposed a LHS scheme that gives the first formal definition of the LHS scheme. In the following years, a large number of LHS schemes have been proposed, which have been further improved in terms of efficiency, privacy, and security. In 2015, Yu et al. [14] suggested an identity-based LHS scheme. The disadvantages of using public key certificates are avoided by adding the identity-based feature. In 2018, Li et al. [15] used a certificateless LHS signature scheme to construct a certificateless public data integrity auditing protocol for data shared among a group. In 2021, Li et al. [16] constructed a data integrity auditing protocol for cloud-assisted wireless body area networks using certificate-based LHS. Their protocol adds timestamps to the HVTs such that adversaries cannot use the expired valid proof to pretend to be the current ones. In 2022, Li et al. [17] introduced a concept of transparent integrity auditing which can keep the CS from misbehaving (i.e., procrastinating auditing).

All the above LHS schemes are constructed in single-user scenarios, and all the signatures entered in the algorithm are produced using the same private key. In real life, there are many application scenarios with multi-users. In these scenarios, a newly generated signature is aggregated from signatures generated by different DOs using their private keys. For these scenarios, an aggregate signature (AS) scheme can be used. The first AS scheme and its formal definition were proposed by [18]. As time went by, identity-based AS [19], certificateless AS [20], and certificate-based AS [21] were gradually proposed. In AS, a problem worth paying attention to is the fully chosen-key attacks [22], which were first proposed by Wu et al. in 2019. The fully chosen-key attacks are currently the most difficult collusion attacks to defend against.

As time progressed, AS were gradually combined with LHS. The first linear homomorphic aggregate signature (LHAS) was put forward by Jing [23] in 2014, which supports linear operations on binary domains. Its security is based on the small integer solution problem [24]. In 2018, Han et al. [25] proposed an efficient error search technique called Lucas search, which is based on the Lucas sequence [26], to efficiently search the corrupted data files once the batch auditing task fails. In 2019, Wang et al. [27] used a certificate-based cryptosystem to construct a data integrity auditing protocol in which they mentioned that their scheme can perform batch auditing. Although their protocol can perform batch auditing and has used a CBC, it cannot prevent collusion attacks. In 2019, Yang et al. [28] constructed a data integrity batch auditing protocol that can work in multi-cloud storage. In 2020, Huang et al. [29] applied the CLC to RDIBA and proposed a certificateless remote data integrity batch auditing protocol. However, most of the above protocols are not resistant to collusion attacks let alone the fully chosen-key attacks.

3. System Model and Objectives

3.1. Preliminaries

Definition 1 (Bilinear Pairing). Let $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ denote three multiplicative groups of the same order q. Let g_1 be a generator of $\mathbb{G}_1$ and let g_2 be a generator of $\mathbb{G}_2$. e: $\mathbb{G}_1 \times \mathbb{G}_2 \longrightarrow \mathbb{G}_T$ is a bilinear mapping with following properties:

(1) Bilinearity: $e(g_1^{a_1}, g_2^{a_2}) = e(g_1, g_2)^{a_1 a_2}$ for any $a_1, a_2 \in \mathbb{Z}_q$ and $g_1 \in \mathbb{G}_1, g_2 \in \mathbb{G}_2$.

(2) Non-degeneracy: if g_1 is a generator of $\mathbb{G}_1$ and g_2 is a generator of $\mathbb{G}_2$, then $e(g_1, g_2)$ is a generator of $\mathbb{G}_T$.

(3) Computability: for any $g_1 \in \mathbb{G}_1$ and $g_2 \in \mathbb{G}_2$, $e(g_1, g_2)$ is a generator of $\mathbb{G}_T$.

3.1.1. Co-Computational Diffie-Hellman Problem (co-CDH). Let $\mathbb{G}_1, \mathbb{G}_2$ be two cyclic groups of the same prime order q. g_1 is a generator of $\mathbb{G}_1$ and g_2 is a generator of $\mathbb{G}_2$; for random $a \in \mathbb{Z}_q^*$, given g_1, g_2, g_2^a, compute g_1^a.

3.1.2. Co-CDH Problem Assumption. We say that the co-CDH assumption holds in $\mathbb{G}_1$ and $\mathbb{G}_2$ if the advantage is negligible in solving the co-CDH problem for any probabilistic polynomial-time algorithm.

3.1.3. k-Collusion Attack Algorithm Problem (k-CAA). Suppose $\mathbb{G}_1$ and $\mathbb{G}_2$ are two cyclic groups with sizable prime q, $a, (h_1, \ldots, h_k) \in \mathbb{Z}_q^*$ are $k+1$ integers. Given $g_1 \in \mathbb{G}_1, g_i^a \in \mathbb{G}_i (i = 1, 2)$, and k pairs $(h_1, g_1{}^{(a+h_1)^{-1}}), \ldots, (h_k, g_1{}^{(a+h_k)^{-1}})$ output a new pair $(h^*, g_1{}^{(a+h^*)^{-1}})$ for some $h^* \notin (h_1, \ldots, h_k)$.

3.1.4. k-CAA Problem Assumption. We say that the k-CAA assumption holds in $\mathbb{G}_1$ and $\mathbb{G}_2$ if the advantage is negligible in solving the k-CAA problem for any probabilistic polynomial-time algorithm.

3.2. The RDIBA System Model. We raise a RDIBA system where the TPA validates timestamps and the integrity of multiple datasets. The system is composed of three kinds of entities: data owners (DOs), cloud service provider (CSP), and third-party auditor (TPA). As the clients of the CSP, the DOs upload their datasets to the CSP and then delete them locally. The DOs are foxy. Two or more data owners may negotiate with each other to generate tags for their own data blocks using each other's private keys. In this case, the TPA will respond "accept," even if the single auditing information is not right. After that, the data owners will extort compensation from the CSP. The CSP is honest but curious. Although the CSP will honestly generate proofs, as the certification authority of the system, it can embed trapdoors in the system parameters to achieve the purpose of forging a valid signature without owning the user's private key. As the system user, the TPA can have access to the datasets and validate the data integrity. Figure 2 shows the system model and the process is elaborated as follows:

(1) Every DO generates timestamps and tags for data blocks that come from the collected dataset. The DOs then upload these data blocks with the related auditing information (such as all tags of data blocks, the label of the dataset, timestamp, and the number of data blocks) to the CS and delete them locally.

(2) The TPA can obtain the auditing information from the CSP, and then generate and send one or more challenges to the CSP. On receiving a challenge or a set of challenges, the CSP will generate a proof or a batch proof and then return it to the TPA.

(3) The TPA will validate the authenticity and correctness of the proof or the batch proof and output "accept" or "reject." If the proof or the batch proof passes the verification, the corresponding one or multiple datasets stored in the CSP are considered to be secure.

When a TPA issues multiple challenges, the CSP only needs to interact with the TPA once because the TPA validates batch proof, which means it validates all corresponding proofs. There is no doubt that a lot of data transmission loss is saved.

3.2.1. System Components. A RDIBA system model Ω = (System Setup, User Registration, Outsourcing Storage, Auditing, Batch Auditing) is an interactive protocol allowing a third party (i.e., TPA) to validate that files are stored truthfully.

(i) System Setup: The CSP initializes a data integrity auditing system. After generating the system master private key msk, the CSP publishes the system parameters params.

(ii) User Registration: The DO generates public/private key pair $(\mathrm{upk}_{\mathrm{ID}}, \mathrm{usk}_{\mathrm{ID}})$ and provides the CSP with its public key $\mathrm{upk}_{\mathrm{ID}}$. The CSP returns a verifiable certificate $\mathrm{Cert}_{\mathrm{ID}}$ to the TPA.

(iii) Outsourcing Storage: On receiving an encoded dataset $F = (m_1, \ldots, m_k)$ named ds and a timestamp t, the DO uploads the encoded dataset F, label τ, and tags $\{\sigma_i\}$ of data blocks to the CS.

(iv) Auditing: The TPA provides the CSP with a challenge chal by sampling some data blocks. And then according to the challenge, the CSP generates a proof PF and returns it to the TPA. Next, on receiving a valid proof PF, the TPA outputs "accept." Otherwise, it outputs "reject." Outputting "Accept" also means that the dataset is stored in the CS intactly and honestly.

(v) Batch Auditing: The TPA provides the CSP with challenges $\mathrm{chal}_z, z = (1, \ldots, l), \quad l \leq L$ (As shown in Figure 2, L indicates the total number of data owners in the system) by sampling some data blocks. And then according to the challenges, the CSP generates a batch proof and returns it to the TPA. Next, on receiving a valid proof PF, the TPA outputs "accept." Otherwise, it outputs "reject." Outputting "Accept" also means that multiple corresponding datasets are stored in the CS intactly and honestly.

3.3. Design Objectives. Our protocol is designed to achieve the following objectives:

(i) Correctness of Auditing: The CSP cannot generate the valid proofs without preserving the entire original dataset and timestamp.

(ii) Correctness of Batch Auditing: The batch proof is valid, if and only if every individual proof used in the Batch Proof Generation algorithm is valid.

(iii) Batch Auditing: The TPA can perform batch auditing. Note that batch auditing means that the TPA only needs to interact with the CSP once.

(iv) Anonymity: In our protocol, only the DO and the CSP can know the real identity RID of the DO. Any other entity like a TPA can only know the pseudonym ID of the DO.

(v) Verifiability: The TPA can validate the integrity of files by sampling some random data blocks with no access to the entire files.

(vi) Security: To ensure the correctness and integrity of the data, a secure CBRDIBA protocol should resist the following attacks: (1) The system user (or CSP) can forge a tag of a data block. (2) The CSP can replace the new challenge response with the expired valid proof to deceive the DO. (3) Two or more data owners can use each other's private key to forge invalid tags which cannot pass the verification, but their batch proof can pass the batch proof verification.

The security model can be defined using the following four games between adversaries $\mathscr{A}_I$, $\mathscr{A}_{II}$, $\mathscr{A}_{III}$, $\mathscr{A}_{IV}$ and challenger $\mathscr{B}$. The goal of the adversaries $\mathscr{A}_I$, $\mathscr{A}_{II}$, and $\mathscr{A}_{III}$ is to forge a valid single proof. $\mathscr{A}_{IV}$ aims to forge a valid batch proof.

Among them, $\mathscr{A}_I$ is an outside attacker (hostile DO) who wants to forge a tag of data block in order to outsmart the TPA. $\mathscr{A}_{II}$ is an inside attacker (honest but curious CSP) who is more inclined to defend itself when mistakes occur. It can hold some system information (such as the system master private key) that the DO does not have.

We can go to the literature [16] and read the attack models (Game I and Game II).

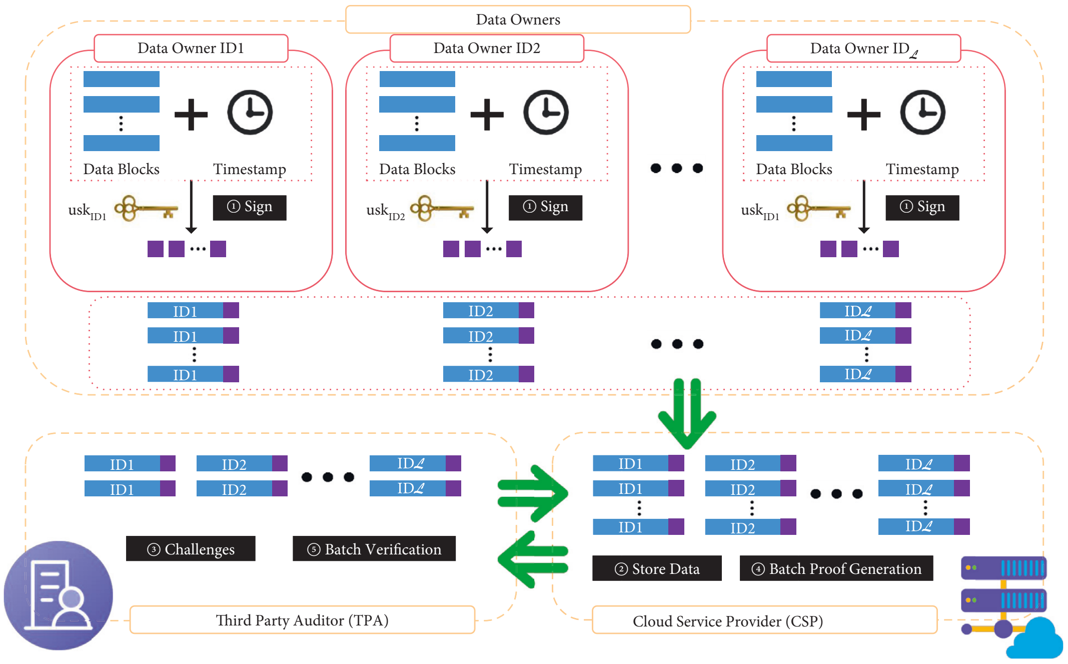

Figure 2: A remote data integrity batch auditing system model.

$\mathscr{A}_{III}$ models the ability of the CSP to forge a valid proof, and attempts to generate a valid proof when some data blocks are damaged. The adversary game of $\mathscr{A}_{III}$ is as follows.

Game III (Type III adversary $\mathscr{A}_{III}$):

(i) System Initialization: On inputting a security parameter 1^λ, $\mathscr{B}$ runs System Setup to generate public parameters params and system master private key msk, and sends params to $\mathscr{A}_{III}$.

(ii) Oracle Simulation: $\mathscr{A}_{III}$ can adaptively make TagGen Queries and ProofCheck Queries.

(iii) Challenge: $\mathscr{B}$ generates a challenge chal and sends it to $\mathscr{A}_{III}$. On receiving chal, $\mathscr{A}_{III}$ generates a proof PF and returns it to $\mathscr{B}$.

(iv) Forge: $\mathscr{A}_{III}$ outputs $(ID^*, upk_{ID^*}, \tau^*, chal^*, PF^*)$. $\mathscr{A}_{III}$ wins the game if the following conditions are satisfied:

(1) Proof $PF = (m^*, \sigma^*)$ can pass the algorithm Proof Verification, where $m^* \notin F$.

(2) There is at least one challenged data block that has never been issued a TagGen Queries.

In our protocol, we consider the DO can be dodgy and that two or more data owners may cheat the TPA and the CSP for profit. We simulate these data owners as adversary $\mathscr{A}_{IV}$. The purpose of $\mathscr{A}_{IV}$ is to use a set of single proofs to generate a valid batch proof with at least one invalid single proof. The adversary $\mathscr{A}_{IV}$ can do the fully chosen-key attacks. Now we revisit the security model in [22] through the following game between an adversary $\mathscr{A}_{IV}$ and a challenger $\mathscr{B}$. The Game IV consists of three steps: System Initialization, Oracle Simulation, and Forge. The adversary game of $\mathscr{A}_{IV}$ is as follows.

Game IV (Type IV adversary $\mathscr{A}_{IV}$):

System Initialization: Inputting the security parameter λ, the challenger $\mathscr{B}$ generates the system parameters params. Furthermore, $\mathscr{B}$ randomly generates the public-secret key pair (upk_{ver}, usk_{ver}) for the TPA, and then $\mathscr{B}$ gives $\mathscr{A}_{IV}$ the params and upk_{ver}.

Oracle Simulation: $\mathscr{A}_{IV}$ can access the following queries:

(i) Corruption Queries: $\mathscr{A}_{IV}$ requests such a query, and $\mathscr{B}$ generates the key pairs (upk_{ID_z}, usk_{ID_z}) by running the algorithm UserkeyGen, and then returns usk_{ID_z} to $\mathscr{A}_{IV}$.

(ii) Batch Proof Check Queries: The simulator $\mathscr{B}$ generates some challenges and sends the challenges to $\mathscr{A}_{IV}$. Then, on receiving the challenges from $\mathscr{B}$, $\mathscr{A}_{IV}$ generates a batch proof BPF and returns it to $\mathscr{B}$. Finally, $\mathscr{B}$ verifies BPF by running the algorithm Batch Proof Verification and gives the result to $\mathscr{A}_{IV}$.

Forge: Finally, $\mathscr{A}_{IV}$ outputs its forgery. If the following two conditions are satisfied, $\mathscr{A}_{IV}$ wins the game:

(1) The batch proof is generated from all the single proofs.

(2) The batch proof σ^* is valid.

(3) At least one single proof of the batch proof is invalid.

Definition 2. A CBRDIBA protocol is secure if the advantage of the adversaries $\mathscr{A}_I$, $\mathscr{A}_{II}$, $\mathscr{A}_{III}$, and $\mathscr{A}_{IV}$ winning game I, game II, game III, and game IV in probabilistic polynomial time (PPT) is negligible, respectively.

4. A CBRDIBA Protocol

Based on the construction of homomorphic verifiable tags in [16], we design a security-enhanced CBRDIBA protocol which is composed of five procedures: System Setup, User Registration, Outsourcing Storage, Auditing, and Batch Auditing. The notations used in this section are shown in Table 1.

4.1. System Setup. Given a security parameter λ, the CSP does:

(i) Generate a bilinear group $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, q, e)$, where $q > 2^{\lambda}$, $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ are three multiplicative cyclic groups with the same prime order q. Bilinear pairing $e\colon \mathbb{G}_1 \times \mathbb{G}_2 \longrightarrow \mathbb{G}_T$. Isomorphic mapping $\psi\colon \mathbb{G}_2 \longrightarrow \mathbb{G}_1$.

(ii) Generate a generator g_1 in $\mathbb{G}_1$ and a generator g_2 in $\mathbb{G}_2$.

(iii) Let $H_0\colon \{0,1\}^* \longrightarrow \mathbb{Z}_q^*$, $H_1\colon \{0,1\}^* \times \mathbb{G}_2 \longrightarrow \mathbb{G}_1$, $H_2\colon \{0,1\}^* \times \{0,1\}^* \times \mathbb{G}_1 \longrightarrow \mathbb{Z}_q^*$, $H_3\colon \{0,1\}^* \times \mathbb{Z}_n^* \times \mathbb{G}_1 \longrightarrow \mathbb{Z}_q^*$, and $H_4\colon \mathbb{G}_2 \longrightarrow \{0,1\}^*$ be five collision resistant hash functions.

(iv) Randomly choose $s \in \mathbb{Z}_q^*$ as the master private key, and the master public key is $mpk = g_2^s$.

The CSP publishes the system parameters params $= \{q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, \psi, g_1, g_2, mpk, H_0, H_1, H_2, H_3, H_4\}$ and keeps the system master private key $msk = s$ secret.

4.2. User Registration. The user registration process is composed of three phases.

(i) Phase 1 (UserKeyGen): Given the system parameters params and the user's real identity $RI\ D$, the user does:

(1) Randomly choose $x \in \mathbb{Z}_q^*$ as the user's secret key which is denoted by $usk_{I\ D}$. Generate the user's pseudonym by computing $I\ D = RI\ D \oplus H_4 (mpk^{usk_{I\ D}})$.

(2) Compute $upk_{I\ D} = g_2^x$ as the user's public key.

(ii) Phase 2 (Certify): Given the system parameter params, master private key msk, the user's public key $upk_{I\ D}$ with $I\ D$, the CSP generates the certificate $Cert_{I\ D} = H_1 (I\ D, upk_{I\ D})^s$ for the user $I\ D$ and sends it to the user.

(iii) Phase 3 (Authentication): The user $I\ D$ validates the certificate. If the equation

$$e(Cert_{I\ D}, g_2) = e(H_1(I\ D, upk_{I\ D}), mpk), \quad (1)$$

holds, then the user succeeds in logging into the system.

4.3. OutsourcingStorage. The outsourcing storage procedure is composed of two phases: Data Processing and Data Upload.

Table 1: Notations and descriptions.

Notation	Descriptions
λ	Security parameter
PF	A single proof
BPF	A batch proof
$chal_z$	The z-th challenge at one auditing
I	A subset selected by the TPA
F	A dataset
ds	The name of a dataset
τ	The label of a dataset
$m_i, i \in [1,k]$	The data blocks of one dataset
$\sigma_i, i \in [1,k]$	The tags of related data blocks in a dataset
t	A timestamp
ID_z	The pseudonym of user z
RID_z	The real identity of user z
upk_{ver}	The public key of the TPA
usk_{ver}	The private key of the TPA

(i) Phase 1 (Data Processing):

(1) Given an encrypted dataset $F \in (0,1)^*$ named ds, the DO splits the dataset into k data blocks $m_1, m_2, \ldots, m_k$, and each block comprises n sectors, i.e., $F = (m_{i,j})_{k \times n}$

(2) The DO randomly chooses $\alpha \in \mathbb{Z}_q^*$, and then computes $\beta = g_2^{\alpha}$ and $\tau_0 = Cert_{I\ D}{}^{1/usk_{I\ D} + H_2(I\ D, ds\ , t, \beta)}$. The DO sets $\tau = \{\tau_0, \beta, ds\ , t, k, n\}$ as the label of the dataset.

(3) For each $m_i, i \in [1,k]$, the DO computes tags $\sigma_1, \ldots, \sigma_k$ with

$$\sigma_i = \left(g_1^{\sum_{j=1}^{n} H_3(I\ D, j, \tau_0) m_{ij}} \right)^{\alpha}. \quad (2)$$

(ii) Phase 2 (Data Upload): After the above phase is completed, the DO uploads the label τ, all data blocks $m_1, \ldots, m_k$, and the corresponding tags $\sigma_1, \ldots, \sigma_k$ to the CSP. Meanwhile, the DO only stores the label τ of the dataset and deletes the data blocks and the corresponding tags locally.

4.4. Auditing. The auditing procedure is composed of three phases: Challenge, Proof Generation, and Proof verification.

(i) Phase 1 (Challenge): The TPA can get the label and send the challenge to the CSP for checking the data integrity. Firstly, the TPA randomly chooses a nonempty subset $I \subseteq [1,k]$ along with random values $c_i \in \mathbb{Z}_q^*$ for every $i \in I$. After that, the TPA sends the user identity $I\ D$, the identifier of dataset ds, and the challenge $chal = \{I\ D, \text{d}s, (i, c_i)\colon i \in I\}$ to the CSP.

(ii) Phase 2 (Proof Generation): The CSP will honestly generate the proof according to the challenge. On receiving a user identity $I\ D$, the identifier of dataset ds, and a challenge $chal$, the CSP computes $m = \sum_{i \in I} c_i m_i$, $\sigma = \prod_{i \in I} \sigma_i^{c_i}$, and returns $PF = \sigma$ and m to the TPA.

(iii) Phase 3 (Proof Verification): Firstly, on receiving the proof PF and corresponding m from the CSP, the TPA can check validity of label τ by

$$e\left(\tau_0, upk_{ID} \cdot g_2^{H_2(ID,\mathrm{ds},t,\beta)}\right) = e\left(H_1\left(ID, upk_{ID}\right), mpk\right). \quad (3)$$

And then, if the above equation holds, the TPA tests

$$e\left(\sigma, g_2\right) = e\left(g_1^{\sum_{j=1}^{n} H_3\left(ID,j,\tau_0\right) m_j}, \beta\right). \quad (4)$$

In the end, if the above equation holds, the TPA thinks the dataset stored in the CSP is unwounded. Otherwise, the dataset has been damaged.

4.5. Batch Auditing. The TPA can improve the efficiency of auditing through batch auditing. The batch auditing procedure is composed of four phases: Challenges, Single Proof Generation, Batch Proof Generation, and Batch Proof verification. We describe the batch auditing process as follows.

(i) Phase 1 (Challenges): The TPA wants to verify the integrity of multiple data once. Firstly, the TPA randomly chooses $s_{ver} \in Z_q^*$ as its private key usk_{ver}, and computes $upk_{ver} = g_2^{s_{ver}}$ as its public key. The registration procedure of the TPA is similar to DO's registration procedure. However, for security and actual needs, the TPA cannot hide its real identity. Then, the TPA gets labels and sends some challenges to the CSP. It randomly selects l nonempty subsets $I_z \subseteq [1, k_z], z = 1, \ldots, l$ along with random values $c_{zi} \in \mathbb{Z}_q^*$ for every $i \in I_z$. After that, the TPA prepares the user identity ID_z, the identifier of dataset ds_z, and the challenge $chal_z = \{ID_z, ds_z, (i, c_{zi}): i \in I_z\}$, where $z = 1, \ldots, l$ is the serial number of $lchals$. Note that for each user, the number and location of the challenged blocks may be different. The TPA sends them and its public key upk_{ver} to the CSP with no secure channels and keeps its private key usk_{ver} secret.

(ii) Phase 2 (Single Proof Generation): Same as Proof Generation 4.4, on receiving user identities ID_z, the identifier of dataset ds_z, and a challenge $chal_z$, the CSP computes $m_z = \sum_{i\in I_z} c_{zi} m_{zi}$, $\sigma_z = \prod_{i\in I_z} \sigma_{zi}^{c_{zi}}$, where $z = 1, \ldots, l$.

(iii) Phase 3 (Batch Proof Generation): After generating every single proof $\sigma_z, z = 1, \ldots, l$, the CSP computes r with the public key of the TPA as follows:

$$r = H_0\left(e\left(\sigma_1, upk_{ver}\right), \ldots, e\left(\sigma_l, upk_{ver}\right)\right), \quad (5)$$

and then, the CSP computes the batch proof $\widetilde{\sigma}$ as follows:

$$\widetilde{\sigma} = \left(\prod_{z=1}^{l} \sigma_z\right)^r. \quad (6)$$

The CSP returns $BPF = \widetilde{\sigma}$ and $m_1, \ldots, m_l$ to the TPA.

(iv) Phase 4 (Batch Proof Verification): Firstly, on receiving a response BPF and corresponding $m_z, z = 1, \ldots, l$ from the CSP, the TPA validates the validity of the corresponding labels of multiple datasets $\tau_z, z = 1, \ldots, l$ by

$$e\left(\prod_{z=1}^{l} H_1\left(ID_z, upk_{ID_z}\right), mpk\right) = \prod_{z=1}^{l} e\left(\tau_{0z}, upk_{ID_z} \cdot g_2^{H_2\left(ID_z, ds_z, t_z, \beta_z\right)}\right). \quad (7)$$

Next, if the above equation holds, the TPA computes

$$r' = H_0\left(e\left(\left(g_1^{\sum_{j=1}^{n} H_3\left(ID_1,j,\tau_{01}\right)\cdot m_{1j}}\right)^{s_{ver}}, \beta_1\right), \ldots, e\left(\left(g_1^{\sum_{j=1}^{n} H_3\left(ID_l,j,\tau_{0l}\right)\cdot m_{lj}}\right)^{s_{ver}}, \beta_l\right)\right), \quad (8)$$

note that s_{ver} is the private key of the TPA.

And then, the TPA tests

$$e\left(\widetilde{\sigma}, g_2\right) = \prod_{z=1}^{l} e\left(g_1^{\sum_{j=1}^{n} H_3\left(ID_z,j,\tau_{oz}\right) m_{zj}}, \beta_z^{r'}\right), \quad (9)$$

where

$$m_z = \prod_{i\in I} c_{zi} m_{zi}, m_z = \left(m_{z1}, \ldots, m_{zn}\right), z = 1, \ldots, l. \quad (10)$$

If all the above equations hold, the TPA thinks multiple corresponding datasets stored in the CSP are unwounded. Otherwise, one or more datasets have been damaged, and then we can use Lucas search to efficiently identify which datasets have been breached. The Lucas search is divided into two cases according to the amount of DOs. Please refer to [25] for details.

5. Properties' Analysis

5.1. Correctness

5.1.1. Correctness of Auditing. In procedure 4.4, the proof of dataset includes the label τ of the dataset and the tags σ of data blocks. Furthermore, every tag is generated by the Data Processing algorithm, and the single proof is generated by the Proof Generation algorithm. All of them are equivalent, i.e., the Proof Verification algorithm can verify both of them. Assuming that all entities operate honestly with the algorithms described above, then the correctness of the scheme can be verified from two aspects:

(1) For any file label τ and for any single proof $PF = (m, \sigma)$, we have:

$$\begin{aligned}\sigma &= \prod_{i\in I} \sigma_j^{c_j} = \prod_{i\in I}\left(\left(g_1^{\sum_{j=1}^{n} H_3(I\,D,j,\tau_0)m_{ij}}\right)^{\alpha}\right)^{c_i}\\ &= \left(g_1^{\sum_{j=1}^{n} H_3(I\,D,j,\tau_0)\sum_{i\in I} c_i m_{ij}}\right)^{\alpha} = \left(g_1^{\sum_{j=1}^{n} H_3(I\,D,j,\tau_0)m_j}\right)^{\alpha}.\end{aligned} \tag{11}$$

(2) For any file label $\tau = \{\tau_0, \beta, \mathrm{d}s, t, k, n\}$ and for any data block $m_i \in F$ with corresponding tag σ_i, we have:

$$e(\sigma_i, g_2) = e\left(\left(g_1^{\sum_{j=1}^{n} H_3(I\,D,j,\tau_0)m_{ij}}\right)^{\alpha}, g_2\right) = e\left(g_1^{\sum_{j=1}^{n} H_3(I\,D,j,\tau_0)m_{ij}}, \beta\right). \tag{12}$$

5.1.2. Correctness of Batch Auditing. If the single proof σ_z is generated by the DO ID_z directly with public key $upk_{I\,D\,z}$, then the following equations hold for $z = 1, \ldots, l$:

$$e(\sigma_z, upk_{ver}) = e\left(\prod_{j=1}^{n} g_1^{\,H_3(ID_z,j,\tau_{0z})\cdot m_{zj}}, \beta_z^{s_{ver}}\right), \tag{13}$$

so,

$$\begin{aligned} r &= H_0\left(e(\widetilde{\sigma}_1, upk_{ver}), \ldots, e(\widetilde{\sigma}_l, upk_{ver})\right)\\ &= H_0\left(e\left(\left(g_1^{\sum_{j=1}^{n} H_3(ID_1,j,\tau_{01})\cdot m_{1j}}\right)^{s_{ver}}, \beta_1\right), \ldots, e\left(\left(g_1^{\sum_{j=1}^{n} H_3(ID_1,j,\tau_{01})\cdot m_{lj}}\right)^{s_{ver}}, \beta_l\right)\right) = r',\end{aligned} \tag{14}$$

and then

$$\begin{aligned} e(\widetilde{\sigma}, g_2) &= e\left(\left(\prod_{z=1}^{l} \sigma_z\right)^{r}, g_2\right)\\ &= \prod_{z=1}^{l} e(\sigma_z, g_2^{r}) = \prod_{z=1}^{l} e\left(g_1^{\sum_{j=1}^{n} H_3(ID_z,j,\tau_{oz})m_{zj}}, \beta_z^{r'}\right).\end{aligned} \tag{15}$$

And for the label of dataset $\mathrm{d}s_z, z = 1, \ldots, l$, we have

$$e\left(\prod_{z=1}^{l} H_1\left(ID_z, upk_{ID_z}\right), mpk\right) = \prod_{z=1}^{l} e\left(H_1\left(ID_z, upk_{ID_z}\right), mpk\right) = \prod_{z=1}^{l} e\left(\tau_{oz}, upk_{ID_z} g_2^{H_2\left(ID_z, ds_z, t_z, \beta_z\right)}\right). \quad (16)$$

5.2. Batch Auditing. If the TPA needs to audit the integrity of multiple data blocks simultaneously, it can issue valid challenges according to ds_z and k_z, where $z = 1, \ldots, l$. When the CSP receives multiple challenges from the same TPA, the CSP generates a batch proof and sends it to the TPA. Batch proof allows the TPA to interact with the CSP only once to verify that the integrity of the stored data that have not been corrupted.

5.3. Anonymity. In our protocol, the user's real name $RI\ D$ is protected by a pseudonym $I\ D = RI\ D \oplus H_4(mpk^{usk_{I\ D}})$. Every entity can know the mpk and the $upk_{I\ D}$, but they cannot get the knowledge of msk or $usk_{I\ D}$. We have $H_4(mpk^{usk_{I\ D}}) = H_4(upk_{I\ D}^{msk})$. Apart from the user with real identity $RI\ D$ and the CSP, any other third entity cannot know the $RI\ D$ of the user because of the secrecy of msk, $usk_{I\ D}$, and$RI\ D$, and the unipolarity of the hash function H_4.

5.4. Security

5.4.1. The Security of Single Data Auditing. In our certificate-based remote data integrity auditing (CBRDIA, not including the Batch Auditing procedure) protocol, we consider three types of PPT adversaries. Type I adversary ($\mathscr{A}_I$) models an attacker who can replace the user's public key. Type II adversary ($\mathscr{A}_{II}$) models the honest but curious CSP who holds the master secret key and is not allowed to replace the target user's public key. $\mathscr{A}_I$ and $\mathscr{A}_{II}$ cannot hold both user's private key and certificate. Type III adversary ($\mathscr{A}_{III}$) models the ability of the CSP to forge a valid proof. We conclude the security of the single data auditing procedure in our protocol by Theorems 1–3.

Theorem 1. *Suppose a PPT adversary $\mathscr{A}_I$ can forge a valid proof with advantage ϵ, and suppose $\mathscr{A}_I$ can make at most q_u times Create User Queries, q_e times Certification Queries, and q_t times TagGen Queries. Then there exists a challenger $\mathscr{B}$ to solve the co-CDH problem with advantage $\epsilon' \geq (1 - 1/q_u)^{q_e} (1 - 1/q_t + 1)^{q_t} 1/(q_t + 1) q_u \epsilon$.*

Proof. The detailed proof is given in *A*. □

Theorem 2. *Suppose a PPT adversary $\mathscr{A}_{II}$ can forge a valid proof with advantage ϵ, and suppose $\mathscr{A}_{II}$ can make at most q_u times Create User Queries, q_r times Collusion Queries, and q_t times TagGen Queries. Then there exists a challenger $\mathscr{B}$ to solve the k-CAA problem with advantage $\epsilon' \geq (1 - 1/q_u)^{q_r} (1 - 1/q_t + 1)^{q_t} 1/(q_t + 1) q_u \epsilon$.*

Proof. The detailed proof is given in *B*. □

Theorem 3. *The probability of $\mathscr{B}$ forging a valid single proof is negligible if the file challenged is damaged or modified.*

Proof. The detailed proof is given in *C*. □

5.4.2. The Security of Batch Auditing

Theorem 4. *Suppose H_0 is a collision-resistant Hash function. The batch proof in the CBRDIBA protocol is valid if and only if every individual proof used in the Batch Proof Generation algorithm is valid.*

Proof. The detailed proof is given in *D*. □

6. Performance Evaluation

Before analyzing the performance of our CBRDIBA protocol, we first compared our CBRDIBA protocol with several data integrity auditing protocols [13, 16, 25, 27–29] in Table 2, and we notice that the CBRDIBA protocol supports batch auditing, resisting collusion attacks, anonymity, Lucas error search, with no key escrow problem, and with no secure channels simultaneously. The protocol uses Lucas search to deal with batch auditing failures, which is a more efficient error search method than binary search. And the protocol also adds timestamps to the HVTs such that it can make adversaries not use the expired valid proof to pretend to be the current ones.

Furthermore, we will demonstrate the efficiency of our data integrity batch auditing scheme in comparison with data integrity auditing of a single DO both theoretically and experimentally. Table 3 shows the notations used in this section.

6.1. Theoretical Analysis

6.1.1. Communication Cost. From Table 4 we can see that the communication cost of Batch Auditing with l different DOs is lower than the cost of l times Auditing procedures.

Note that as the number of DOs involved in a batch auditing procedure increases, the number of interactions between the CSP and the TPA is constant. However, there is no doubt that in the Auditing procedure, the number of DOs is directly proportional to the number of interactions.

6.1.2. Computation Cost. In Table 4, we list the computation and communication costs of the Auditing procedure and the Batch Auditing procedure. Note that, in order to facilitate to compare, we ignore the time cost of map to Z_q hash operation, additive operation in Z_q, and inverse operation in Z_q. The comparison results show that the computation cost of batch auditing with l different DOs is $(3l + 2)T_p + lT_{H_1} + lT_{M_1} + lT_{M_2} + 2lT_{E_1} + 2lT_{E_2}$, which is

Table 2: Comparison of multiple features.

Feature	Ateniese et al. [13]	Li et al. [16]	Yang et al. [28]	Han et al. [25]	Huang et al. [29]	Wang et al. [27]	CBRDIBA
Can do batch auditing	No	No	Yes	Yes	Yes	Yes	Yes
Can resist collusion attacks	—	—	No	No	No	No	Yes
Lucas error search	—	—	No	Yes	No	No	Yes
Anonymity	No	No	No	No	No	No	Yes
No key escrow problem	No	Yes	Yes	No	Yes	Yes	Yes
No secure channel	No	Yes	Yes	No	No	Yes	Yes
Add timestamp	No	Yes	No	No	No	No	Yes

Table 3: The notations of operations.

Notations	Meanings
T_H	Map-to-point hash operation
T_P	Bilinear pairing operation
T_{E_1}	Exponential operation in $\mathbb{G}_1$
T_{E_2}	Exponential operation in $\mathbb{G}_2$
T_{M_1}	Multiplicative operation in $\mathbb{G}_1$
T_{M_2}	Multiplicative operation in $\mathbb{G}_2$
n	The number of data block's sectors
c	The number of challenged data blocks
$\lvert Z_q \rvert$	The binary length of an element in $\mathbb{Z}_q$
$\lvert G_1 \rvert$	The binary length of an element in $\mathbb{G}_1$
l	The number of proofs used in the batch proof

Table 4: Comparison of the computation cost and the communication cost.

Auditing approach	Proof generation cost	Proof verification cost	Communication cost of auditing	Interactions times
l times single auditing	$c(T_{E_1} + T_{M_1})$	$l(4T_p + T_{E_1} + T_{E_2} + T_{M_2} + T_H)$	$l(\lvert G_1\rvert + (c+n)\lvert Z_q\rvert)$	l
Batch auditing with l DOs	$lT_p + lT_{M_1} + lT_{E_1} + lc(T_{E_1} + T_{M_1})$	$(3l+2)T_p + lT_H + lT_{M_1} + lT_{M_2} + 2lT_{E_1} + 2lT_{E_2}$	$l(c+n)\lvert Z_q\rvert + \lvert G_1\rvert + \lvert Z_q\rvert$	1

lower than the cost of l times Auditing procedures. Although the efficiency improvement was not very significant (mainly only n pairing operations off), it was due to the fact that we considered our protocol could resist collusion attacks.

6.2. Experiment Analysis. We implement our CBRDIBA protocol using the Java Pairing-Based Cryptography (JPBC) Library [30] and evaluate it on a personal computer with Intel i7 2.20 GHz quad-cores processor, 16 GB RAM. In our implementation, we use the parameter f.param which is one of the standard parameter settings of the JPBC library. f.param provides an asymmetric pairing. For 80-bit security, only 160 bits are needed to represent elements of $\mathbb{G}_1$ and 320 bits for $\mathbb{G}_2$. To effectively evaluate the performance, the size of the test file we choose is 112 KB (114763 bytes), and we split a test dataset F with size Size(F) bits into m blocks in our experiments. Then, we further divide each data block into n sectors, and the length of each sector is 160 bits. The number m of the data blocks and the number n of the data block's sectors satisfy $(m-1)n \leq \text{Size}(F)/160 \leq mn$.

6.2.1. Proof Verification Cost. We simulate the TPA to run the algorithms Proof Verification and Batch Proof Verification when the total data blocks are 375 with different numbers of DOs from 10 to 100. We define the algorithms Proof Verification and Batch Proof Verification to verify the proof. Both of them will finish the same amount of users' data auditing mission in their own way. In this case, Figure 3 shows that the cost of algorithm Batch Proof Verification is lower than that of Proof Verification.

Furthermore, the TPA runs the algorithms Proof Verification and Batch Proof Verification when the total data blocks are 375 with different numbers of challenged blocks from 10 to 100. In order to control variables, we set the number of data owners in this experiment as 20. In this environment, Figure 4 shows that in most cases, the cost of

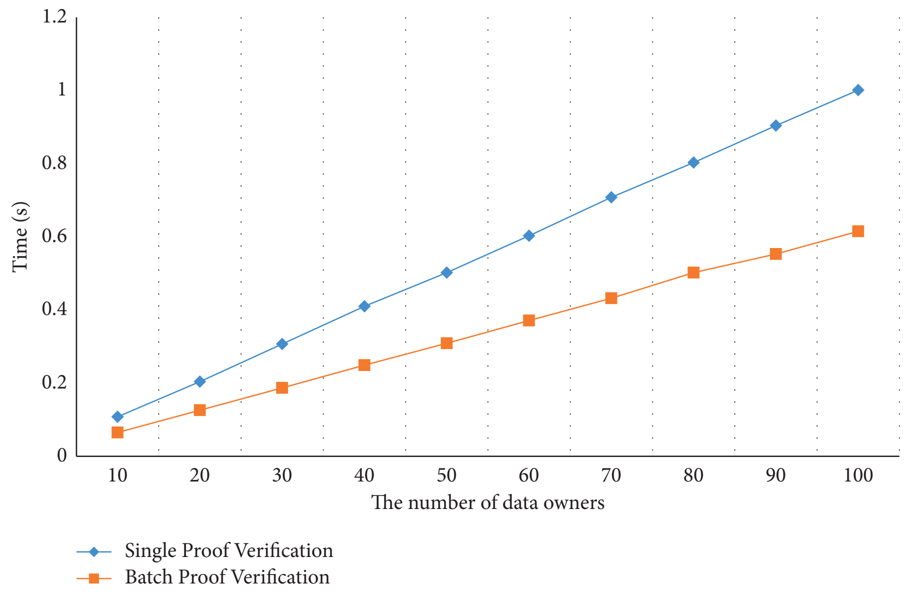

FIGURE 3: Comparison of proof verification cost for different number of data owners.

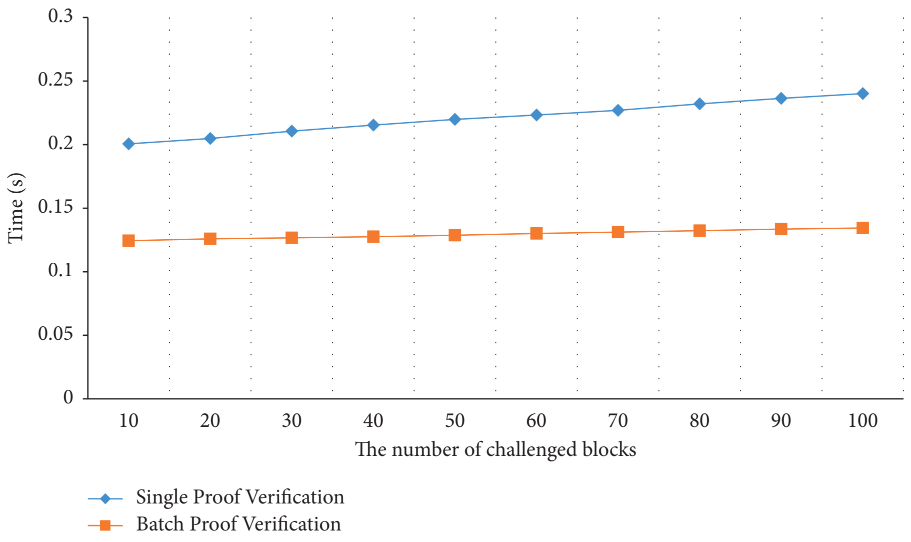

FIGURE 4: Comparison of proof verification cost for different number of challenged blocks.

algorithm Batch Proof Verification is lower than that of Proof Verification.

7. Conclusion

In this paper, we first present a security-enhanced CBRDIBA protocol for cloud-IoT. In our protocol, the TPA can audit the integrity of multiple data simultaneously. If the file is corrupted or lost, the DO will require the CSP to compensate for the damaged file. The correctness and security of the proposed protocol are proved. The security games show that our protocol can resist the highest level of collusion attacks. The communication and computation costs of batch auditing in our protocol have been evaluated through experiments and theoretic analysis. The results indicate that in the case of enhanced security, our batch auditing protocol still has computational efficiency and practicability.

Appendix

A. The Proof of Theorem 1

Proof. Suppose $\mathscr{A}_1$ can break the basic scheme's existential unforgeability against adaptive chosen messages attacks (EUF-CMA) security, then with inputting a random instance $(g_1 \in \mathbb{G}_1, g_2 \in \mathbb{G}_2, W = g_2^a \in \mathbb{G}_2)$, the challenger $\mathscr{B}$ can use $\mathscr{A}_I$ to compute $g_1^a \in \mathbb{G}_1$, and solve the co-CDH problem in PPT. $\mathscr{B}$'s interaction with $\mathscr{A}_I$ is as follows. □

A.1. System Initialization. $\mathscr{B}$ sets $mpk = W$, and runs the system setup algorithm System Setup in our protocol to generate the public parameters params $= (q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e, \psi, mpk)$. Then, $\mathscr{B}$ sends params to $\mathscr{A}_I$. Hash functions H_1, H_2, H_3 are random oracles.

A.2. Oracle Simulation. $\mathscr{A}_I$ is allowed to adaptively issue the queries as follows:

(i) Create User Queries: $\mathscr{B}$ takes the v-th query as ID_v, and assumes the V-th query is the aim identity ($V \in \{1, \ldots, q_u\}$). $\mathscr{B}$ holds a list L_u: (ID_v, upk_v, usk_v) which is initially empty. When $\mathscr{B}$ receives the identity ID_v's query, if ID_v already exists in the list L_u, $\mathscr{B}$ replies the corresponding public key to $\mathscr{A}_I$; otherwise, $\mathscr{B}$ randomly selects $x_v \in \mathbb{Z}_q^*$ and computes ID_v's public key $upk_v = g_2^{x_v}$. $\mathscr{B}$ replies upk_v to $\mathscr{A}_I$ and inserts (ID_v, upk_v, usk_v) into the list L_u.

(ii) H_1 Queries: $\mathscr{B}$ holds a list $L_{H_1} = (ID_v, H_1(ID_v, upk_v), d_v, c)$ which is initially empty. If ID_v has already existed in L_{H_1}, simulator $\mathscr{B}$ will return $H_1(ID_v, upk_v)$ to $\mathscr{A}_1$; otherwise, $\mathscr{B}$ tosses a biased coin with two sides. The probability of the coin coming up heads is ζ, and then $\mathscr{B}$ records $c = 1$. The probability of the coin coming up tails is $1 - \zeta$, and then $\mathscr{B}$ records $c = 0$. $\mathscr{B}$ randomly selects $d_v \in \mathbb{Z}_q^*$ and computes H_1's hash value as follows:

$$H_1(ID_v, upk_v) = \begin{cases} \psi(g_2)^{d_v}, & v \neq V \\ g_1^{d_v}, & v = V, c = 1 \\ \psi(g_2)^{d_v}, & v = V, c = 0 \end{cases} \quad \text{(A.1)}$$

$\mathscr{B}$ replies $H_1(ID_v, upk_v)$ to $\mathscr{A}_I$ and inserts $(ID_v, H_1(ID_v, upk_v), d_v, c)$ into the list L_{H_1}.

(iii) Corruption Queries: In terms of the v-th corruption query, $\mathscr{B}$ checks out the list L_u and gives the corresponding x_v to $\mathscr{A}_I$.

(iv) Certification Queries: $\mathscr{B}$ holds a list L_c consisting of $(ID_v, Cert_{ID_v})$. $\mathscr{A}_I$ issues a certification query on an identity ID_v. If the ID_v exists in the L_c list, then $\mathscr{B}$ replies $Cert_{ID_v}$ to $\mathscr{A}_I$. Otherwise, $\mathscr{B}$ extracts d_v from list L_{H_1} and computes $Cert_v$ as follows:

$$Cert_v = \begin{cases} \psi(W)^{d_v}, & v \neq V \\ \perp, & v = V \end{cases} \quad \text{(A.2)}$$

(v) Key Replace Queries: On receiving a new private/public key pair (usk', upk') on the identity ID_v, $\mathscr{B}$ checks the equation $g_2^{usk'} = upk'$. If the equation holds, then $\mathscr{B}$ updates (ID_v, usk', upk') into the list L_u.

(vi) H_2 Queries: $\mathscr{B}$ holds a list L_{H_2} which is initially empty consisting of $(ID_v, \mathrm{ds}, t, \beta, h, c)$. If $(ID_v, ds\ , t, \beta)$ has already existed in the list L_{H_2}, then $\mathscr{B}$ replies h to $\mathscr{A}_I$. Otherwise, $\mathscr{B}$ randomly selects a $h \in \mathbb{Z}_q^*$, and let $H_2(ID_v, \mathrm{ds}, t, \beta) = h$. $\mathscr{B}$ sends h to $\mathscr{A}_I$, and inserts (ID_v, β, h) to L_{H_2} list.

(vii) H_3 Queries: $\mathscr{B}$ holds a list L_{H_3} which is initially empty consisting of $(ID_v, \tau_0, \gamma_1, \ldots, \gamma_k)$. If (ID_v, τ_0) in L_{H_3} list, $\mathscr{B}$ replies $\gamma_1, \ldots, \gamma_k$ to $\mathscr{A}_I$. Otherwise, $\mathscr{B}$ randomly selects $\gamma_l \in \mathbb{Z}_q^*, l \in [1, k]$, and let $H_3(ID_v, l, \tau_0) = \gamma_l$. $\mathscr{B}$ sends $\gamma_1, \ldots, \gamma_k$ to $\mathscr{A}_I$, and inserts $(ID_v, \tau_0, \gamma_1, \ldots, \gamma_k)$ into the list L_{H_3}.

(viii) TagGen Queries: $\mathscr{A}_I$ self-adaptively selects an identity ID_v and a dataset F with its name ds. Firstly, $\mathscr{B}$ randomly selects $\alpha \in \mathbb{Z}_q^*$ and computes $\beta = g_2^{\alpha}$. If $(ID_v, ds\ , t, \beta)$ has been in L_{H_2}, then terminate the game and output $\perp$. Otherwise, $\mathscr{B}$ simulates $\mathscr{A}_I$ to execute H_2 query and compute τ_0:

$$\tau_{0v} = \begin{cases} Cert_v^{1/usk_v + h}, & v \neq V \\ \perp, & v = V, c = 1 \\ \psi(W)^{d_v/usk_v + h}, & v = V, c = 0 \end{cases} \quad \text{(A.3)}$$

$\mathscr{B}$ sets the file's label $\tau_v = \{\tau_{0v}, \beta_v, ds\ , t, k, n\}$, and then computes the tags for $m_1, \ldots, m_k$. $\mathscr{B}$ computes σ_j for m_j as follows:

$$\sigma_{v,j} = \begin{cases} \left(g_1^{\sum_{l=1}^{k} \gamma_l m_{jl}} \right)^{\alpha}, & v \neq V \\ \perp, & v = V, c = 1 \\ \left(g_1^{\sum_{l=1}^{k} \gamma_l m_{jl}} \right)^{\alpha}, & v = V, c = 0 \end{cases} \quad \text{(A.4)}$$

$\mathscr{B}$ responds the label $\tau_v = \{\tau_{0v}, \beta_v, ds\ , t, k, n\}$ and computes the tags $\sigma_1, \ldots, \sigma_k$ for $m_1, \ldots, m_k$ to $\mathscr{A}_I$.

(ix) ProofCheck Queries: As a TPA, $\mathscr{B}$ issues a challenge on a dataset F (The tags of the dataset have been queried) and then $\mathscr{A}_I$ as a voucher returns the corresponding answer to $\mathscr{B}$.

(1) $\mathscr{B}$ generates a challenge $chal = \{(i, c_i) \colon i \in I, c_i \in \mathbb{Z}_q\}$, where $I \subseteq [1, k]$, and sends it to $\mathscr{A}_I$.

(2) $\mathscr{A}_I$ generates a proof $PF = \{\tau_i, m, \sigma\}$ and returns it to $\mathscr{B}$, where $m = \sum_{i \in I} c_i m_i$ and $\sigma = \prod_{i \in I} \sigma_i{}^{c_i}$.

(3) $\mathscr{B}$ verifies PF and replies the result to $\mathscr{A}_I$.

A.3. Forge. Finally, the adversary $\mathscr{A}_I$ outputs $(ID^*, upk^*, chal^*, PF^*)$. $\mathscr{A}_I$ wins game I if the following conditions are satisfied:

(1) Proof (m^*, σ^*) can pass the algorithm Proof Verification.

(2) $\mathscr{A}_{II}$ has never issued a Corruption Queries on ID^*.

(3) $\mathscr{A}_{II}$ has never issued a TagGen Queries on $(ID^*, F^*, \mathrm{ds}^*)$, or $\mathscr{A}_I$ has issued a TagGen Queries on (ID^*, F^*, ds^*) but $\tau_v \neq \tau^*$.

A.4. Analysis. If $ID^* \neq ID_V$, then $\mathscr{B}$ terminates the simulation and outputs $\perp$. Otherwise, $\mathscr{B}$ first iterates over L_{H_2} list, if $c = 0$, and then terminates the simulation and outputs $\perp$. If $c = 1$, we have:

$$\tau_0^* = Cert^{*\ 1/usk^*+H_2(ID^*,ds\ ,t,\beta)} = \left(g_1^a\right)^{d^*/usk^*+h^*} \Rightarrow g_1^a = \tau_0^{*\ usk^*+h^*/d^*}. \tag{A.5}$$

If public key upk^* is the latest public key which maybe has not been replaced, or maybe has been replaced, $\mathscr{B}$ can figure out the solution of the k-CAA problem $g_1^a = \tau_0{}^{*usk^*+h^*/d^*}$.

A.5. Probability Analysis. If $\mathscr{B}$ can get the solution of the co-CDH problem, it should satisfy the following situations: (1) E_1: The simulator $\mathscr{B}$ never outputs $\perp$. (2) E_2: The adversary $\mathscr{A}_I$ wins the game. (3) E_3: $ID^* = ID_V$ and $c = 1$. The probability that $\mathscr{B}$ succeeds is $\epsilon' = \Pr[E_1 \wedge E_2 \wedge E_3] = \Pr[E_1] \cdot \Pr(E_2|E_1) \cdot \Pr(E_3|E_1 \wedge E_2)$.

(1) If E_1 happens, then we consider the following two circumstances:

(a) $\mathscr{B}$ doesn't output $\perp$ in the Certification Query phase. In this case, the probability is $(1 - 1/q_u)^{q_e}$.

(b) $\mathscr{B}$ doesn't output $\perp$ in the TagGen Query phase. In this case, the probability is $(1 - 1/q_u\zeta)^{q_t}$.

Therefore, we have:

$$\Pr[E_1] = \left(1 - \frac{1}{q_u}\right)^{q_e}\left(1 - \frac{1}{q_u}\zeta\right)^{q_t} \geq \left(1 - \frac{1}{q_u}\right)^{q_e}(1-\zeta)^{q_t}. \tag{A.6}$$

(2) If E_2 happens, then we have $\Pr(E_2|E_1) = \epsilon$.

(3) If E_3 happens, then we have $\Pr(E_3|E_1 \wedge E_2) = \zeta/q_u$.

In summary, we have $\epsilon' \geq (1 - 1/q_u)^{q_e}(1-\zeta)^{q_t}\zeta/q_u\epsilon$. The function $(1-\zeta)^{q_t}\zeta$ is maximized when $\zeta = 1/q_t + 1$. Therefore, $\epsilon' \geq (1 - 1/q_u)^{q_e}(1 - 1/q_t + 1)^{q_t} 1/(q_t + 1)q_u\epsilon$.

B. The Proof of Theorem 2

Proof. Suppose $\mathscr{A}_{II}$ can break the basic scheme's EUF-CMA security, then with inputting a random instance $(h_1, \ldots, h_k \in Z_q^*,\ g_1 \in \mathbb{G}_1, W = g_2^a \in \mathbb{G}_2, (h_1, g_1^{1/a+h_1}), \ldots, (h_k, g_1^{1/a+h_k}))$, the challenger $\mathscr{B}$ can use $\mathscr{A}_{II}$ to output a new pair $(h^*, g_1^{1/a+h^*})$, and solve the k-CAA problem in PPT. $\mathscr{B}$'s interaction with $\mathscr{A}_{II}$ is as follows. ☐

B.1. System Initialization. $\mathscr{B}$ randomly selects $s \in Z_q^*$ as the system master private key msk and computes the system public key $mpk = g_2^s$. $\mathscr{B}$ sends the system public parameters params $= (q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_{\mathbb{T}}, e, \psi, g_1, g_2, mpk)$ and the system private key msk to $\mathscr{A}_{II}$. Hash functions H_1, H_2, H_3 are random oracles.

B.2. Oracle Simulation. The adversary $\mathscr{A}_{II}$ is allowed to adaptively issue the queries as follows:

(i) Create User Queries: $\mathscr{B}$ takes the v-th query as ID_v, and assumes the V-th query is the aim identity $(V \in \{1, \ldots, q_u\})$. $\mathscr{B}$ holds a list L_u: (ID_v, upk_v, usk_v) which is initially empty. When $\mathscr{B}$ receives the identity ID_v's query, if ID_v has already existed in the list L_u, $\mathscr{B}$ replies the corresponding public key to $\mathscr{A}_{II}$; otherwise, $\mathscr{B}$ randomly selects $x_v \in \mathbb{Z}_q^*$ and computes ID_v's public/private key pair (upk_v, usk_v) as follows.

$$(upk_v, usk_v) = \begin{cases} (g_2^{x_v}, x_v), & v \neq V \\ (W, \Delta), & v = V \end{cases} \tag{B.1}$$

$\mathscr{B}$ replies upk_v to $\mathscr{A}_{II}$ and inserts (ID_v, upk_v, usk_v) into the list L_u.

(ii) H_1 Queries: $\mathscr{B}$ holds a list $L_{H_1} = (ID_v, H_1(ID_v, upk_v), d_v)$ which is initially empty. If ID_v has already existed in L_{H_1}, $\mathscr{B}$ will return $H_1(ID_v, upk_v)$ to $\mathscr{A}_{II}$; otherwise, $\mathscr{B}$ randomly selects $d_v \in \mathbb{Z}_q^*$ and computes $H_1(ID_v, upk_v) = g_1^{d_v}$. $\mathscr{B}$ sends $H_1(ID_v, upk_v)$ to $\mathscr{A}_{II}$ and inserts $(ID_v, H_1(ID_v, upk_v), d_v)$ into the list L_{H_1}.

(iii) Corruption Queries: In terms of the v-th corruption query, if ID_v does not exist in list L_u or $ID_v = ID_V$, $\mathscr{B}$ terminates the simulation and outputs $\perp$. Otherwise, $\mathscr{B}$ checks out the list L_u and gives the corresponding private key x_v to $\mathscr{A}_{II}$.

(iv) H_2 Queries: $\mathscr{B}$ holds a list L_{H_2} which is initially empty consisting of $(ID_v, ds\ , t, \beta, h, c)$. If $(ID_v, ds\ , t, \beta)$ has already existed in the list L_{H_2}, then $\mathscr{B}$ replies h to $\mathscr{A}_I$. Otherwise, $\mathscr{B}$ tosses a biased coin with two sides. The probability of the coin coming up heads is ζ, and then $\mathscr{B}$ records $c = 1$. The probability of the coin coming up tails is $1 - \zeta$, and then $\mathscr{B}$ records $c = 0$. Furthermore, $\mathscr{B}$ selects h as follows.

$$H_2(ID_v, ds\ , t, \beta) = \begin{cases} h \in Z_q^*, & v \neq V \\ h \notin \{h_1, \ldots, h_k\}, & v = V, c = 1 \\ h \in \{h_1, \ldots, h_k\}, & v = V, c = 0 \end{cases} \tag{B.2}$$

$\mathscr{B}$ sends $H_2(ID_v, ds\ , t, \beta) = h$ to $\mathscr{A}_{II}$, and inserts $(ID_v, ds\ , t, \beta, h, c)$ to L_{H_2} list.

(v) H_3 Queries: $\mathscr{B}$ holds a list L_{H_3} which is initially empty consisting of $(ID_v, \tau_0, \gamma_1, \ldots, \gamma_k)$. If (ID_v, τ_0) is in L_{H_3} list, $\mathscr{B}$ replies $\gamma_1, \ldots, \gamma_k$ to $\mathscr{A}_{II}$. Otherwise, $\mathscr{B}$ randomly selects $\gamma_l \in \mathbb{Z}_q^*, l \in [1, k]$, and let $H_3(ID_v, l, \tau_0) = \gamma_l$. $\mathscr{B}$ sends $\gamma_1, \ldots, \gamma_k$ to $\mathscr{A}_{II}$, and inserts $(ID_v, \tau_0, \gamma_1, \ldots, \gamma_k)$ into the list L_{H_3}.

(vi) TagGen Queries: $\mathscr{A}_{II}$ adaptively selects an identity ID_v and a dataset F with its name ds. Firstly, $\mathscr{B}$ randomly selects $\alpha \in \mathbb{Z}_q^*$ and computes $\beta = g_2^{\alpha}$. If $(ID_v, ds\ , t, \beta)$ has been in L_{H_2}, then, terminate the game and output $\perp$. Otherwise, $\mathscr{B}$ simulates $\mathscr{A}_{II}$ to execute H_2 query and compute τ_0:

$$\tau_0 = \begin{cases} \left(g_1^{d_v s}\right)^{1/x_v+h} & v \neq V \\ \perp, & v = V, c = 1 \\ \left(g_1^{1/a+h}\right)^{d_v s}, & v = V, c = 0 \end{cases} \tag{B.3}$$

$\mathscr{B}$ sets the file's label $\tau = \{\tau_0, \beta, ds\ , t\}$, and then computes the tags for $m_1, \ldots, m_k$. $\mathscr{B}$ computes σ_j for m_j as follows:

$$\sigma_j = \begin{cases} \left(g_1^{\sum_{l=1}^{k}} \gamma_l m_{jl} \right)^{\alpha}, & v \neq V \\ \perp, & v = V, c = 1 \\ \left(g_1^{\sum_{l=1}^{k}} \gamma_l m_{jl} \right)^{\alpha}, & v = V, c = 0 \end{cases} \tag{B.4}$$

$\mathscr{B}$ replies the label $\tau = \{\tau_0, \beta, ds\ , t\}$ and computes the tags $\sigma_1, \ldots, \sigma_k$ for $m_1, \ldots, m_k$ to $\mathscr{A}_I$.

B.3. Forge. Finally, the adversary $\mathscr{A}_I$ outputs $(ID^*, upk^*, chal^*, PF^*)$. $\mathscr{A}_I$ wins the game if the following conditions are satisfied:

(1) Proof (m^*, σ^*) can pass the algorithm Proof Verification.

(2) $\mathscr{A}_{II}$ has never issued a Corruption Queries on ID^*.

(3) $\mathscr{A}_{II}$ has never issued a TagGen Queries on (ID^*, F^*, ds^*), or $\mathscr{A}_I$ has issued a TagGen Queries on (ID^*, F^*, ds^*) but $\tau_v \neq \tau^*$.

B.4. Analysis. If $ID^* \neq ID_V$, then $\mathscr{B}$ terminates the simulation and outputs $\perp$. Otherwise, $\mathscr{B}$ first iterates over L_{H_2} list, if $c = 0$, and then terminates the simulation and outputs $\perp$. If $c = 1$ and $h^* \notin \{h_1, \ldots, h_k\}$, we have:

$$\begin{aligned} \tau_0^* &= Cert^{*\ 1/usk^* + H_2 \left(ID^*, ds\ , t, \beta\right)} \\ &= \left(H_1\left(ID^*, upk^*\right)^s\right)^{1/usk^* + h^*} = \left(g_1^{d^* s}\right)^{1/a+h^*}. \end{aligned} \tag{B.5}$$

Therefore, $\mathscr{B}$ can figure out the solution of the k-CAA problem $g_1^{1/a+h^*} = \tau_0^{1/d^* s}$

B.5. Probability Analysis. If $\mathscr{B}$ can get the solution of the k-CAA problem, it should satisfy the following situations: (1) E_1: The simulator $\mathscr{B}$ never outputs $\perp$. (2) E_2: The adversary $\mathscr{A}_{II}$ wins the game. (3) E_3: $ID^* = ID_V$ and $c = 1$. The probability that $\mathscr{B}$ succeeds is $\epsilon' = \Pr[E_1 \wedge E_2 \wedge E_3] = \Pr[E_1] \cdot \Pr(E_2 | E_1) \cdot \Pr(E_3 | E_1 \wedge E_2)$.

(1) If E_1 happens, then we consider the following two circumstances:

(a) $\mathscr{B}$ doesn't output $\perp$ in the Corruption Queries phase. In this case, the probability is $(1 - 1/q_u)^{q_r}$.

(b) $\mathscr{B}$ doesn't output $\perp$ in the TagGen Queries phase. In this case, the probability is $(1 - 1/q_u \zeta)^{q_t}$.

Therefore, we have:

$$\Pr[E_1] = \left(1 - 1/q_u\right)^{q_r} \left(1 - 1/q_u \zeta\right)^{q_t} \geq \left(1 - 1/q_u\right)^{q_r} \left(1 - \zeta\right)^{q_t} \tag{B.6}$$

(2) If E_2 happens, then we have $\Pr(E_2 | E_1) = \epsilon$.

(3) If E_3 happens, then we have $\Pr(E_3 | E_1 \wedge E_2) = \zeta / q_u$.

In summary, we have $\epsilon' \geq (1 - 1/q_u)^{q_r} (1 - \zeta)^{q_t} \zeta / q_u \epsilon$. The function $(1 - \zeta)^{q_t} \zeta$ is maximized when $\zeta = 1/q_t + 1$. Therefore, $\epsilon' \geq (1 - 1/q_u)^{q_r} (1 - 1/q_t + 1)^{q_t} 1/(q_t + 1) q_u \epsilon$.

C. The Proof of Theorem 3

Proof. Suppose adversary $\mathscr{A}_{III}$ can forge a valid single proof successfully. The System Initialization and the Oracle Simulation are the same as those in Game I or Game II. ☐

C.1. ProofCheck. $\mathscr{A}_{III}$ generates the proof PF using some data blocks and the corresponding tags, and sends PF and challenge to $\mathscr{B}$. $\mathscr{B}$ validates the proof and returns the result to $\mathscr{A}_{III}$.

C.2. Challenge. The simulator $\mathscr{B}$ generates a challenge $chal = \left\{(i, c_i)\colon i \in I, c_i \in Z_q\right\}$, where $I \subseteq [1, k]$. There is at least a challenged data block having never been queried tag. And then $\mathscr{B}$ sends the challenge to $\mathscr{A}_{III}$.

C.3. Forge. The adversary $\mathscr{A}_{III}$ outputs a valid proof $\overline{PF} = \{m, \overline{\sigma}\}$ and returns it to $\mathscr{B}$, where $m = \sum_{i \in I} c_i m_i$ and $\overline{\sigma} = \prod_{i \in I} \overline{\sigma}_i^{c_i}$.

C.4. Analysis. Since the forged proof is valid, it can make the following equation hold.

$$e\left(\overline{\sigma}, g_2\right) = e\left(g_1^{\sum_{l=1}^{k} H_3 \left(ID^*, j, \tau_0\right) m_j}, \beta \right). \tag{C.1}$$

Assume that the real proof for the challenge *chal* is $PF = \{m, \sigma\}$; it can also make the equation holds.

$$e\left(\sigma, g_2\right) = e\left(g_1^{\sum_{l=1}^{k} H_3 \left(ID^*, j, \tau_0\right) m_j}, \beta \right). \tag{C.2}$$

Due to the collision resistance of the hash function, the adversary $\mathscr{A}_{III}$ can get the only response when it issues a H_1 queries and similarly, H_2 queries and H_3 queries. Obviously, the above two equations are equal, i.e., $\sigma = \overline{\sigma}$, i.e., $\prod_{i\in I}\sigma_i{}^{c_i} = \prod_{i\in I}\overline{\sigma_i}^{c_i}$. Because $\sigma_i, \overline{\sigma}_i \in \mathbb{G}_1$, there exists $x_i, y_i \in Z_q^*$ satisfying $\sigma_i = g_1^{x_i}$ and $\overline{\sigma_i} = g_1^{y_i}$.

We get $g_1^{\sum_{i\in I} c_i x_i} = g_1^{\sum_{i\in I} c_i y_i}$, i.e., $\sum_{i\in I} c_i x_i = \sum_{i\in I} c_i y_i$, which means $\sum_{i\in I} c_i (x_i - y_i) = 0$. Since $c_i \in Z_q^*$, we get $x_i = y_i$ mod q. This is contrary to the previous results. According to Theorems 1 and 2, the probability of forging a single tag is negligible. Therefore, the probability of the adversary $\mathscr{A}_{III}$ forging a valid proof successfully is negligible if the file has been damaged or modified.

Above all, Theorem 3 is proved.

D. The Proof of Theorem 4

Proof. Suppose adversary $\mathscr{A}_{IV}$ has advantage ϵ in forging a valid batch proof; then, there exists a simulator $\mathscr{B}$ that has advantage ϵ in breaking the collision-resistance property of hash function H_0. $\mathscr{B}$ interacts with $\mathscr{A}_{IV}$ as follows. □

D.1. System Initialization. On inputting the security parameter λ, the challenger $\mathscr{B}$ generates the system parameter params $= \{q, G_1, G_2, G_T, e, \psi, g_1, g_2, mpk, H_0, H_1, H_2, H_3\}$. Furthermore, $\mathscr{B}$ randomly selects $s_{ver} \in Z_q^*$ as the TPA's private key and computes $upk_{ver} = g_2^{s_{ver}}$ as the TPA's public key. Then $\mathscr{B}$ gives $\mathscr{A}_{IV}$ the params and upk_{ver}.

D.2. Oracle Simulation. $\mathscr{A}_{IV}$ can adaptively issue the following queries:

(i) Corruption Queries: $\mathscr{A}_{IV}$ requests such a query, and $\mathscr{B}$ generates the key pairs $(upk_{I\,D}, usk_{I\,D})$ by running the algorithm UserKeyGen, and then returns $usk_{I\,D}$ to $\mathscr{A}_{IV}$.

(ii) Batch Proof Check Queries: The simulator $\mathscr{B}$ generates some challenges and sends the challenges $chal_z = \{ID_z, \mathrm{ds}_z, (i, c_{zi}), i \in I_z\}$ to $\mathscr{A}_{IV}$. Then, on receiving the challenges from $\mathscr{B}$, $\mathscr{A}_{IV}$ generates a batch proof BPF by running the algorithm Batch Proof Generation and returns the result to $\mathscr{A}_{IV}$.

D.3. Forge. At last, $\mathscr{A}_{IV}$ outputs its forgery, i.e., a set of four-tuples $\{(ID_z, upk_{ID_z}, m_z, \sigma_z), z = 1, \ldots, l\}$ consisting of identities, user public keys, challenged data, challenged tags, and a value $\widetilde{\sigma}^*$ supposed to be the batch proof on the set of four-tuples. If the following conditions are satisfied, $\mathscr{A}_{IV}$ wins the game:

(1) The batch proof $\widetilde{\sigma}^*$ is generated from all the single challenged tags.

$$r = H_0\left(e\left(\sigma_1, upk_{ver}\right), \ldots, e\left(\sigma_l, upk_{ver}\right)\right). \tag{D.1}$$

(2) The batch proof $\widetilde{\sigma}^*$ is valid, i.e.,

$$r = H_0\left(e\left(\left(g_1^{\sum_{j=1}^{n} H_3(ID_1, j, \tau_{01})\cdot m_{1j}}\right)^{s_{ver}}, \beta_1\right), \ldots, e\left(\left(g_1^{\sum_{j=1}^{n} H_3(ID_l, j, \tau_{0l})\cdot m_{lj}}\right)^{s_{ver}}, \beta_l\right)\right) \tag{D.2}$$

(3) There exists at least one $z^* \in [1, l]$, such that σ_{z^*} is invalid, i.e.,

$$e\left(\sigma_{i^*}, g_2\right) \neq e\left(g_1^{\sum_{j=1}^{n} H_3(ID_{i^*}, j, \tau_{0i^*}) m_{i^*\,j}}, \beta_{i^*}\right). \tag{D.3}$$

Note that the hash values r and r' are the same since

$$\begin{aligned} r &= H_0\left(e\left(\sigma_1, upk_{ver}\right), \ldots, e\left(\sigma_l, upk_{ver}\right)\right) \\ &= H_0\left(e\left(\left(g_1^{\sum_{j=1}^{n} H_3(ID_1, j, \tau_{01})\cdot m_{1j}}\right)^{s_{ver}}, \beta_1\right), \ldots, e\left(\left(g_1^{\sum_{j=1}^{n} H_3(ID_l, j, \tau_{0l})\cdot m_{lj}}\right)^{s_{ver}}, \beta_l\right)\right) = r' \end{aligned} \tag{D.4}$$

On the other hand, these two inputs are different since

$$e\left(\sigma_{i^*}, g_2\right) \neq e\left(g_1^{\sum_{j=1}^{n} H_3(ID_{i^*}, j, \tau_{0i^*}) m_{i^*\,j}}, \beta_{i^*}\right). \tag{D.5}$$

Therefore, $\mathscr{B}$ presents a pair of collisions of hash function H_0.

We complete the description of how $\mathscr{B}$ outputs a pair of collisions. Then, we analyze the advantage of $\mathscr{B}$ who holds the master secret key msk and can answer the Corruption Queries. Moreover, $\mathscr{B}$ simulates the TPA such that $\mathscr{B}$ holds the TPA's private key usk_{ver}, and hence $\mathscr{B}$ can answer the Corruption Queries. Our simulation scheme is indistinguishable from the real one. If $\mathscr{A}_{IV}$ has advantage ϵ in forging a valid batch proof, then $\mathscr{B}$ has advantage ϵ in generating a pair of collisions of hash function H_0.

Above all, Theorem 4 is proved.

Data Availability

The data supporting the findings of this study are available from the corresponding author upon reasonable request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by the Nature Science Foundation of Jiangsu Province under Grant BK20190696.

References

[1] H. Yan, J. Li, J. Han, and Y. Zhang, "A novel efficient remote data possession checking protocol in cloud storage," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 1, pp. 78–88, 2016.

[2] B. Grobauer, W. Tobias, and E. Stocker, "Understanding cloud computing vulnerabilities," *IEEE Security & privacy*, vol. 9, no. 2, pp. 50–57, 2010.

[3] C. Stergiou, K. E. Psannis, B G Kim, B. Kim, and B. Gupta, "Secure integration of iot and cloud computing," *Future Generation Computer Systems*, vol. 78, pp. 964–975, 2018.

[4] Y. Deswarte, J J Quisquater, and A. Saïdane, "Remote integrity checking," in *Proceedings of the Working conference on integrity and internal control in information systems*, pp. 1–11, Springer, New York, NY, USA, June 2003.

[5] D. Boneh, D. Freeman, J. Waters, and B. Waters, "Signing a linear subspace: signature schemes for network coding," in *Proceedings of the International Workshop on Public Key Cryptography*, pp. 68–87, Springer, New York, NY, USA, 2009.

[6] K. He, C. Huang, and J. Wang, "An efficient public batch auditing protocol for data security in multi-cloud storage," in *Proceedings of the 2013 8th ChinaGrid Annual Conference*, pp. 51–56, IEEE, Los Alamitos, CA, USA, August 2013.

[7] Li Xiong, S. Liu, and R. Lu, "An efficient privacy-preserving public auditing protocol for cloud-based medical storage system," *IEEE Journal of Biomedical and Health Informatics*, vol. 2022, Article ID 3140831, 1 page, 2022.

[8] D. Franklin and M. Franklin, "Identity-based encryption from the weil pairing, Advances in Cryptology - CRYPTO 2001," in *Proceedings of the Annual International Cryptology Conference*, pp. 213–229, Springer, August 2001.

[9] S. S. Al-Riyami and K. G. Paterson, "Certificateless public key cryptography, Advances in Cryptology - ASIACRYPT 2003," in *Proceedings of the International Conference on the Theory and Application of Cryptology and Information Security*, pp. 452–473, Springer, New York, NY, USA, May 2003.

[10] G. Gentry, "Certificate-based encryption and the certificate revocation problem, Lecture Notes in Computer Science," in *Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 272–293, Springer, New York, NY, USA, May 2003.

[11] R. L. Rivest, *Two signature schemes*, Cambridge University, Cambridge, England, 2000.

[12] F. Zhao, T. Kalker, M. Médard, and J H. Keesook, "Signatures for content distribution with network coding," in *Proceedings of the 2007 IEEE International Symposium on Information Theory*, pp. 556–560, IEEE, Nice, France, June 2007.

[13] G. Ateniese, R. Burns, and R. Curtmola, "Provable data possession at untrusted stores," in *Proceedings of the 14th ACM Conference on Computer and Communications Security*, pp. 598–609, ACM, New York, NY, USA, 2007.

[14] Y. Yu, Y. Zhang, Yi Mu, W. Liu, and H. Liu, "Provably secure identity based provable data possession," in *Proceedings of the International Conference on Provable Security*, pp. 310–325, Springer, New York, NY, USA, July 2015.

[15] J. Li, H. Yan, and Y. Zhang, "Certificateless public integrity checking of group shared data on cloud storage," *IEEE Transactions on Services Computing*, vol. 14, no. 1, pp. 71–81, 2018.

[16] Y. Li and F. Zhang, "An Efficient Certificate-Based Data Integrity Auditing Protocol for Cloud-Assisted Wbans," *IEEE Internet of Things Journal*, vol. 9, 2021.

[17] S. Li, C. Xu, Y. Zhang, Y. Du, and K. Chen, "Blockchain-based Transparent Integrity Auditing and Encrypted Deduplication for Cloud Storage," *IEEE Transactions on Services Computing*, vol. 2022, Article ID 3144430, 2022.

[18] D. Boneh, G. Gentry, B. Shacham, and H. Shacham, "Aggregate and verifiably encrypted signatures from bilinear maps, Lecture Notes in Computer Science," in *Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 416–432, Springer, New York, NY, USA, May 2003.

[19] G. Gentry and Z. Ramzan, "Identity-based aggregate signatures, Public Key Cryptography - PKC 2006," in *Proceedings of the International Workshop on Public Key Cryptography*, pp. 257–273, Springer, New York, NY, USA, August 2006.

[20] L. Shen, J. Ma, Y. Liu, and H. Liu, "Provably secure certificateless aggregate signature scheme with designated verifier in an improved security model," *IET Information Security*, vol. 13, no. 3, pp. 167–173, 2019.

[21] G. K. Verma, B. B. Singh, N. Kumar, O. Kaiwartya, and M. S. Obaidat, "Pfcbas: pairing free and provable certificate-based aggregate signature scheme for the e-healthcare monitoring system," *IEEE Systems Journal*, vol. 14, no. 2, pp. 1704–1715, 2019.

[22] Ge Wu, F. Zhang, L. Shen, F. Susilo, and W. Susilo, "Certificateless aggregate signature scheme secure against fully chosen-key attacks," *Information Sciences*, vol. 514, pp. 288–301, 2020.

[23] Z. Jing, "An efficient homomorphic aggregate signature scheme based on lattice," *Mathematical Problems in Engineering*, vol. 2014, Article ID 536527, 2014.

[24] M. Ajtai, "Generating hard instances of the short basis problem," in *Proceedings of the International Colloquium on Automata, Languages, and Programming*, pp. 1–9, Springer, Heidelberg, Germany, October 1999.

[25] J. Han, Y. Li, J. Liu, and M. Zhao, "An efficient lucas sequence-based batch auditing scheme for the internet of medical things," *IEEE Access*, vol. 7, Article ID 10077, 2018.

[26] P. McDaniel and W. L. McDaniel, "The square terms in lucas sequences," *Journal of Number Theory*, vol. 58, no. 1, pp. 104–123, 1996.

[27] F. Wang, Li Xu, and K. K. R. Choo, "Lightweight certificate-based public/private auditing scheme based on bilinear pairing for cloud storage," *IEEE Access*, vol. 8, pp. 2258–2271, 2019.

[28] H. Yang, Ye Su, J. Qin, J. Wang, and H. Wang, "An efficient public batch auditing scheme for data integrity in standard model, Machine Learning for Cyber Security," in *Proceedings of the International Conference on Machine Learning for Cyber Security*, pp. 578–592, Springer, New York, NY, USA, January 2020.

[29] L. Huang, J. Zhou, G. Zhang, and M. Zhang, "Certificateless public verification for data storage and sharing in the cloud," *Chinese Journal of Electronics*, vol. 29, no. 4, pp. 639–647, 2020.

[30] A. De Caro and V. Iovino, "jpbc: Java pairing based cryptography," in *Proceedings of the 2011 IEEE Symposium on Computers and Communications (ISCC)*, pp. 850–855, IEEE, Kerkyra, Greece, June 2011.