Research Article

Security Analysis and Improvements on a Remote Integrity Checking Scheme for Regenerating-Coding-Based Distributed Storage

Guangjun Liu ¹, Wangmei Guo ², Ximeng Liu ³, and Jinbo Xiong ⁴

¹School of Information Engineering, Xi'an University, Xi'an 710065, China

²State Key Laboratory of Integrated Services Networks, Xidian University, Xi'an 710071, China

³College of Mathematics and Computer Science, Fuzhou University, Fuzhou 350108, China

⁴College of Mathematics and Informatics, Fujian Normal University, Fuzhou 350117, China

Correspondence should be addressed to Guangjun Liu; liuguangjuns@gmail.com

Received 12 December 2020; Revised 24 March 2021; Accepted 5 April 2021; Published 21 April 2021

Academic Editor: Mamoun Alazab

Copyright © 2021 Guangjun Liu et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Enabling remote data integrity checking with failure recovery becomes exceedingly critical in distributed cloud systems. With the properties of a lower repair bandwidth while preserving fault tolerance, regenerating coding and network coding (NC) have received much attention in the coding-based storage field. Recently, an outstanding outsourced auditing scheme named NC-Audit was proposed for regenerating-coding-based distributed storage. The scheme claimed that it can effectively achieve lightweight privacy-preserving data verification remotely for these networked distributed systems. However, our algebraic analysis shows that NC-Audit can be easily broken due to a potential defect existing in its schematic design. That is, an adversarial cloud server can forge some illegal blocks to cheat the auditor with a high probability when the coding field is large. From the perspective of algebraic security, we propose a remote data integrity checking scheme RNC-Audit by resorting to hiding partial critical information to the server without compromising system performance. Our evaluation shows that the proposed scheme has significantly lower overhead compared to the state-of-the-art schemes for distributed remote data auditing.

1. Introduction

Distributed cloud storage provides an on-demand data outsourcing service and has become a popular research trend due to its elasticity and low maintenance cost. However, the shortcoming of this system is that some storage nodes could be untrustworthy, which makes data integrity or reliability become increasingly troublesome tasks for data owners. Regenerating coding, an alternative technology designed for these networked storage systems, shows a potential advantage to reduce data repair bandwidth if the outsourced data integrity is the precondition of such systems. Therefore, enabling remote data integrity verification becomes fundamental and crucial for regenerating-coding-based cloud storage systems [1–3].

Many solutions realizing outsourced integrity checking without local copy have been presented under several system and security models, such as provable data possession (PDP) [2–6] and proof of retrievability (PoR) [7–10]. PDP approaches always resort to some homomorphic authentication schemes for ensuring possession of files on untrusted storages, while PoR approaches combine spot-checking and error correcting codes to ensure both possession and retrievability of files on archive service systems. To enable integrity verification, both solutions require a cloud server to return the proof response of some particular hosted data blocks specified by the auditor (or the user itself). If the returned proof cannot pass the auditor's verification checking, the auditor concludes that data hosted in that server are corrupted. Both PDP and PoR are the earliest

solutions of data integrity auditing in a single cloud environment, in which only one copy of user data is stored in the cloud. Considering that files are usually striped and redundantly stored across some distributed systems (i.e., multiple servers or multiple clouds), the works in [7, 9–12] explore integrity verification suitable for such distributed setting with different redundancy schemes, such as replication, erasure codes, and regenerating codes. In this paper, we focus on the remote data integrity verification problem in regenerating-code-based distributed storage systems.

Most traditional PoR approaches cannot support popular cloud storage applications because of their random permutations for auditing. Although some PDP schemes combined error correction codes suggesting to preprocess the outsourced data, this is inefficient by decoupling error correction. Therefore, the state-of-the-art approaches can only partially solve cloud data integrity checking, and they still face usability challenges in practical scenarios due to their low efficiency for regenerating-coding-based distributed cloud storage [13–15]. As we all know, usability requires that a cloud data integrity scheme is practically secure, supports real-time applications, and also runs fast. Actually, most of the existing solutions either involve large-scale encoding computation of the stored data or need to generate much-processed parity data authenticated. When these issues meet regeneration-coding-based systems [5, 8, 9], the distributed storage performance will degrade dramatically, far from reaching the usability.

There have been only a few number of works that have been devoted to usable regeneration-coding-based applications. For achieving lightweight implementation cost, Le et al. [11] proposed a symmetric-key-based privacy-preserving auditing scheme called NC-Audit, which presents relatively efficient performance. The authors claimed that it can realize remote data privacy-preserving auditing along with failure repair to cloud storage nodes. However, the scheme has some fatal security weakness to break the integrity checking protocol as illustrated in this paper. Recently, Lakshmi and Deepthi [10] proposed a homomorphic encryption scheme based on channel coding for regenerating-coding-based storage system, which realizes verifiable computation and error correction with a very small amount of bandwidth. However, large-scale matrix multiplication operations are involved during the process of audit and error correction, which brings in heavy online computation overhead.

In parallel, another kind of interesting and possible solution to maintain remote system security is presented in [16–18], which targets malware detection or tamper resistance in storage networks. However, these works only focus on the software or hardware security, orthogonal to the data security in this paper. Different from these active attack detection mechanisms, we only focus on remote data integrity verification solutions in distributed cloud storage.

The contributions of this paper are threefold as listed below.

- (1) We point out that the scheme NC-Audit for regenerating-coding-based storage is actually insecure for that it cannot satisfy the fundamental

auditing security, which makes an adversarial storage node to successfully forge an illegal response to pass the auditing verification even if the storage node has deleted the user's whole file.

- (2) We further propose an improved algebraic security-based remote data auditing scheme named RNC-Audit (Revised NC-Audit) to fill the security gap between the coding reliability and the usability of integrity checking. The methodology behind this work is to scramble partial key parameters to strengthen the security.
- (3) The proposed scheme is practically secure and does not rely on any computation-heavy arithmetic calculations. It supports existing distributed cloud storage applications and works over computation-efficient finite fields.

The rest of the paper is organized as follows. In Section 2, we formulate the system model and the threat model. In Section 3, we describe the scheme NC-Audit execution between the user and a single storage node. In Section 4, we elaborate security analysis on NC-Audit. In Section 5, we put forward an improved algebraic security-based scheme RNC-Audit. In Section 6, we explain the correctness of RNC-Audit. In Section 7, we evaluate the communication and computational efficiency. In Section 8, we conclude the paper.

2. System and Threat Model

2.1. System Model. Similar to [11], we consider a cloud storage service involving a user and a third-party auditor (TPA) and some regenerating-coding-based storage nodes which make up the cloud storage provider (CSP). The user uploads his data to the storage nodes and resorts to the TPA to check for the integrity of the outsourced data at each node. Particularly, the user does not want the TPA to learn about his data privacy. The auditing system model is shown in Figure 1.

Before data uploading, the user encodes the file using a regenerating code and uploads the encoded data to N storage nodes $\mathbb{N}_1, \mathbb{N}_2, \dots, \mathbb{N}_N$ in a distributed way. The detailed procedure is as follows.

- (1) Divide the file into a sequence of initial message vectors, i.e., $\hat{\mathbf{v}}_1, \hat{\mathbf{v}}_2, \dots, \hat{\mathbf{v}}_m$, where $\hat{\mathbf{v}}_i \in \mathbb{F}_q$, $i = 1, 2, \dots, m$, and $\mathbb{F}_q$ is a finite field of size q .
- (2) Expand the vector $\hat{\mathbf{v}}_i$ to $\mathbf{v}_i = (\hat{\mathbf{v}}_i, \mathbf{e}_i) \in \mathbb{F}_q^{n+m}$ where $n \gg m$, $i = 1, 2, \dots, m$, and $\mathbf{e}_i$ is an m -dimensional unit vector whose i -th element is 1.
- (3) Compute

$$\mathbf{y}_{sj} = \sum_{i=1}^m \alpha_{sij} \mathbf{v}_i = (\hat{\mathbf{y}}_{sj}, \mathbf{g}_{\mathbf{y}_{sj}}), \quad (1)$$

for $j = 1, 2, \dots, M$, where α_{sij} is randomly chosen in $\mathbb{F}_q$ and $\mathbf{g}_{\mathbf{y}_{sj}}$ is termed as $\mathbf{y}_{sj}$'s encoding coefficient and is composed of the last m elements of $\mathbf{y}_{sj}$.

FIGURE 1: Entities and steps involved in an auditing system: three entities, i.e., CSP, TPA, and a user, are always included in this model, where TPA is fully responsible for the remote audit of the user data stored in CSP using a challenge-response authentication protocol.

- (4) For $s = 1, 2, \dots, N$, upload the digital signed vectors $\mathbf{y}_{si}, i = 1, 2, \dots, M$, to the storage node N_s .

When a (N, k, d, M, β) regenerating code is adopted in a distributed storage system with N storage nodes (each node stores M blocks), the data file distributed this system can be restored by accessing M data blocks from any of the k healthy nodes. When a server node fails, the data stored in the failed server can be reconstructed by retrieving β ($\beta \leq \alpha$) data blocks from any d ($k \leq d \leq n - 1$) healthy servers, and therefore, the repair bandwidth is $\gamma = d\beta$. In the example given in Figure 2, $N = 4, k = 2, d = 3, M = 2, \beta = 1$, and $\gamma = 3$.

The symbolic representations described above will continue to be used in the following text. We refer the interested reader to the literature about regenerating code construction [19–12].

2.2. Threat Model. We consider semitrusted storage nodes that are faithful and do not deviate from the auditing protocol. However, they may deliberately delete rarely accessed user's data to reduce storage costs; they may also try to hide data corruptions caused by either internal or external factors to maintain reputation. For clarity, we concentrate on our discussion between a single storage node and the TPA.

We assume that the TPA, who is in charge of remote data integrity checking (i.e., data auditing), is independent and reliable. The TPA has no willingness to collude with existing storage nodes but has a strong desire to extract or leak user's secret keys. This is a general assumption when relying on a TPA for data auditing to reduce the user's burden [5, 6, 8].

As the practical requirements of cryptographic protocols, both the TPA and the storage node are fully aware of the protocols used.

2.3. Auditing Model. Generally, a remote data auditing scheme for cloud storage always includes a three-stage process, i.e., initialization, outsourcing upload, and data auditing. The initialization stage mainly generates system parameters and protocol security keys; in outsourcing upload, a user needs to generate authentication tags for all the outsourced blocks and outsources them to storage nodes; in the stage of data auditing, cloud server computes a response (i.e., a linear combination of some given blocks and its authentication tag) challenged by the TPA, and the TPA verifies whether the tag is a valid tag of the combined block.

3. Description of NC-Audit

For clarity, this section focuses on the discussion of NC-Audit execution between the user and a single CSP storage node.

In NC-Audit, every initial message block $\hat{\mathbf{v}}_i$ consists of n characters in $\mathbb{F}_q$, while the last two characters are padded randomly. The scheme introduces three pseudo-random functions (PRFs):

$$\begin{aligned} \mathbf{F}_1: \mathbf{K} \times \mathbf{ID} \times \mathbb{Z}^+ &\longrightarrow \mathbb{F}_q, \\ \mathbf{F}_2: \mathbf{K} \times \mathbb{Z}^+ \times \mathbb{Z}^+ &\longrightarrow \mathbb{F}_q, \\ \mathbf{F}_3: \mathbf{K} \times \{0, 1\}^\lambda \times \mathbb{Z}^+ &\longrightarrow \mathbb{F}_q, \end{aligned} \quad (2)$$

where $\mathbb{Z}^+$ indicates the positive integer set, $\mathbf{K}$ denotes the PRF key set, and $\mathbf{ID}$ is the file identifier set. NC-Audit consists of three phases as follows.

(1) Initialization:

- (a) Setting security parameters λ and PRFs, then the user shares a unique key k_e and k_v with the storage node and the TPA, respectively, where k_e

FIGURE 2: Repairing a failed node [19]: the original file consists of four blocks— A , B , C , and D . A $(4, 2)$ MDS (maximum distance separable) code is applied such that any two healthy nodes can restore the original file while the repair bandwidth is 1 for each helper node. The repair involves combining blocks and the total repair bandwidth consisting of 3 blocks instead of 4, where 4 is the amount of blocks needed to reconstruct the whole file.

is used for encryption at the storage node and k_v is for verification at the TPA.

- (b) Inputting k_v and $\text{id} \in \mathbf{ID}$, the user and TPA compute the vectors $\bar{\mathbf{r}} \in \mathbb{F}_q^{n-2}$ and $\mathbf{r} \in \mathbb{F}_q^{n+m}$ with $\mathbf{F}_1$, where $\bar{\mathbf{r}}$ is constructed by the first $n-2$ elements of $\mathbf{r}$ and $\mathbf{r}[i] = \mathbf{F}_1(k_v, \text{id}, i) \in \mathbb{F}_q$ for $i = 1, 2, \dots, n+m$.
- (c) Inputting k_e , the user generates the vector $\bar{\mathbf{p}}_i \in \mathbb{F}_q^{n-2}$ with $\mathbf{F}_2$ and computes

$$p_i = \bar{\mathbf{r}} \cdot \bar{\mathbf{p}}_i, \quad (3)$$

for $i = 1, 2, \dots, n-2$, and then sends $p_1, p_2, \dots, p_{n-1}$ to $\mathbb{N}_s$. Here the j -th element of the vector $\bar{\mathbf{p}}_i$ is determined by $\bar{\mathbf{p}}_i[j] = \mathbf{F}_2(k_e, \text{id}, j) \in \mathbb{F}_q$.

- (2) Outsourcing upload:

For $j = 1, 2, \dots, M$, the user works as follows.

- (a) Generate $\mathbf{y}_{sj}$ as in Section 2.1 and compute

$$t_{sj} = \mathbf{y}_{sj} \cdot \mathbf{r} \in \mathbb{F}_q \quad (4)$$

as the authentication tag of the vector $\mathbf{y}_{sj}$.

- (b) Retain the secret keys and the encoding coefficient vector $\mathbf{g}_{\mathbf{y}_{sj}}$, send $\mathbf{g}_{\mathbf{y}_{sj}}$ to TPA, and upload the vector $\mathbf{z}_{sj} = (\mathbf{y}_{sj}, t_{sj})$ to $\mathbb{N}_s$. In fact, the user can delete the whole data of the file denoted by id .

- (3) Data auditing:

- (a) The TPA selects an outsourced vector index set $\Delta \subseteq \{1, 2, \dots, M\}$, chooses $\varepsilon_i \in \mathbb{F}_q (i \in \Delta)$ randomly, and then sends the challenge message $\text{chal} = \{|\langle i, \varepsilon_i \rangle| i \in \Delta\}$ to $\mathbb{N}_s$.
- (b) $\mathbb{N}_s$ computes the aggregated vector:

$$\mathbf{e} = \sum_{i \in \Delta} \varepsilon_i \mathbf{z}_{si} = (\bar{\mathbf{e}}, e_{n-1}, e_n, \mathbf{g}_e, t), \quad (5)$$

and randomly chooses $r \in \{0, 1\}^\lambda$ to perform following operations:

Step 1: computing $\beta_i = \mathbf{F}_3(k_e, r, i)$ for $i = 1, 2, \dots, n-1$ and a mask vector

$$\bar{\mathbf{m}} = \sum_{i=1}^{n-1} \beta_i \bar{\mathbf{m}}_i. \quad (6)$$

Step 2: computing

$$p = \sum_{i=1}^{n-1} \beta_i p_i, \quad (7)$$

$$\bar{\mathbf{c}} = \bar{\mathbf{e}} + \bar{\mathbf{m}}.$$

Step 3: sending the response message $\text{resp} = (\langle \bar{\mathbf{c}}, (r, p) \rangle, e_{n-1}, e_n, t)$ to TPA.

- (c) The TPA computes $\mathbf{g}_e$ according to chal , extracts $\mathbf{c} = (\bar{\mathbf{c}}, e_{n-1}, e_n, \mathbf{g}_e)$, and then verifies if the equation holds.

$$\mathbf{r} \cdot \mathbf{c} = t + p. \quad (8)$$

If so, $\mathbb{N}_s$ passes the auditing in that time and outputs 1; otherwise, it outputs 0.

4. Security Analysis on NC-Audit

The authors in [11] have presented the scheme NC-Audit with ciphertext indistinguishability under chosen plaintext attacks. However, it is not enough to guarantee the security of the auditing mechanism. As claimed, the vector $\bar{\mathbf{r}}$ (or $\mathbf{r}$)

can only be shared between the user and the TPA, which means that once $\bar{\mathbf{r}}$ is leaked to an adversarial storage node, the adversarial storage node is able to forge at least one illegal vector to pass the auditing verification when q is large. The following analysis shows that how an adversary can deduce the private vector $\bar{\mathbf{r}}$ easily.

According to equation (3), we can construct a system of linear equations about vector $\bar{\mathbf{r}}$ (with $n-2$ unknowns) as follows:

$$\bar{\mathbf{r}} \cdot (\bar{\mathbf{p}}_1^T, \bar{\mathbf{p}}_2^T, \dots, \bar{\mathbf{p}}_{n-1}^T) \triangleq \bar{\mathbf{r}} \cdot \mathbf{A} = (p_1, p_2, \dots, p_{n-1}). \quad (9)$$

Since $\bar{\mathbf{p}}_i \in \mathbb{F}_q^{n-2}$, we can see that $\mathbf{A}$ is clearly a $(n-2) \times (n-1)$ dimensional matrix in $\mathbb{F}_q$. Meanwhile, every element of $\mathbf{A}$ can be considered random in $\mathbb{F}_q$ because it is generated by a PRF, so the probability that the rank of the matrix $\mathbf{A}$ is $n-2$ can be high up to 1 when q is large enough. This means that the adversary has the ability to solve the vector $\bar{\mathbf{r}}$ with a high probability.

According to equation (8), we then have

$$\mathbf{r} \cdot \mathbf{c} = \bar{\mathbf{r}} \cdot \bar{\mathbf{c}} + \sum_{i=n-1}^{n+m} e_i \mathbf{F}_1(k_v, \text{id}, i), \quad (10)$$

$$= \bar{\mathbf{r}} \cdot (\bar{\mathbf{c}} + \bar{\mathbf{m}}) + \sum_{i=n-1}^{n+m} e_i \cdot \mathbf{F}_1(k_v, \text{id}, i). \quad (11)$$

As stated above, the adversary can solve the vector $\bar{\mathbf{r}}$ with a high probability when q is large enough. Moreover, he can indeed know the public vectors $\bar{\mathbf{c}}$. Therefore, the adversary can easily deduce at most $q^{n-3} - 1$ vectors $\bar{\mathbf{c}}' (\neq \bar{\mathbf{c}})$ satisfying

$$\bar{\mathbf{r}} \cdot \bar{\mathbf{c}}' = \bar{\mathbf{r}} \cdot \bar{\mathbf{c}}. \quad (12)$$

Since the vector $\bar{\mathbf{c}}$ is fixed during each encryption in NC-Audit, it holds that

$$\bar{\mathbf{r}} \cdot (\bar{\mathbf{c}} + \bar{\mathbf{m}}') = \bar{\mathbf{r}} \cdot (\bar{\mathbf{c}} + \bar{\mathbf{m}}). \quad (13)$$

According to equations (10) and (11), we have

$$\bar{\mathbf{r}} \cdot (\bar{\mathbf{c}} + \bar{\mathbf{m}}) + \sum_{i=n-1}^{n+m} e_i \cdot \mathbf{F}_1(k_v, \text{id}, i) = t + p, \quad (14)$$

which means

$$\bar{\mathbf{r}} \cdot \bar{\mathbf{c}}' = t + p, \quad (15)$$

still holds regarding the forged vector $\bar{\mathbf{m}}'$. That is, the adversary can elaborately disguise an illegal plaintext vector $\bar{\mathbf{m}}' (\neq \bar{\mathbf{m}})$ with the specific vector $\bar{\mathbf{c}}$ and the tag t , such that the forged response message $\text{resp}' = (\langle \bar{\mathbf{c}} + \bar{\mathbf{m}}', (r, p) \rangle, e_{n-1}, e_n, t)$ can succeed in passing the auditing verification.

5. An Improved Remote Data Integrity Checking Scheme for Distributed Storage

According to the algebraic analysis in Section 4, we can conclude that the security of NC-Audit depends on the security of the secret vector $\bar{\mathbf{r}}$, and vice versa. Inspired by the

notion of algebraic security in [20], an alternative and feasible method of realizing secure integrity checking is to prevent an adversary to deduce the vector $\bar{\mathbf{r}}$. For example, if the adversary can get less than $n-3$ values out of $p_i (i = 1, 2, \dots, n-1)$, there is no way for him to launch such attack as in Section 4.

Subsequently, we present an algebraic security-based auditing scheme called RNC-Audit, where a special randomization is performed in order to protect partial critical security parameters, which makes the auditing protocol satisfy the algebraic security criterion, i.e., the adversary has no ability to solve the linear system of equation constructed from his known information. To attain this end, without loss of generality, we assume that the values p_{n-2} and p_{n-1} in equation (9) have been randomized to prevent the adversary's security analysis in the following section.

Besides the PRFs $\mathbf{F}_1$ and $\mathbf{F}_2$ used in NC-Audit, RNC-Audit also introduces another PRF:

$$\mathbf{F}_4: \mathbf{K} \times \mathbf{WID} \times \mathbb{Z}^+ \longrightarrow \mathbb{F}_q, \quad (16)$$

where $\mathbf{WID}$ is the set of identifiers of auditing tasks. Seeing that the system parameters of the protocol are distinct from that of NC-Audit, RNC-Audit will perform different auditing computation.

The detailed execution of RNC-Audit between the user and a single CSP storage node is highlighted as follows.

(1) Initialization:

- (a) Setting security parameters λ and PRFs, then the user shares a unique key k_e and k_v with the storage node and the TPA, respectively, where k_e is used for encryption at the storage node and k_v is for verification at the TPA.
- (b) Both the user and the TPA compute the vectors $\bar{\mathbf{r}} \in \mathbb{F}_q^{n-2}$ and $\mathbf{r} \in \mathbb{F}_q^{n+m}$ as in NC-Audit.
- (c) The user generates the vector $\bar{\mathbf{p}}_i \in \mathbb{F}_q^{n-2}$ as in NC-Audit, then computes and sends the parameters $p_1, p_2, \dots, p_{n-3}, \alpha_1, \alpha_2$ to $\mathbb{N}_s$ (rather than the former parameters $p_1, p_2, \dots, p_{n-1}$ used in NC-Audit), and simultaneously sends θ_1, θ_2 to TPA, where

$$\begin{aligned} \theta_1 &= \alpha_1^{-1} p_{n-2}, \\ \theta_2 &= \alpha_2^{-1} p_{n-1}, \end{aligned} \quad (17)$$

in which α_1 and α_2 are both selected randomly in $\mathbb{F}_q^*$.

(2) Outsourcing upload:

For $j = 1, 2, \dots, M$, the user works as follows.

- (a) Generate $\mathbf{y}_{sj}$ as in Section 2.1 and compute

$$t_{sj} = \mathbf{y}_{sj} \cdot \mathbf{r} \in \mathbb{F}_q, \quad (18)$$

as the authentication tag of the vector $\mathbf{y}_{sj}$.

- (b) Retain his secret key and $\mathbf{g}_{y_{sj}}$, send the vector $\mathbf{g}_{y_{sj}}$ to TPA, and upload the vector $\mathbf{z}_{sj} = (\mathbf{y}_{sj}, t_{sj})$ to $\mathbb{N}_s$. Usually, the user can delete the whole data of the file denoted by id.

(3) Data auditing:

(a) The TPA generates and sends the challenge message $\text{chal} = \{|\langle i, \varepsilon_i \rangle| i \in \Delta\}$.

(b) $\mathbb{N}_s$ computes the aggregate vector $\mathbf{e}$, i.e.,

$$\mathbf{e} = \sum_{i \in \Delta} \varepsilon_i \mathbf{z}_{si} = (\bar{\mathbf{e}}, e_{n-1}, e_n, \mathbf{g}_e, t), \quad (19)$$

and then performs the following operations:

Step 1: generating $\beta_i = F_4(k_e, \text{wid}, i)$, $i = 1, 2, \dots, n-1$ and computing a mask vector

$$\bar{\mathbf{m}} = \sum_{i=1}^{n-1} \beta_i \bar{\mathbf{m}}_i, \quad (20)$$

where $\text{wid} \in \mathbf{WID}$ is the identifier to label the current audit task.

Step 2: computing

$$\begin{aligned} \nu &= \sum_{i=1}^{n-3} \beta_i p_i, \\ \bar{\mathbf{c}} &= \bar{\mathbf{e}} + \bar{\mathbf{m}}. \end{aligned} \quad (21)$$

Step 3: sending the response message $\text{resp} = (\langle \bar{\mathbf{c}}, \nu \rangle, e_{n-1}, e_n, t, \gamma_1, \gamma_2)$ to TPA, where

$$\begin{aligned} \gamma_1 &= \beta_{n-2} \alpha_1, \\ \gamma_2 &= \beta_{n-1} \alpha_2. \end{aligned} \quad (22)$$

(c) TPA computes $\mathbf{g}_e$ according to chal , extracts $\mathbf{c} = (\bar{\mathbf{e}}, e_{n-1}, e_n, \mathbf{g}_e)$ from resp , and computes

$$\begin{aligned} \delta_1 &= \theta_1 \gamma_1, \\ \delta_2 &= \theta_2 \gamma_2, \end{aligned} \quad (23)$$

and then verifies if the following equation holds.

$$\mathbf{r} \cdot \mathbf{c} = t + \nu + \delta_1 + \delta_2. \quad (24)$$

If so, $\mathbb{N}_s$ passes the auditing in that time and outputs 1; otherwise, it outputs 0.

6. Correctness of RNC-Audit

The correctness of RNC-Audit is guaranteed if the file is corrected by the following derivation process.

Let $\mathbf{r} = (r_1, r_2, \dots, r_{n+m}) = (\bar{\mathbf{r}}, r_{n-1}, \dots, r_{n+m})$, where $\bar{\mathbf{r}} = (r_1, \dots, r_n)$.

$$\begin{aligned} \mathbf{r} \cdot \mathbf{c} &= \mathbf{r} \cdot (\bar{\mathbf{c}}, e_{n-1}, e_n, \mathbf{g}(\mathbf{e})) \\ &= \mathbf{r} \cdot (\bar{\mathbf{e}} + \bar{\mathbf{m}}, e_{n-1}, e_n, \mathbf{g}(\mathbf{e})) \\ &= \mathbf{r} \cdot (\bar{\mathbf{e}}, e_{n-1}, e_n, \mathbf{g}(\mathbf{e})) + (\bar{\mathbf{r}}, r_{n+1}, \dots, r_{n+m}) \cdot (\bar{\mathbf{m}}, \mathbf{0}_{m+2}) \\ &= t + \bar{\mathbf{r}} \cdot \bar{\mathbf{m}} \\ &= t + \sum_{i=1}^{n-1} \beta_i \bar{\mathbf{r}} \cdot \bar{\mathbf{p}}_i = t + \sum_{i=1}^{n-3} \beta_i \bar{\mathbf{r}} \cdot \bar{\mathbf{p}}_i + \sum_{i=n-2}^{n-1} \beta_i \bar{\mathbf{r}} \cdot \bar{\mathbf{p}}_i \\ &= t + \sum_{i=1}^{n-3} \beta_i p_i + \sum_{i=n-2}^{n-1} \beta_i p_i \\ &= t + \nu + \delta_1 + \delta_2, \end{aligned} \quad (25)$$

where $\mathbf{0}_{m+2}$ is a zero vector of size $m+2$.

7. Performance Analysis

7.1. Security Guarantee. Similar to the analysis in Theorems 2 and 4 in [11], it can be easily proven that RNC-Audit can provide data possession proof and privacy-preserving guarantee.

Especially, RNC-Audit can effectively conquer the security weakness existing in NC-Audit in Section 4. The user has the ability to protect the values p_{n-2}, p_{n-1} against both the honest-but-curious server and the TPA, which makes the adversary to only at most obtain a linear system with $n-3$ equations with respect to $n-2$ unknowns as follows:

$$\bar{\mathbf{r}} \cdot (\bar{\mathbf{p}}_1^T, \bar{\mathbf{p}}_2^T, \dots, \bar{\mathbf{p}}_{n-3}^T) \triangleq \bar{\mathbf{r}} \cdot \mathbf{A}' = (p_1, p_2, \dots, p_{n-3}). \quad (26)$$

It is easy to see that when q is large properly, the adversary has no ability to solve the vector $\bar{\mathbf{r}}$ except the brute force guess, which can effectively resist the adversary analysis as in Section 4 and thus guarantee the auditing security of RNC-Audit. Note that we can easily conclude that the proposed scheme realizes the algebraic security as defined in [20].

7.2. Computation and Communication. Under the same security level, the scheme NC-Audit and the scheme in [10] were once considered to be efficient in computation performance among the current data auditing schemes for regenerating-coding-based storage. Therefore, the following text will compare the performance of RNC-Audit with that of the schemes in [10] and NC-Audit. The systematic performance features of the three schemes are shown in Table 1.

TABLE 1: Systematic performance comparison of three typical schemes.

Scheme	NC-Audit	Ref. [10]	RNC-Audit
Auditing privacy	No	Yes	Yes
Communication overhead	$\mathcal{O}(n)$	$\mathcal{O}(m)$	$\mathcal{O}(n)$
Computation cost (TPA)	$\mathcal{O}(m+n)$	$\mathcal{O}(nm)$	$\mathcal{O}(m+n)$
Computation cost (CSP)	$\mathcal{O}(n \cdot \Delta)$	$\mathcal{O}(n \cdot \Delta)$	$\mathcal{O}(n \cdot \Delta)$

The scheme in [10] is the one specifically constructed for online user auditing, which differs in system model from the existing schemes because the entity TPA is never needed. This design simplifies the protocol process and eliminates the security risks from the TPA, but the incidental side effect is that the user itself must be always online and it incurs high processing burden. Furthermore, the scheme involves large-scale error correction encoding and channel decoding operations, which makes the computation overhead more expensive compared to RNC-Audit and NC-Audit.

Although NC-Audit is also excellent in computation cost, it cannot guarantee user's data privacy. In contrast, RNC-Audit can ensure data privacy and achieve similar implementation performance as NC-Audit. Actually, RNC-Audit only operates two more multiplications in $\mathbb{F}_q$ compared to NC-Audit during each audit. In addition, it does not increase the storage overhead of every entity in the system during the protocol running.

For each audit round, the main communication overhead is the response transmission from the storage node to the TPA (or the user), which is dominated by the size of the (encrypted) data block. NC-Audit and RNC-Audit also keep the equivalent performance in communication efficiency. Comparatively, the scheme in [10] has lower communication overhead because of its simplified auditing method.

7.3. Implementation. We implement in C to compare the online computation performance of RNC-Audit with two typical schemes, i.e., NC-Audit and the scheme in [10]. The experimental result reported for these three schemes is the average of 1000 runs on a computer with Intel(R) Core (TM) i7-8650U 1.9 GHz and 16G RAM.

For a fair comparison, we set $\lambda = 80$, $q = 2^8$, $n = 2^{12}$ (i.e., 4 kB block), $m = 200$, and $M = 300$. The experiment ignores the addition operation and focuses on the online processing time of multiplications over $\mathbb{F}_q$ with the help of lookup table. Table 2 shows the average computation performance of the TPA and one CSP server (both have the same configuration) during one auditing work, when $|\Delta| = 200$ and 300. $t_{i,j}$ represents the average online computing time of entity i when $j = |\Delta|$. The result shows that RNC-Audit outperforms the other two schemes in execution efficiency. Since the scheme in [10] needs to additionally perform complex decoding operations, the user-side computation overhead is relatively large, but the server-side overhead is small. Among the three schemes under comparison, the computational time always increases with $|\Delta|$.

Taking the above discussions into account, the proposed scheme RNC-Audit is excellently customized to achieve

TABLE 2: Online computation time comparison of three typical audit schemes.

Scheme	$t_{\text{TPA},200}$	$t_{\text{CSP},200}$	$t_{\text{TPA},300}$	$t_{\text{CSP},300}$
NC-Audit	5.55 ms	2.13 ms	6.44 ms	2.77 ms
Ref. [10]	11.09 ms	3.14 ms	13.73 ms	3.14 ms
RNC-Audit	5.54 ms	2.13 ms	6.40 ms	2.72 ms

providing proof of retrievability and privacy-preserving auditing without any data security compromise while potentially being efficient in real-time applications and thus much usable for the regenerating-coding-based distributed storage.

8. Conclusion

Privacy-preserving data auditing is one of the key issues in distributed cloud storage applications. This paper first points out that there exists a fatal security flaw in the scheme NC-Audit [11] according to our algebraic analysis. Inspired by the algebraic security, an improved PoR-based scheme called RNC-Audit is presented, which can not only effectively prevent the algebraic analysis but also maintain the competitive implementation efficiency in coding-based storage systems. Our analysis and evaluation results demonstrate that RNC-Audit is more efficient and usable than the state-of-the-art schemes in practical resource-constrained scenarios.

The ongoing work targets the application extensions of our research methodology from regenerating-coding-based storage to privacy-preserving forwarding [21], security-as-service frameworks [22], mobile crowdsensing [23, 24], authorized deduplication [25, 26], and the adoption of a stronger threat model involving malware detection [16, 18, 27].

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This study was supported by the National Natural Science Foundation of China (nos. 61702105 and 61872088), Natural Science Foundation of Fujian Province (no. 2019J01276), Natural Science Foundation of Shaanxi Province (no. 2021JQ-196), China Postdoctoral Science Foundation (no. 2019M663629), Xi'an Science and Technology Project (no. 2020KJWL02), and China National Study Abroad Fund Grant.

References

- [1] B. Chen, R. Curtmola, and G. Ateniese, "Remote data checking for network coding-based distributed storage

- systems,” in *Proceedings of the 2010 ACM Workshop on Cloud Computing Security Workshop*, pp. 31–42, New York, NY, USA, October 2010.
- [2] L. Zhou, A. Fu, S. Yu, M. Su, and B. Kuang, “Data integrity verification of the outsourced big data in the cloud environment: a survey,” *Journal of Network and Computer Applications*, vol. 122, pp. 1–15, 2018.
 - [3] 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*, pp. 598–609, Alexandria, VA, USA, November 2007.
 - [4] S. D. Khambalkar, S. D. Kamble, N. V. Thakur, N. U. Sambhe, and N. S. Mangrulkar, “An overview on privacy preservation and public auditing on outsourced cloud data,” in *Smart Innovation, Systems and Technologies*, pp. 463–470, Springer, Singapore, 2020.
 - [5] J. Liu, K. Huang, and H. Rong, “Privacy-preserving public auditing for regenerating-code-based cloud storage,” *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 7, pp. 1513–1528, 2015.
 - [6] C. Wang, Q. Wang, and K. Ren, “Privacy-preserving public auditing for data storage security in cloud computing,” in *Proceedings of the IEEE Infocom 2010*, pp. 1–9, IEEE, San Diego, CA, USA, March 2010.
 - [7] H. C. H. Chen and P. P. C. Lee, “Enabling data integrity protection in regenerating-coding-based cloud storage: theory and implementation,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 2, pp. 407–416, 2013.
 - [8] Z. Ren, L. Wang, and Q. Wang, “Dynamic proofs of retrievability for coded cloud storage systems,” *IEEE Transactions on Services Computing*, vol. 11, no. 4, pp. 685–698, 2015.
 - [9] K. He, C. Huang, and J. Shi, “Public integrity auditing for dynamic regenerating code based cloud storage,” in *Proceedings of the 2016 IEEE Symposium on Computers and Communication (ISCC)*, pp. 581–588, IEEE, Messina, Italy, June 2016.
 - [10] V. S. Lakshmi and P. P. Deepthi, “A secure regenerating code-based cloud storage with efficient integrity verification,” *International Journal of Communication Systems*, vol. 32, no. 9, p. e3948, 2019.
 - [11] A. Le, A. Markopoulou, and A. G. Dimakis, “Auditing for distributed storage systems,” *IEEE/ACM Transactions on Networking*, vol. 24, no. 4, pp. 2182–2195, 2015.
 - [12] H. C. H. Chen, Y. Hu, and P. P. C. Lee, “NCCloud: a network-coding-based storage system in a cloud-of-clouds,” *IEEE Transactions on Computers*, vol. 63, no. 1, pp. 31–44, 2013.
 - [13] W. F. Hsien, C. C. Yang, and M. S. Hwang, “A survey of public auditing for secure data storage in cloud computing,” *IJ Network Security*, vol. 18, no. 1, pp. 133–142, 2016.
 - [14] C. B. Tan, M. H. A. Hijazi, Y. Lim, and A. Gani, “A survey on proof of retrievability for cloud data integrity and availability: cloud storage state-of-the-art, issues, solutions and future trends,” *Journal of Network and Computer Applications*, vol. 110, pp. 75–86, 2018.
 - [15] F. Chen, F. Meng, T. Xiang, H. Dai, J. Li, and J. Qin, “Towards usable cloud storage auditing,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 31, no. 11, pp. 2605–2617, 2020.
 - [16] M. Alazab, S. Huda, and J. Abawajy, “A hybrid wrapper-filter approach for malware detection,” *Journal of Networks*, vol. 9, no. 11, pp. 2878–2891, 2014.
 - [17] F. SubhanW. Z. Khan et al., “A systematic review on clone node detection in static wireless sensor networks,” *IEEE Access*, vol. 8, pp. 65450–65461, 2020.
 - [18] D. Vasan, M. Alazab, S. Wassen, H. Naeem, B. Safaei, and Q. Zheng, “IMCFN: image-based malware classification using fine-tuned convolutional neural network architecture,” *Computer Networks*, vol. 171, Article ID 107138, 2020.
 - [19] A. G. Dimakis, P. B. Godfrey, Y. Wu, M. J. Wainwright, and K. Ramchandran, “Network coding for distributed storage systems,” *IEEE Transactions on Information Theory*, vol. 56, no. 9, pp. 4539–4551, 2010.
 - [20] L. Lima, M. Médard, and J. Barros, “Random linear network coding: a free cipher?” in *Proceedings of the IEEE 2007 International Symposium on Information Theory*, pp. 546–550, Nice, France, June 2007.
 - [21] C. Borrego, M. Amadeo, A. Molinaro, and R. H. Jhaveri, “Privacy-preserving forwarding using homomorphic encryption for information-centric wireless Ad hoc networks,” *IEEE Communications Letters*, vol. 23, no. 10, pp. 1708–1711, 2019.
 - [22] G. T. Reddy, K. Sudheer, and K. Rajesh, “Employing data mining on highly secured private clouds for implementing a security-as-a-service framework,” *Journal of Theoretical and Applied Information Technology*, vol. 59, no. 2, pp. 317–326, 2014.
 - [23] J. Xiong, H. Liu, and B. Jin, “A lightweight privacy protection scheme based on user preference in mobile crowdsensing,” *Transactions on Emerging Telecommunications Technologies*, vol. 40, p. e4000, 2020.
 - [24] J. Xiong, R. Ma, and L. Chen, “A personalized privacy protection framework for mobile crowdsensing in IIoT,” *IEEE Transactions on Industrial Informatics*, vol. 16, no. 6, pp. 4231–4241, 2019.
 - [25] J. Xiong, Y. Zhang, S. Tang, X. Liu, and Z. Yao, “Secure encrypted data with authorized deduplication in cloud,” *IEEE Access*, vol. 7, pp. 75090–75104, 2019.
 - [26] J. Xiong, Y. Zhang, X. Li, M. Lin, Z. Yao, and G. Liu, “RSE-PoW: a role symmetric encryption PoW scheme with authorized deduplication for multimedia data,” *Mobile Networks and Applications*, vol. 23, no. 3, pp. 650–663, 2018.
 - [27] S. Venkatraman and M. Alazab, “Use of data visualisation for zero-day malware detection,” *Security and Communication Networks*, vol. 2018, Article ID 1728303, , 2018.