

Article

Keyword-Based Medical Cloud Storage Integrity Auditing with Privacy Protection and Data Dynamics

Meijuan Huang ¹ , Siyu Chen ¹, Bo Yang ^{2,3,*} and Xiaoyan Zhao ¹
¹ School of Mathematics and Information Science, Baoji University of Arts and Sciences, Baoji 721013, China; huangmeijuan@bjwlxy.edu.cn (M.H.); chensiyu@bjwlxy.edu.cn (S.C.); zhaoxiaoyan@bjwlxy.edu.cn (X.Z.)

² School of Computer Science, Shaanxi Normal University, Xi'an 710119, China

³ Koal Software Co., Ltd., Shanghai 201601, China

* Correspondence: byang@snnu.edu.cn

Abstract

Keyword-based remote integrity auditing schemes effectively address the integrity of electronic medical records (EMRs) stored in the cloud. In practice, users expect to be able to perform flexible dynamic data updates while also protecting data privacy against a third-party auditor during the auditing process. However, existing schemes fail to simultaneously satisfy both requirements: they either incur prohibitive overhead for block-level updates or disclose to the auditor which EMRs match the target keyword and the number of such EMRs. To address this, we propose a new keyword-based auditing scheme for medical cloud. Specifically, we design a novel authentication identifier set. Unlike the keyword tags in Shen et al.'s scheme, this set aggregates the block hashes and thereby enables the auditor to verify integrity without obtaining sensitive information. Furthermore, we introduce a dynamic hash list. By updating this list during block insertion and deletion, the scheme eliminates the need to recompute the authenticators of subsequent blocks, significantly enhancing the efficiency of dynamic data updates. Security analysis confirms that the proposed scheme is secure. Performance analysis shows our scheme reduces block insertion and deletion overhead by over 60% compared to Gao et al.'s scheme, demonstrating high efficiency and practicality.

Keywords: cloud storage; privacy protection; dynamic data updates; data auditing

1. Introduction

Electronic medical records (EMRs) have become an integral component of medical systems, and the surge in data volume has prompted hospitals to outsource them to cloud servers (CSs) for storage. However, while outsourcing data to the cloud alleviates local storage burdens, it also introduces a spectrum of security risks [1]: users lose physical control over their outsourced data, and the explosive growth in data volumes substantially increases the potential for data leakage. In particular, the loss of users' physical control over their data implies that the cloud server may either arbitrarily delete data to conserve storage resources or become compromised by malicious attacks, both of which could severely impair clinical diagnosis and medical research [2]. To mitigate this modern network security risk in cloud data, periodic integrity verification can ensure data integrity and availability. Traditional data integrity auditing schemes need to verify all EMRs in the cloud, which brings a heavy financial burden to hospitals. In practice, hospitals often focus on the integrity of EMRs only partially, such as the EMRs of current patients or a specific



Academic Editor: Kamel Barkaoui

Received: 2 June 2026

Revised: 22 July 2026

Accepted: 26 July 2026

Published: 29 July 2026

Copyright: © 2026 by the authors.

Licensee MDPI, Basel, Switzerland.

This article is an open access article distributed under the terms and conditions of the [Creative Commons Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

disease. Therefore, a scheme that supports keyword-based auditing is needed: hospitals submit a keyword, and the CS generates an auditing proof only for all EMRs matching the target keyword, which is verified by a third-party auditor (TPA). This can significantly reduce the auditing cost.

The EMRs contain a vast amount of sensitive private patient information, including real names, contact numbers and residential addresses, thereby exposing these cloud-stored EMRs to severe risks of data leakage [3]. Hospitals must preprocess the data to erase patients' sensitive information and generate a search token for each keyword before uploading them. However, recent empirical studies demonstrate that adversaries can still identify patients via frequency analysis and quasi-identifiers (e.g., medical visit statistics) even with preprocessed electronic medical records [4]. Meanwhile, sensitive information, such as rare diseases and visit popularity, can also be inferred from the number of EMRs associated with a specific keyword. Therefore, for the integrity verification of all EMRs matching the target keyword, hospitals only need to supply a TPA with the corresponding search token. The TPA is not allowed to learn the intrinsic association (i.e., quasi-identifiers) between the target keyword and the EMRs. Specifically, it cannot know which EMRs are matching the target keyword or the number of such EMRs. Furthermore, attending physicians need to update the cloud-stored EMRs according to the dynamic changes in treatment effect, drug dosage, and complications during hospitalization [5]. Dynamic operations on cloud-stored data include block-level and file-level updates.

In recent years of research on keyword-based auditing, Xue et al. [6] adopted fuzzy matching techniques to conceal auditing frequencies from CSs. Zhang et al. [7] utilize the Block Version State Table (BVST) structure to reduce the complexity of data version backtracking and resist backtracking attacks. Yet they overlook the necessity of concealing the intrinsic association between the target keyword and EMRs from TPAs. Gao et al. [8] proposed a Relation Authentication Label to satisfy the privacy protection requirement against curious TPAs. However, all these schemes only support file-level updates and cannot achieve efficient block-level updates. The root cause lies in the fact that block authenticators embed corresponding block indexes, which shift along with the physical positions of data blocks. Consequently, physicians have to recalculate authenticators for all subsequent blocks whenever data blocks are inserted or deleted. Obviously, repeated recalculations during block-level updates introduce tremendous computational overhead. Shen et al. [9] introduced an authenticator index to improve the efficiency of block-level updates. However, their scheme fails to provide privacy protection against TPAs. This is because, in previous approaches, when block indexes were used to generate authenticators, data blocks at the same location in different files shared the same public block indexes. After the authenticator index is introduced to generate authenticators, each data block has its own exclusive authenticator index. Data blocks at the same location in different files may have different authenticator indexes after block-level operations. In the proof verification, the TPA retrieves the authenticator indexes of the blocks being challenged and calculates their product. That is, the intrinsic association between the target keyword and the EMRs is revealed to TPAs during the verification process.

In practical cloud-based electronic medical record systems, hospitals need to ensure patient data privacy while achieving efficient block-level updates of EMRs in the cloud. The above keyword-based auditing schemes cannot satisfy both requirements simultaneously. Therefore, studying a keyword-based medical cloud storage auditing scheme with privacy protection and dynamic data updates is of great importance.

We summarize our contributions as follows:

- (1) We present a more efficient and practical keyword-based auditing scheme for medical cloud storage. In this scheme, we innovatively design an authentication identifier

set for each keyword. The authentication identifier contains the file identifiers of all EMRs matching the target keyword. Based on this set, TPAs can validate the integrity of all EMRs matching the target keyword, without learning the intrinsic association between the target keyword and the EMRs. Furthermore, we introduce a dynamic hash list to support efficient block-level updates. Block insertion and block deletion only need to update the list synchronously, without regenerating the authenticators of subsequent blocks, which significantly enhances update efficiency.

- (2) Compared with existing keyword-based auditing schemes, the authentication identifier set we introduce offers two major advantages. First, unlike Shen et al. [9]’s scheme, it aggregates the block hashes of all EMRs matching the target keyword, preventing TPAs from learning the intrinsic association between the keyword and the EMRs. Second, in contrast to Gao et al. [8]’s scheme, it is independent of the block index, thus avoiding recomputation of subsequent authenticators during block insertion/deletion. Consequently, our scheme simultaneously supports privacy protection against TPAs and efficient block-level dynamic updates.
- (3) We provide specific security analysis to verify that our scheme satisfies correctness, auditing reliability, and privacy protection against TPAs. Rigorous performance evaluation demonstrates that our scheme is highly efficient and practical.

1.1. Related Work

As data volumes continue to expand, users increasingly outsource their data to cloud servers. While this mitigates local storage burdens, it introduces critical security risks [1]: users forfeit physical control over their data, and the surge in data volume exacerbates leakage vulnerabilities. To address the issue of users losing control over their data, the Provable Data Possession (PDP) scheme was first proposed in [10]. By leveraging homomorphic signatures and random sampling, this scheme enables blockless verification. Since then, the PDP scheme has been widely adopted to verify the integrity of cloud-stored data.

Numerous remote data integrity auditing schemes have been proposed to meet diverse requirements. As cloud data grows, user auditing burdens intensify. Wang et al. [11] introduced public verification to delegate heavy tasks to third-party auditors. To satisfy user preferences for trusted auditors, designated verifier schemes have emerged [12–14]. To mitigate key leakage, Yu et al. [15] employed pre-order traversal and binary trees for key updates, albeit only supporting forward security; they [16] later proposed a strong key exposure resilient scheme in 2016. To avoid data redundancy, Yuan et al. [17] proposed a blockchain-based auditing protocol for deduplication and fair arbitration. Yu et al. [18] designed a privacy-preserving deduplication auditing protocol for cloud EMRs using a filter to reduce storage overhead. For resource-constrained mobile devices, proxy-based schemes [19–21] have been proposed to alleviate computational burdens. Regrettably, all the above schemes adopt the full-data auditing mode, incurring excessive financial costs for users.

The escalating risks of data leakage have prompted extensive research into approaches for protecting cloud data privacy. Wang et al. [22] designed the first public auditing protocol that uses a random mask technique to achieve privacy protection. To conceal sensitive information, Yang et al. processed sensitive data blocks by employing a third-party sanitizer in the scheme [23]. Ling et al. [21] proposed an assistant-based transparent auditing protocol, which employs blockchain technology to achieve TPA-side privacy preservation during cloud auditing. Other privacy-preserving schemes include those presented in [24–26]. Nevertheless, these traditional auditing paradigms not only lack effective privacy preservation mechanisms for keyword-targeted auditing, but also ignore the heavy computational burden brought by block-level dynamic updates. To satisfy users’

demands for dynamic data updates, Erway et al. [27] designed the first data integrity auditing protocol with update capabilities. Liu et al. [28] presented a hierarchical Merkle-tree-based auditing scheme supporting dynamic data updates, which improved the update efficiency by shortening the authentication path. Wang et al. [29] extended the rank-based Merkle hash tree using a non-leaf node sampling strategy, thereby achieving dynamic updates and reducing the communication overhead. In the scheme [30], Zheng et al. proposed an auditing protocol supporting secure deduplication and data dynamics, which reduces the computational overhead of dynamic updates via lightweight authenticators and multi-set hash functions. Despite promising optimization results for dynamic data updates, all such schemes are limited to conventional auditing architectures and lack support for hiding sensitive data against TPAs.

In recent years, researchers have integrated searchable encryption with data integrity auditing, utilizing keyword trapdoors to selectively verify matched files instead of conventional auditing models. Wang et al. [31] proposed a searchable anonymous authorized auditing scheme, which empowers designated TPAs to audit outsourced data integrity while preserving keyword privacy. Yu et al. [32] developed a multi-keyword searchable data auditing scheme, which enables TPAs to verify the integrity of outsourced training data while guaranteeing keyword privacy. Miao et al. [33] devised a searchable auditing protocol for multi-replica data, which employs blockchain to ensure the integrity of search results and guarantees auditing credibility. To reduce users' payment costs, Sengupta et al. [34] pioneered a keyword-based public auditing scheme. The scheme enables TPAs to verify only EMRs matching the target keyword without checking all data. Xue et al. considered the issue of concealing auditing frequency from cloud servers in the scheme [6]. Zhang et al. [7] proposed the DKCIA-B framework, which reduces the complexity of historical data version backtracking while providing robustness against backtracking attacks. The above schemes break through the limitations of conventional auditing models and realize searchable auditing/keyword-based auditing, which effectively reduces the heavy overhead incurred by traditional full-data auditing and protects keyword privacy against TPAs to a certain degree. Nevertheless, all these schemes fail to address two critical issues: they cannot conceal the intrinsic association between the target keyword and corresponding files from TPAs, and they suffer from low efficiency during block-level dynamic data updates.

To hide sensitive information from TPAs, Gao et al. [8] developed a keyword-based checking protocol that prevents TPAs from learning the intrinsic association, yet it merely supports file-level updates. Shen et al. [9] designed a keyword-based auditing scheme that supports efficient dynamic data operations. However, to achieve efficient block-level updates, the scheme [9] used authenticator indexes instead of block indexes to generate block authenticators. During proof verification, TPA needs to retrieve the authenticator indexes of the blocks being challenged and compute their product. This means that TPAs must learn which EMRs are related to the target keyword or the number of such EMRs. Therefore, supporting both data dynamics and privacy protection in keyword-based medical cloud storage integrity auditing remains an open challenge.

1.2. Organization

Following this introduction, Section 2 provides the preliminaries and notations. Section 3 formulates the security model and system model. Section 4 presents the detailed construction of the practical keyword-based medical cloud storage auditing scheme. Section 5 analyzes the security of our scheme. Section 6 presents the experimental outcomes. Section 7 concludes the proposed scheme.

2. Notations and Preliminaries

2.1. Notations

Table 1 summarizes the main notations employed in our scheme.

Table 1. Notations.

Notation	Description
$\pi(\cdot)$	Pseudo-random permutation
$\mathbb{G}_i$	The i-th multiplicative cyclic group
$H_i : \{0, 1\}^* \rightarrow \mathbb{G}_1$	The i-th hash function
$e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$	A bilinear mapping
g, u	Two generators in $\mathbb{G}_1$
K	Number of keywords
w_k	The k-th keyword
τ_k	Search token for keyword w_k
M_i	The i-th EMR
M	The EMR set
MID_i	The identifier of M_i
MID	Identifier set for M
$m_{i,j}$	The j-th block of M_i
$\hat{M}_i$	The i-th preprocessed EMR
$\hat{M}$	The preprocessed EMR set
$\hat{m}_{i,j}$	The j-th block of $\hat{M}_i$
$\sigma_{i,j}$	Authenticator for block $\hat{m}_{i,j}$
Φ_i	Authenticator set for $\hat{M}_i$
$\mathcal{HL}_i$ ¹	Dynamic hash list for $\hat{M}_i$
$\mathcal{I}_{w_k}$	Inverted index set for keyword w_k
$\theta_{w_k,j}$ ²	The j-th authentication identity for keyword w_k
$Chal$	Auditing challenge
$Proof$	Auditing proof

¹ $\mathcal{HL}_i$ enables efficient block-level updates; ² $\theta_{w_k,j}$ hides internally between keywords and EMRs

2.2. Preliminaries

(1) Bilinear Mapping: Consider two distinct multiplicative cyclic groups $\mathbb{G}_1, \mathbb{G}_2$ of prime order p . A map $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ satisfying the following property is called a bilinear mapping:

- Bilinearity: $\forall h, z \in \mathbb{Z}_p^*$ and $\forall f, s \in \mathbb{G}_1, e(f^h, s^z) = e(f, s)^{hz}$.
- Computability: For $\forall f, s \in \mathbb{G}_1, e(f, s)$ can be computed by a known efficient algorithm.
- Non-degeneracy: $\exists f, s \in \mathbb{G}_1, e(f, s) \neq 1_{\mathbb{G}_2}$.

(2) Computational Diffie–Hellman (CDH) Problem: Given $g, g^h, g^z \in \mathbb{G}_1$ for $h, z \in \mathbb{Z}_p^*$, the CDH assumption asserts that computing $g^{hz} \in \mathbb{G}_1$ is computationally infeasible, namely:

$$\Pr[g^{hz} \leftarrow \mathcal{A}_{CDH}(g, g^h, g^z)] \leq \varepsilon$$

where ε represents a negligible function and $\mathcal{A}$ is a polynomial-time adversary.

(3) Discrete Logarithm (DL) Problem: Given $g, g^z \in \mathbb{G}_1$ for $z \in \mathbb{Z}_p^*$, the DL problem assumption asserts that computing z is computationally infeasible, namely:

$$\Pr[z \leftarrow \mathcal{A}_{DL}(g, g^z)] \leq \varepsilon$$

where ε represents a negligible function and $\mathcal{A}$ is a polynomial-time adversary.

3. System Model and Security Model

3.1. System Model

As illustrated in Figure 1, the system model includes four types of entities: CS, TPA, Data Owner (DO) and Hospital Information System Administrator (HIS administrator).

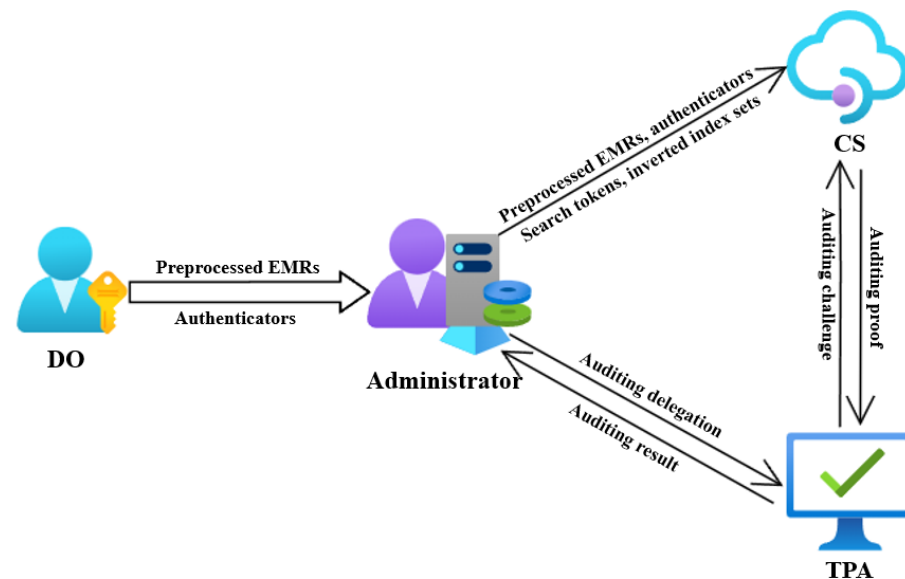


Figure 1. System model.

CS: It provides the hospital with extensive storage and computing resources. Upon receiving an auditing challenge from a TPA, it locates the relevant EMRs and returns a corresponding auditing proof. However, since cloud-stored EMRs may be damaged or lost for various reasons, the CS might forge the proof to preserve its reputation. Thus, it is considered a semi-trusted entity.

TPA: It possesses abundant computing resources and is authorized by the hospital to perform auditing tasks for all EMRs. The TPA verifies the integrity of relevant EMRs using the auditing proof returned by the CS. However, it may become curious about the intrinsic association between the target keyword and the EMRs, or it may repeatedly audit the same keyword to gather additional information. Therefore, it is considered a semi-trusted entity.

DO: The data owner typically refers to hospital medical staff, such as doctors, examiners, and anesthesiologists. The medical staff preprocess the EMRs and generate block authenticator sets for them. To support dynamic data updates and protect data privacy against the TPA, the preprocessed EMRs, along with their identifiers and block authenticator sets, are uploaded to the HIS administrator.

HIS administrator: The HIS administrator extracts keywords and inverted index sets from the preprocessed EMRs sent by the DO. After that, the administrator uploads the preprocessed EMRs, together with their identifiers, block authenticator sets, search token sets, and inverted index sets, to the CS. For each EMR, a dynamic hash list is calculated and stored by the administrator. The administrator updates the list as requested by the DO. As a hospital commissioner, the HIS administrator cannot access plaintext EMRs or the data owner's private key. Even if adversaries compromise them, they can hardly recover full sensitive patient data. Accordingly, we define this entity as trusted.

In practice, the HIS administrator is typically a designated hospital staff member responsible for managing the internal hospital information system. This entity operates within the hospital's administrative domain and is subject to the hospital's security policies, access controls, and auditing procedures. Therefore, treating the HIS administrator as

trusted is consistent with the actual deployment architecture of most hospital information systems.

Our scheme consists of six algorithms, which are as follows:

- (1) **Setup** (1^λ): Taking 1^λ , the algorithm outputs the DO's private key x , public key $y = g^x$ and public parameters $\mathbb{PP}$.
- (2) **Pretreat and AuthDen** (x, MID, M): Taking the DO's private key x , identifier set MID and EMR set M , the algorithm returns the preprocessed EMR set $\hat{M}$ and authenticator set $\{\Phi_i\}_{i=1}^n$.
- (3) **KeywordExt and IndexGen** ($\hat{M}$): Taking the preprocessed EMR set $\hat{M}$, the algorithm outputs the search token set $\{\tau_k\}_{k=1}^K$, inverted index set $\{\mathcal{I}_{w_k}\}_{k=1}^K$ and dynamic hash list $\{\mathcal{H}\mathcal{L}_i\}_{i=1}^n$.
- (5) **ChalGen** (w_k): Inputting the target keyword w_k , the algorithm outputs the authentication identity set $\{\theta_{w_k,j}\}_{j=1}^t$ and an auditing challenge $Chal$.
- (5) **ProofGen** ($\hat{M}, \{\Phi_i\}_{i=1}^n, \tau_k, Chal$): Inputting the preprocessed EMR set $\hat{M}$, authenticator set $\{\Phi_i\}_{i=1}^n$, search token τ_k and challenge $Chal$, the algorithm outputs an auditing proof $Proof$.
- (6) **ProofVerify** ($Chal, Proof, \{\theta_{w_k,j}\}_{j=1}^t$): Inputting the challenge $Chal$, proof $Proof$ and authentication identity set $\{\theta_{w_k,j}\}_{j=1}^t$, the algorithm outputs "1" if the proof is valid, and "0" otherwise.

3.2. Security Model

The security of our scheme is defined by an interactive game between a challenger C and an adversary A , as follows:

- (1) **Setup**: The algorithm is executed by C , which generates the private key x for DO and public parameters $\mathbb{PP}$ and transmits $\mathbb{PP}$ to A .
- (2) **Query phase**: A intends to query the authenticator of block $\hat{m}_{i,j}$. By executing the *AuthGen* algorithm, C obtains and returns the corresponding authenticator $\sigma_{i,j}$.
- (3) **Challenge phase**: In this step, the challenge $Chal = \{\tau_k, \{v_j, j\}_{j \in S}\}$ is generated by C and sent to A . Here, τ_k denotes the search token for keyword w_k , $v_j \in_R \mathbb{Z}_p^*$ and $j \in S \subseteq [1, t]$.
- (4) **Forgery phase**: Using the challenged blocks of $\{\hat{M}_i\}_{i \in \mathcal{I}_{w_k}}$, A generates a corresponding proof, $Proof$. A is said to win the game if its proof, $Proof$, passes the integrity verification with non-negligible probability.

Based on the aforementioned security model, the security of our scheme is defined by the following property: if A has not stored all challenged blocks, then the probability that its proof $Proof$ is accepted by C is negligible. The detailed proof process is presented in Section 5 (Security Analysis).

3.3. Threat Model

In our threat model, both CSs and TPAs are treated as potential attackers.

CS: It may delete infrequently accessed data without the DO's knowledge to optimize storage space. Furthermore, data may be corrupted or modified due to software or hardware failures. To protect its reputation, the CS attempts to fabricate a corresponding proof to deceive the TPA into accepting it during the auditing process.

TPA: The TPA is considered honest and reliable during the auditing process, and therefore does not actively attempt to launch leakage attacks such as access pattern or search pattern leakage. Since keywords are transmitted as randomized ciphertexts, we do not further consider the risk of keyword frequency leakage to TPA during the auditing process. However, the TPA may still be curious about the intrinsic association between the

keywords and the EMRs, and may try to recover the real data by repeatedly auditing the same keyword, which is referred to as a “replay attack” [35].

In the practical deployment of independent audit trails, the TPA typically acts as a regulated auditing institution, while the CS serves as a commercial cloud provider. Each entity is subject to independent legal and contractual obligations. More importantly, in real-world medical scenarios, the TPA is commissioned by the DO to perform audits and receives compensation for its services. This establishes a principal–agent relationship and aligns the interests between the TPA and DO. Therefore, we consider the motivation and risk of collusion between the TPA and CS to be extremely low. Meanwhile, it is worth noting that although the TPA may observe multiple auditing sessions, it can only obtain randomized aggregation results. Given this limited view, we believe it is difficult for the TPA to derive the intrinsic association between the keyword and the EMRs.

3.4. Design Goals

To improve the practicality of existing integrity auditing schemes for medical cloud, the objectives of the proposed scheme are as follows:

Correctness: If the CS has completely stored all EMRs matching the target keyword, its generated proof can successfully pass the TPA’s verification.

Soundness: If the CS fails to store all EMRs matching the target keyword, the proof, *proof*, it fabricates cannot be accepted by the TPA.

Privacy protection and data dynamic updates: (1) Privacy protection: The leakage profile to the TPA is strictly limited to the ciphertext of the target keyword, the authentication identifier and the auditing proof returned by the CS. The TPA cannot obtain the intrinsic association between the keyword and the EMRs, nor can it infer the real data from the auditing proof by a replay attack. (2) Data dynamic updates: Medical staff can perform efficient dynamic data updates (block-level and file-level) on outsourced EMRs while ensuring privacy against the TPA during data updates.

We clarify three security notions: (i) Confidentiality ensures ciphertext-only storage and transmission for keywords. (ii) Privacy preservation specifically prevents the TPA from deriving the intrinsic association between the keyword and the EMRs. (iii) Although the TPA knows the identity of the auditing initiator, the information it obtains is limited to aggregated results, as described above. Hence, anonymity is not a goal of our scheme.

4. Our Proposed Scheme

We introduce the algorithm of our scheme in detail as follows:

(1) *Setup* (1^λ):

Given a security parameter 1^λ , the algorithm outputs the public parameter $\mathbb{PP} = \{p, g, u, \mathbb{G}_1, \mathbb{G}_2, e, H_1, H_2, H_3, H_4, \pi(\cdot)\}$. DO randomly selects $x \in \mathbb{Z}_p^*$ and computes $y = g^x$, which serve as its private key and public key, respectively.

(2) *Pretreat and AuthDen* (x, MID, M):

Assume the DO has n EMRs outsourced to CS initially, and divides each EMR M_i into t blocks, where $m_{i,j}$ denotes the j -th block of EMR M_i .

- (a) For blocks containing patients’ sensitive information, DO computes $\hat{m}_{i,j} = m_{i,j} \oplus b \cdot H_4(MID_i \parallel j)$, where $b \in \mathbb{Z}_p^*$. Otherwise, DO simply sets $\hat{m}_{i,j} = m_{i,j}$. Then, DO gets the preprocessed EMR $\hat{M}_i = \{\hat{m}_{i,j}\}_{j=1}^t$. Note: The preprocessing operation is a one-time pad, and the DO only needs to store the random value b to recover the plaintext as follows: $m_{i,j} = \hat{m}_{i,j} \oplus b \cdot H_4(MID_i \parallel j)$. In addition, preprocessing is only for hiding sensitive data from the cloud. Medical staff within the department already have access to these records in plaintext.

The value b is a secret parameter stored at the department level rather than a personal key. Therefore, personnel changes do not affect data availability, and successors can directly continue using the same b .

- (b) For each block $\hat{m}_{i,j}$, DO computes $\sigma_{i,j} = \left(H_1(MID_i) \cdot H_2(\hat{m}_{i,j}) \cdot u^{\hat{m}_{i,j}} \right)^x$.
 - (c) DO sends $\{\hat{M}_i, MID_i, \Phi_i\}_{i=1}^n$ to the HIS administrator, where $\Phi_i = \{\sigma_{i,j}\}_{j=1}^t$.
- (3) **KeywordExt and IndexGen** ($\hat{M}$):
- (a) The HIS administrator extracts a keyword set $\{w_k\}_{k=1}^K$ from all preprocessed EMRs. For each keyword w_k , a corresponding inverted index set $\mathcal{I}_{w_k}$ is built to collect the indexes of all preprocessed EMRs that are matching keyword w_k ; specifically, if the EMR $\hat{M}_i$ is matching the keyword w_k , its index i is inserted into $\mathcal{I}_{w_k}$. Initially, all sets $\mathcal{I}_{w_k}$ are empty.
 - (b) For each EMR $\hat{M}_i$, the HIS administrator calculates a dynamic hash list $\mathcal{H}\mathcal{L}_i = \{H_2(\hat{m}_{i,j})\}_{j=1}^t$.
 - (c) The HIS administrator stores $\{\tau_k, \mathcal{I}_{w_k}\}_{k=1}^K$ and $\{MID_i, \mathcal{H}\mathcal{L}_i\}_{i=1}^n$. Then, the administrator sends $\{\hat{M}_i, MID_i, \Phi_i\}_{i=1}^n$ and $\{\tau_k, \mathcal{I}_{w_k}\}_{k=1}^K$ to the CS, where $\tau_k = \pi(w_k)$.
- (4) **ChalGen** (w_k):
- Suppose DO wants to check the integrity of all preprocessed EMRs matching the keyword w_k .
- (a) The HIS administrator sends an auditing request $\left\{ \tau_k, \{\theta_{w_k,j}\}_{j=1}^t \right\}$ for the keyword w_k to the TPA, where the authentication identifier $\theta_{w_k,j} = \prod_{i \in \mathcal{I}_{w_k}} (H_1(MID_i) \cdot H_2(\hat{m}_{i,j}))$. Since the authentication identifier obtained by the TPA is an aggregation of the block hashes of all EMRs matching the target keyword, and no direct mapping exists between the keyword and the EMRs, the TPA is unable to reconstruct the intrinsic association between the keyword and the EMRs.
 - (b) Upon receiving an auditing request for keyword w_k , the TPA randomly samples a c -element subset $S \subseteq [1, t]$, and for each $j \in S$ picks $v_j \in_R \mathbb{Z}_p^*$.
 - (c) The TPA generates the auditing challenge for the keyword w_k as $Chal = \{\tau_k, \{v_j, j\}_{j \in S}\}$ and sends it to the CS.
- (5) **ProofGen** ($\hat{M}, \{\Phi_i\}_{i=1}^n, \tau_k, Chal$):
- (a) After receiving $Chal$, the CS retrieves all preprocessed EMRs matching the keyword w_k from the inverted index set $\mathcal{I}_{w_k}$.
 - (b) The CS picks a random $r \in \mathbb{Z}_p^*$, calculates a corresponding auditing proof $Proof = \{R, T, \gamma\}$, and returns it to the TPA, where $R = u^r$, $T = \prod_{i \in \mathcal{I}_{w_k}} \prod_{j \in S} \sigma_{i,j}^{v_j}$, $\gamma = \sum_{i \in \mathcal{I}_{w_k}} \sum_{j \in S} \hat{m}_{i,j} \cdot v_j + r \cdot H_3(R)$.
- (6) **ProofVerify** ($Chal, Proof, \{\theta_{w_k,j}\}_{j=1}^t$):
- The *Proof*'s correctness is checked by the TPA using the equation below:

$$e(g, T) = e\left(y, \prod_{j \in S} (\theta_{w_k,j})^{v_j} \cdot u^\gamma \cdot R^{-H_3(R)}\right) \quad (1)$$

In the equation, T denotes the aggregated authenticator of all challenged blocks belonging to the EMRs that match the keyword w_k , γ is the masked linear combination of the challenged data blocks (which includes a random blinding factor), and $y = g^x$ is

the DO's public key, where x is the private key. The correctness of the proof, *Proof*, is verified by exploiting the bilinearity of the pairing e . When the equation is satisfied, the TPA outputs 1, indicating that the CS correctly stores all preprocessed EMRs matching the keyword w_k . Otherwise, the TPA returns 0, meaning that the relevant EMRs have been corrupted or lost. Detailed derivation of this equation is presented in Theorem 1 of Section 5.

(7) Data Dynamic Updates

(a) File-level updates:

(1) File insertion: Suppose DO wants to upload a new EMR M_{n+1} . DO first preprocesses the EMR as previously described, yielding $\hat{M}_{n+1} = \{\hat{m}_{n+1,j}\}_{j=1}^t$. For $j \in [1, t]$, the DO computes the authenticator $\sigma_{n+1,j}$ and sets $\Phi_{n+1} = \{\sigma_{n+1,j}\}_{j=1}^t$. Then, DO sends a file insertion request $\{MIn, \hat{M}_{n+1}, MID_{n+1}, \Phi_{n+1}\}$ to the HIS administrator. Upon receiving the request, the HIS administrator extracts keywords from the preprocessed EMR $\hat{M}_{n+1}$ and updates the corresponding inverted index set $\mathcal{I}'_{w_k}$ (adding index $n+1$ of $\hat{M}_{n+1}$ to the set). The HIS administrator computes a dynamic hash list $\mathcal{H}\mathcal{L}_{n+1} = \{H_2(\hat{m}_{n+1,j})\}_{j=1}^t$ corresponding to $\hat{M}_{n+1}$. Then, the administrator stores $\{MID_{n+1}, \mathcal{H}\mathcal{L}_{n+1}\}$ and $\{\tau_k, \mathcal{I}'_{w_k}\}_{k=1}^K$. The file insertion request MIn and $\{\tau_k, \mathcal{I}'_{w_k}\}_{k=1}^K$ are subsequently sent to the CS. Upon receiving MIn , the CS inserts $\hat{M}_{n+1}$ and its authenticator set Φ_{n+1} , and replaces $\{\tau_k, \mathcal{I}_{w_k}\}_{k=1}^K$ with $\{\tau_k, \mathcal{I}'_{w_k}\}_{k=1}^K$.

(2) File deletion: If the DO wishes to delete the EMR $\hat{M}_i$, it sends a file deletion request $\{MDe, MID_i\}$ to the HIS administrator. Upon receiving MDe , the HIS administrator updates the inverted index set $\mathcal{I}_{w_k}$ to $\mathcal{I}_{w_k}''$ by deleting the index i of $\hat{M}_i$ and removes the corresponding dynamic hash list $\mathcal{H}\mathcal{L}_i$. The administrator sends MDe and $\{\tau_k, \mathcal{I}_{w_k}''\}_{k=1}^K$ to the CS. Upon receiving MDe , the CS deletes the EMR $\hat{M}_i$, along with its MID_i and the authenticator set Φ_i , and replaces $\{\tau_k, \mathcal{I}_{w_k}\}_{k=1}^K$ with $\{\tau_k, \mathcal{I}_{w_k}''\}_{k=1}^K$.

(b) Block-level update:

(1) Block insertion: Suppose DO intends to insert $m'_{i,j}$ into the EMR M_i at position j . DO first preprocesses the block as previously described, yielding $\hat{m}'_{i,j}$. Then, DO computes the authenticator $\sigma'_{i,j}$ and sends a block insertion request $\{BIn, MID_i, j, \hat{m}'_{i,j}, \sigma'_{i,j}\}$ to the HIS administrator. After receiving BIn , the HIS administrator computes $H_2(\hat{m}'_{i,j})$, inserts it into $\mathcal{H}\mathcal{L}_i$ at position j , and sends BIn to the CS. The CS then places both $\hat{m}'_{i,j}$ and $\sigma'_{i,j}$ at position j in $\hat{M}_i$ and Φ_i , respectively.

(2) Block deletion: When DO wishes to delete the j -th block of EMR M_i , it issues a block deletion request $\{BDe, MID_i, j\}$ to the HIS administrator. After receiving BDe , the HIS administrator locates and deletes the j -th entry in the dynamic hash list $\mathcal{H}\mathcal{L}_i$, and sends BDe to the CS. The CS then deletes $\hat{m}_{i,j}$ from $\hat{M}_i$ and $\sigma_{i,j}$ from Φ_i .

(3) Block modification: When DO intends to modify $m_{i,j}$ to $m''_{i,j}$ in the EMR M_i , it first preprocesses the block, as previously described, yielding $\hat{m}''_{i,j}$. DO then computes the authenticator $\sigma''_{i,j}$ and sends a block modification request $\{BMf, MID_i, j, \hat{m}''_{i,j}, \sigma''_{i,j}\}$ to the HIS administrator. Upon receiving BMf , the administrator computes $H_2(\hat{m}''_{i,j})$ and uses it to replace the value at the j -th position of the dynamic hash list $\mathcal{H}\mathcal{L}_i$. Then, the administrator sends the block modification request BMf to the CS. The CS then replaces the j -th value in the EMR $\hat{M}_i$ and its authenticator set Φ_i with $\hat{m}''_{i,j}$ and $\sigma''_{i,j}$, respectively.

Note: We assume the CS is honest-but-curious; i.e., it will correctly perform every user request. The HIS administrator, as part of the hospital system, is also trusted to act honestly. Nevertheless, if either party fails to synchronise the updates, the verification equation will not hold.

Suppose the EMR $\hat{M}_i$ is divided into four blocks. The process of block deletion, insertion, and modification, together with their effect on the dynamic hash list $\mathcal{H}\mathcal{L}_i$, is illustrated in Figure 2. Here, $\hat{m}_{i,0}$ is a virtual block of the EMR $\hat{M}_i$, which is used as a substitute when the challenged block of the EMR $\hat{M}_i$ does not exist. Since this is not the main focus of this paper, we omit the details here.

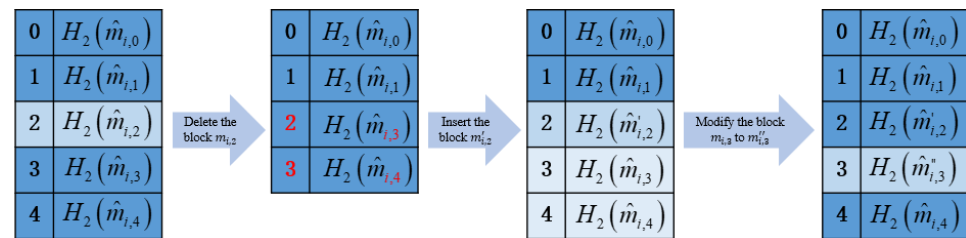


Figure 2. The dynamic hash list $\mathcal{H}\mathcal{L}_i$ for block-level updates.

5. Security Analysis

Theorem 1 (Correctness). *If the CS has completely stored all EMRs $\{\hat{M}_i\}_{i \in \mathcal{I}_{w_k}}$, its generated corresponding proof, $Proof$, can successfully pass the TPA's verification.*

Proof of Theorem 1. Equation (1) can be shown to be correct as follows:

$$\begin{aligned}
 & e(g, T) \\
 &= e\left(g, \prod_{i \in \mathcal{I}_{w_k}} \prod_{j \in S} \sigma_{i,j}^{v_j}\right) \\
 &= e\left(g, \prod_{i \in \mathcal{I}_{w_k}} \prod_{j \in S} \left(\left(H_1(MID_i) \cdot H_2(\hat{m}_{i,j}) \cdot u^{\hat{m}_{i,j}}\right)^x\right)^{v_j}\right) \\
 &= e\left(g^x, \prod_{i \in \mathcal{I}_{w_k}} \prod_{j \in S} \left(H_1(MID_i) \cdot H_2(\hat{m}_{i,j}) \cdot u^{\hat{m}_{i,j}}\right)^{v_j}\right) \\
 &= e\left(g^x, \prod_{j \in S} \left(\prod_{i \in \mathcal{I}_{w_k}} H_1(MID_i) \cdot H_2(\hat{m}_{i,j})\right)^{v_j} \cdot u^{\sum_{i \in \mathcal{I}_{w_k}} \sum_{j \in S} \hat{m}_{i,j} \cdot v_j + r \cdot H_3(R)} \cdot u^{-r \cdot H_3(R)}\right) \\
 &= e\left(y, \prod_{j \in S} \theta_{w_k,j}^{v_j} \cdot u^\gamma \cdot u^{-r \cdot H_3(R)}\right) \quad \square
 \end{aligned}$$

Theorem 2 (Soundness). *Based on the hardness of the CDH and DL problems in G_1 , any malicious CS has only a negligible probability of forging a valid auditing proof in our scheme.*

Proof of Theorem 2. We prove this via the following two security games:

Game 1: Assume that the adversary $\mathcal{A}$ can forge a valid proof ($T^* \neq T$) that passes the verification with advantage $\epsilon(\kappa)$. Then there exists a PPT adversary $\mathcal{B}$ that can solve the CDH problem with an advantage of at least $\epsilon(\kappa) - \frac{1}{p}$. Namely, given g, g^x , and ω , $\mathcal{B}$ intends to compute ω^x .

In the subsequent steps, $\mathcal{B}$ acts as the challenger for $\mathcal{A}$ and embeds the CDH problem into the interaction with $\mathcal{A}$.

Setup: $\mathcal{B}$ randomly selects $a, b \in \mathbb{Z}_p^*$, computes $u = \omega^b \cdot g^a$ and sends the public parameter $\mathbb{PP} = \{p, g, u, \mathbb{G}_1, \mathbb{G}_2, e, H_1, H_2, \pi(\cdot)\}$ and public key $y = g^x$ to $\mathcal{A}$. Here, the secret key is implicitly set as x .

Query phase:

- (1) H_1 Query: $\mathcal{B}$ maintains a list H_1^{list} , which is initially empty, consisting of tuples of the form (MID_i, b_i, y_i) . When $\mathcal{A}$ queries MID_i , $\mathcal{B}$ checks H_1^{list} . If a matching entry exists, $\mathcal{B}$ returns the corresponding y_i ; otherwise, $\mathcal{B}$ randomly selects $b_i \in \mathbb{Z}_p^*$ and computes $y_i = g^{b_i}$. Then, $\mathcal{B}$ inserts (MID_i, b_i, y_i) into H_1^{list} and returns y_i .
- (2) H_2 Query: $\mathcal{B}$ maintains a list H_2^{list} , which is initially empty, consisting of tuples of the form $(\hat{m}_{i,j}, r_{i,j}, \hat{y}_{i,j})$. When $\mathcal{A}$ queries $\hat{m}_{i,j}$, $\mathcal{B}$ checks H_2^{list} . If a matching entry exists, $\mathcal{B}$ returns the corresponding $\hat{y}_{i,j}$; otherwise, $\mathcal{B}$ randomly selects $r_{i,j} \in \mathbb{Z}_p^*$ and computes $\hat{y}_{i,j} = g^{r_{i,j}} \cdot \omega^{-b \cdot \hat{m}_{i,j}}$. Subsequently, $\mathcal{B}$ inserts $(\hat{m}_{i,j}, r_{i,j}, \hat{y}_{i,j})$ into H_2^{list} and returns $\hat{y}_{i,j}$. That is, $\mathcal{B}$ embeds the CDH problem into H_2^{list} .
- (3) Authenticator query: $\mathcal{B}$ maintains a list T^{list} , which is initially empty, consisting of tuples of the form $(\hat{m}_{i,j}, b_i, y_i, r_{i,j}, \hat{y}_{i,j}, \sigma_{i,j})$. When adversary $\mathcal{A}$ queries $\hat{m}_{i,j}$, $\mathcal{B}$ checks T^{list} . If a matching entry exists, $\mathcal{B}$ returns the corresponding $\sigma_{i,j}$; otherwise, $\mathcal{B}$ computes $\sigma_{i,j} = (g^x)^{b_i + r_{i,j} + a \cdot \hat{m}_{i,j}}$. Subsequently, $\mathcal{B}$ inserts $(\hat{m}_{i,j}, b_i, y_i, r_{i,j}, \hat{y}_{i,j}, \sigma_{i,j})$ into T^{list} and returns $\sigma_{i,j}$. This is possible because

$$\begin{aligned} \sigma_{i,j} &= \left(y_i \cdot \hat{y}_{i,j} \cdot \left(\omega^b \cdot g^a \right)^{\hat{m}_{i,j}} \right)^x \\ &= g^{b_i \cdot x} \cdot g^{r_{i,j} \cdot x} \cdot \omega^{-b \cdot \hat{m}_{i,j} \cdot x} \cdot \omega^{b \cdot \hat{m}_{i,j} \cdot x} \cdot g^{a \cdot \hat{m}_{i,j} \cdot x} \\ &= (g^x)^{b_i + r_{i,j} + a \cdot \hat{m}_{i,j}} \end{aligned}$$

Challenge Phase: In this step, a challenge $Chal = \{\tau_k, \{v_j, j\}_{j \in S}\}$ is generated by $\mathcal{B}$ and sent to $\mathcal{A}$. There exists at least one challenged block $\hat{m}_{i,j}$ whose authenticator has never been queried. Subsequently, $\mathcal{B}$ asks that $\mathcal{A}$ output a proof *Proof*.

Forgery phase: Suppose $\mathcal{A}$ forges a valid proof $Proof^* = \{R, T^*, \gamma^*\}$ that passes verification with advantage $\varepsilon(\kappa)$. Here, $\gamma^* \neq \gamma$; otherwise, $T^* = T$. $\mathcal{B}$ then obtains:

$$e(g, T^*) = e\left(y, \prod_{j \in S} \theta_{w_{k,j}}^{v_j} \cdot u^{\gamma^*} \cdot R^{-H_3(R)}\right)$$

In addition, $\mathcal{B}$ possesses a valid proof $Proof = \{R, T, \gamma\}$ and can derive:

$$e(g, T) = e\left(y, \prod_{j \in S} \theta_{w_{k,j}}^{v_j} \cdot u^{\gamma} \cdot R^{-H_3(R)}\right)$$

From the two equalities above, $\mathcal{B}$ deduces:

$$\begin{aligned} &e(g, T^* \cdot T^{-1}) \\ &= e\left(y, \prod_{j \in S} \theta_{w_{k,j}}^{v_j} \cdot u^{\gamma^*} \cdot R^{-H_3(R)}\right) \cdot e\left(y, \prod_{j \in S} \theta_{w_{k,j}}^{v_j} \cdot u^{\gamma} \cdot R^{-H_3(R)}\right)^{-1} \\ &= e(g^x, u^{\gamma^* - \gamma}) \\ &= e\left(g, \left(\omega^b \cdot g^a\right)^{(\gamma^* - \gamma) \cdot x}\right) \\ &= e\left(g, (\omega^x)^{b \cdot (\gamma^* - \gamma)} \cdot (g^x)^{a \cdot (\gamma^* - \gamma)}\right) \end{aligned}$$

Since $T^* \cdot T^{-1} = (\omega^x)^{b \cdot (\gamma^* - \gamma)} \cdot (g^x)^{a \cdot (\gamma^* - \gamma)}$, $\mathcal{B}$ can obtain:

$$\omega^x = \left((g^x)^{-a \cdot (\gamma^* - \gamma)} \cdot T^* \cdot T^{-1} \right)^{-b \cdot (\gamma^* - \gamma)}$$

The probability that this result holds is at least $\varepsilon(\kappa) - \frac{1}{p}$ (p is large prime), because $a \cdot (\gamma^* - \gamma) \not\equiv 0 \pmod{p}$. Hence, we demonstrate that $\mathcal{B}$ can compute ω^x with an advantage of at least $\varepsilon(\kappa) - \frac{1}{p}$, which contradicts the CDH assumption. By the hardness of the CDH problem, we can conclude that $T^* = T$.

Game 2: Assume that the adversary $\mathcal{A}$ can forge a valid proof ($\gamma^* \neq \gamma$) that passes the verification with advantage $\varepsilon(\kappa)$. Then there exists a PPT adversary $\mathcal{B}$ that can solve the DL problem with an advantage of at least $\varepsilon(\kappa) - \frac{1}{p}$. Namely, given g and g^x , $\mathcal{B}$ intends to compute x .

In the subsequent steps, $\mathcal{B}$ acts as the challenger for $\mathcal{A}$ and embeds the DL problem into the interaction with $\mathcal{A}$.

Setup: $\mathcal{B}$ randomly selects $a, b \in \mathbb{Z}_p^*$, computes $u = (g^x)^b \cdot g^a$ and sends the public parameter $\mathbb{PP} = \{p, g, u, \mathbb{G}_1, \mathbb{G}_2, e, H_1, H_2, \pi(\cdot)\}$ and public key $y = g^x$ to $\mathcal{A}$. Here, the secret key is implicitly set as x .

Query phase:

- (1) H_1 Query: The same as *Game 1*.
- (2) H_2 Query: $\mathcal{B}$ maintains a list H_2^{list} , which is initially empty, consisting of tuples of the form $(\hat{m}_{i,j}, r_{i,j}, \hat{y}_{i,j})$. When $\mathcal{A}$ queries $\hat{m}_{i,j}$, $\mathcal{B}$ checks H_2^{list} . If a matching entry exists, $\mathcal{B}$ returns the corresponding $\hat{y}_{i,j}$; otherwise, $\mathcal{B}$ randomly selects $r_{i,j} \in \mathbb{Z}_p^*$ and computes $\hat{y}_{i,j} = g^{r_{i,j}} \cdot (g^x)^{-b \cdot \hat{m}_{i,j}}$. Subsequently, $\mathcal{B}$ inserts $(\hat{m}_{i,j}, r_{i,j}, \hat{y}_{i,j})$ into H_2^{list} and returns $\hat{y}_{i,j}$. That is, $\mathcal{B}$ embeds the DL problem into H_2^{list} .
- (3) Authenticator query: $\mathcal{B}$ maintains a list T^{list} , which is initially empty, consisting of tuples of the form $(\hat{m}_{i,j}, b_i, y_i, r_{i,j}, \hat{y}_{i,j}, \sigma_{i,j})$. When adversary $\mathcal{A}$ queries $\hat{m}_{i,j}$, $\mathcal{B}$ checks T^{list} . If a matching entry exists, $\mathcal{B}$ returns the corresponding $\sigma_{i,j}$; otherwise, $\mathcal{B}$ computes $\sigma_{i,j} = (g^x)^{b_i + r_{i,j} + a \cdot \hat{m}_{i,j}}$. Subsequently, $\mathcal{B}$ inserts $(\hat{m}_{i,j}, b_i, y_i, r_{i,j}, \hat{y}_{i,j}, \sigma_{i,j})$ into T^{list} and returns $\sigma_{i,j}$. This is possible because

$$\begin{aligned}\sigma_{i,j} &= \left(y_i \cdot \hat{y}_{i,j} \cdot \left((g^x)^b \cdot g^a \right)^{\hat{m}_{i,j}} \right)^x \\ &= \left(g^{b_i} \cdot g^{r_{i,j}} \cdot (g^x)^{-b \cdot \hat{m}_{i,j}} \cdot (g^x)^{b \cdot \hat{m}_{i,j}} \cdot g^{a \cdot \hat{m}_{i,j}} \right)^x \\ &= (g^x)^{b_i + r_{i,j} + a \cdot \hat{m}_{i,j}}\end{aligned}$$

Challenge Phase: The same as *Game 1*.

Forgery phase: Suppose $\mathcal{A}$ forges a valid proof $Proof^* = \{R, T^*, \gamma^*\}$ that passes verification with advantage $\varepsilon(\kappa)$. As shown in the analysis of *Game 1*, $T = T^*$. $\mathcal{B}$ then obtains:

$$e(g, T) = e\left(y, \prod_{j \in S} \theta_{w_{k,j}}^{v_j} \cdot u^{\gamma^*} \cdot R^{-H_3(R)}\right)$$

In addition, $\mathcal{B}$ possesses a valid proof $Proof = \{R, T, \gamma\}$ and can derive:

$$e(g, T) = e\left(y, \prod_{j \in S} \theta_{w_{k,j}}^{v_j} \cdot u^{\gamma} \cdot R^{-H_3(R)}\right)$$

From the two equalities above, $\mathcal{B}$ deduces $u^{\gamma^*} = u^{\gamma}$. Since $\frac{u^{\gamma^*}}{u^{\gamma}} = \left((g^x)^b \cdot g^a\right)^{\gamma^* - \gamma} = 1$, $\mathcal{B}$ can obtain $g^x = g^{-\frac{a}{b}}$; then $x = -\frac{a}{b}$. The probability that this result holds is at least $\varepsilon(\kappa) - \frac{1}{p}$ (p is large prime) because $b \not\equiv 0 \pmod{p}$. Hence, we demonstrate that $\mathcal{B}$ can compute x with an advantage of at least $\varepsilon(\kappa) - \frac{1}{p}$, which contradicts the DL assumption. By the hardness of the DL problem, we can conclude that $\gamma^* = \gamma$.

Therefore, if the CS fails to store all EMRs matching the target keyword, the proof, *proof*, it fabricates cannot be accepted by the TPA. $\square$

Theorem 3 (Privacy protection). *Our scheme conceals the intrinsic association between keywords and the EMRs from the TPA and prevents replay attacks.*

Proof of Theorem 3. During the auditing process, the TPA only obtains the search token τ_k , the authentication identifier set $\{\theta_{w_{k,j}}\}_{j=1}^t$, and the auditing proof $Proof = \{R, T, \gamma\}$ returned by the CS. $\{\theta_{w_{k,j}}\}_{j=1}^t$ is the aggregated product of the block hashes of all blocks belonging to the EMRs matching the target keyword. Similarly, T in the auditing proof is the aggregated product of the authenticators of these blocks, and γ is the blinded linear combination of the real content of these blocks. Since all these values are in aggregated form and contain no individual EMR identifiers or index information, the TPA cannot infer the intrinsic association between the keyword and the EMRs from such aggregated values.

Replay attacks: the curious TPA may repeatedly launch multiple auditing challenges for the same keyword in an attempt to infer the real data [35]. After receiving the auditing challenge $Chal = \{\tau_k, \{v_j, j\}_{j \in S}\}$, the CS picks a random $r \in \mathbb{Z}_p^*$, calculates a corresponding auditing proof $Proof = \{R, T, \gamma\}$, and returns it to the TPA, where $R = u^r$, $T = \prod_{i \in \mathcal{I}_{w_k}} \prod_{j \in S} \sigma_{i,j}^{v_j}$, $\gamma = \sum_{i \in \mathcal{I}_{w_k}} \sum_{j \in S} \hat{m}_{i,j} \cdot v_j + r \cdot H_3(R)$. γ is generated by aggregating the challenged data blocks and blinding the result with a random value r . Here $r \in \mathbb{Z}_p^*$ is chosen randomly and kept secret from TPA. Thus, due to the randomness of r , the proof, $Proof$, changes even when the same challenge $Chal$ is reused. A malicious TPA can hardly infer the real data from multiple auditing proofs corresponding to the identical auditing challenge. $\square$

Defense-in-Depth Discussion: We have argued in the threat model that the motivation and risk of collusion between the CS and TPA are extremely low. However, during the auditing process, although the TPA is assumed to honestly execute the auditing protocol, it may still be curious about DO's data. Once the TPA and CS actually collude, they can interact through multiple audit challenge response sessions, and the TPA can solve a system of equations using the auditing proofs returned by the CS, potentially recovering the data blocks stored by the DO in the cloud, thereby causing data leakage [11].

In our scheme, the DO preprocesses the original EMRs before uploading, and only the preprocessed ciphertext blocks $\hat{m}_{i,j} = m_{i,j} \oplus b \cdot H_4(MID_i \parallel j)$ are stored on the CS, where $b \in \mathbb{Z}_p^*$. Consequently, even if the TPA and CS collude and successfully recover $\hat{m}_{i,j}$ from auditing proofs, they cannot derive the original plaintext $m_{i,j}$ without the secret value b used in the preprocessing. This ensures that sensitive medical information remains protected even in the event of collusion.

6. Performance Analysis

This section presents a comparison between our scheme and several existing keyword-based auditing schemes. The comparison covers four aspects: functionality, communication overhead, computation overhead, and storage overhead.

Function: Table 2 indicates that our scheme concurrently supports data integrity verification, privacy protection, replay attack resistance, and both file-level and block-level updates. Thus, from a functional perspective, our scheme is particularly suitable for practical application scenarios.

Table 2. Function comparison.

Scheme	Integrity	Privacy Protection	Replay Attack	File-Level Updates	Block-Level Updates
Xu et al. [7]	✓	×	×	✓	×
Gao et al. [8]	✓	✓	×	✓	×
Shen et al. [9]	✓	×	×	✓	✓
Our scheme	✓	✓	✓	✓	✓

Communication overhead: Unlike the schemes in [7–9], our scheme introduces additional communication overhead during the data integrity auditing phase due to the authentication identifier set $\{\theta_{w_k,j}\}_{j=1}^t$. Specifically, in the challenge generation phase, the HIS administrator sends this set to the TPA as part of the auditing request. The total communication overhead introduced is $t \cdot |\mathbb{G}_1|$ bits, where t is the number of blocks in each EMR and $|\mathbb{G}_1|$ is the bit length of an element in $\mathbb{G}_1$. With $t = 1000$ and $|\mathbb{G}_1| = 512$ bits, this overhead amounts to approximately 512 Kb (64 KB) per auditing request. Since the

auditing request is transmitted only once per audit and its size does not scale with the number of EMRs matching the keyword w_k , this overhead is acceptable in practice.

Computation overhead: We detail the experimental hardware and software setup in Table 3. Let E denote an exponentiation operation, H denote a hash operation, M denote a multiplication operation, and P denote a pairing operation. As tabulated in Table 4, we implement 1000 independent tests for each kind of cryptographic operation, with the averaged computation time taken as the final experimental datum.

Table 3. Experimental setup.

Item	Version
CPU	AMD Ryzen 5 4500U with Radeon Graphics
Operating system	Windows 11
RAM	16.0 GB
Language	IntelliJ IDEA 2025
Cryptographic library	JPBC2.0 library

Table 4. Average runtime for fundamental cryptographic operations.

Operation Type	Average Runtime (ms)
Exponentiation operation E	7.7009
Hash operation H	18.0998
Multiplication operation M	0.0328
Pairing operation P	5.2651

We compare the computational overhead of our scheme with those of the schemes in [7–9] during the auditing process and block-level updates. Suppose the DO uploads n EMRs, each divided into t data blocks. During the auditing phase, each EMR has c of its blocks challenged. A keyword relates to W EMRs. When a block-level update occurs, let l denote the number of blocks being modified, inserted, or deleted, and let a represent the number of blocks following the updated block. Table 5 presents a comparison of theoretical computational overheads among the schemes in the auditing phase: authenticator generation, proof generation, and proof verification. The theoretical computational overheads of the schemes in the three block-level update operations (block modification, block insertion, and block deletion) are compared in Table 6.

To ensure a fair and objective comparison, we re-implemented the schemes of [7–9] based on their original descriptions and conducted all experiments under identical hardware and software environments to guarantee consistency.

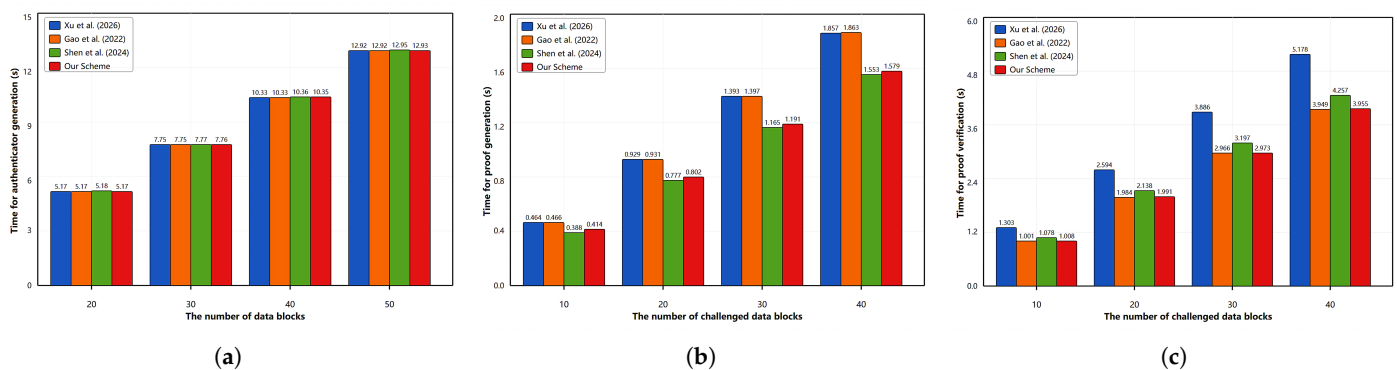
Table 5. Computational efficiency 1.

Scheme	Authenticator Generation	Proof Generation	Proof Verification
Xu et al. [7]	$n \cdot t \cdot (H + M + E)$	$W \cdot c \cdot (M + E) + c \cdot (2M + E)$	$W \cdot c \cdot H + (W \cdot c + 2 \cdot W) \cdot M + W \cdot c \cdot E + 2P$
Gao et al. [8]	$n \cdot t \cdot (H + M + E)$	$W \cdot c \cdot (2M + E) + c \cdot (M + E)$	$W \cdot c \cdot H + 2c \cdot M + (c + 1)E + 2P$
Shen et al. [9]	$n \cdot t \cdot (H + 3M + E)$	$W \cdot c \cdot (2M + E)$	$(2c + W - 1) \cdot M + W \cdot c \cdot H + 2P + (2c + 1) \cdot E$
Our scheme	$n \cdot t \cdot (H + 2M + E)$	$W \cdot c \cdot (2M + E) + H + M + E$	$H + (c + 2) \cdot M + (c + 2) \cdot E + 2P$

Table 6. Computational efficiency 2.

Scheme	Block Modification	Block Insertion	Block Deletion
Xu et al. [7]	$l \cdot (H + M + E)$	$l \cdot a \cdot (H + M + E)$	$l \cdot a \cdot (H + M + E)$
Gao et al. [8]	$l \cdot (H + M + E)$	$l \cdot a \cdot (H + M + E)$	$l \cdot a \cdot (H + M + E)$
Shen et al. [9]	$l \cdot (H + 3M + E)$	$l \cdot (H + 3M + E)$	—
Our scheme	$l \cdot (H + 2M + E)$	$l \cdot (3H + 3M + E)$	—

As shown in Figure 3, as the number of data blocks and challenged blocks increases, the computational overhead of all schemes in the three phases—authenticator generation, proof generation, and proof verification—exhibits a linear growth trend. In the authenticator generation phase, the overheads of all schemes are roughly comparable. Taking the case of 50 data blocks as an example, our scheme incurs 12.93 s, which is on par with Xu et al.’s [7] scheme (12.92 s) and Gao et al.’s [8] scheme (12.92 s). In the proof generation phase, the advantage of our scheme becomes more pronounced. As the number of challenged blocks increases from 10 to 40, the time cost of our scheme grows linearly from 0.414 s to 1.579 s, consistently lower than that of Xu et al. [7] (from 0.464 s to 1.857 s) and Gao et al. [8] (from 0.466 s to 1.863 s), though slightly higher than that of Shen et al. [9] (from 0.388 s to 1.553 s). In the proof verification phase, our scheme also demonstrates favourable efficiency, with the time cost increasing from 1.008 s to 3.955 s, which is lower than those of Xu et al. [7] (from 1.303 s to 5.178 s) and Shen et al. [9] (from 1.078 s to 4.257 s), and comparable to that of Gao et al. [8] (from 1.001 s to 3.949 s). Overall, during the entire auditing process, our scheme achieves computational efficiency that is comparable to or even better than existing schemes.

**Figure 3.** Auditing process. (a) Authenticator generation; (b) Proof generation; (c) Proof verification [7–9].

We further evaluate the efficiency of block-level updates, including block modification, insertion, and deletion.

For block modification, as illustrated in Figure 4, all four schemes exhibit nearly identical performance. When the number of modified blocks increases from 10 to 40, the time cost of our scheme ranges from 0.259 s to 1.035 s, which is comparable to that of Xu et al. [7] (0.258 s to 1.033 s), Gao et al. [8] (0.258 s to 1.033 s), and Shen et al. [9] (0.259 s to 1.036 s). This indicates that our scheme achieves modification efficiency on par with existing schemes.

For block insertion, as shown in Figure 5, our scheme and Shen et al.’s [9] scheme significantly outperform those of Xu et al. [7] and Gao et al. [8]. Specifically, when the number of inserted blocks increases from 10 to 40, the time costs reported by Xu et al. [7] and Gao et al. [8] grow linearly from 1.292 s to 5.167 s, whereas our scheme and that of Shen et al. [9] grow linearly from 0.259 s to 1.035 s and from 0.259 s to 1.036 s, respectively.

Although all schemes exhibit linear growth, the growth rate of our scheme and that of Shen et al. [9] is substantially smaller than those of Xu et al. [7] and Gao et al. [8]. This is because our scheme and that of Shen et al. [9] avoid recomputing authenticators for subsequent blocks after insertion, thereby greatly reducing the computational burden.

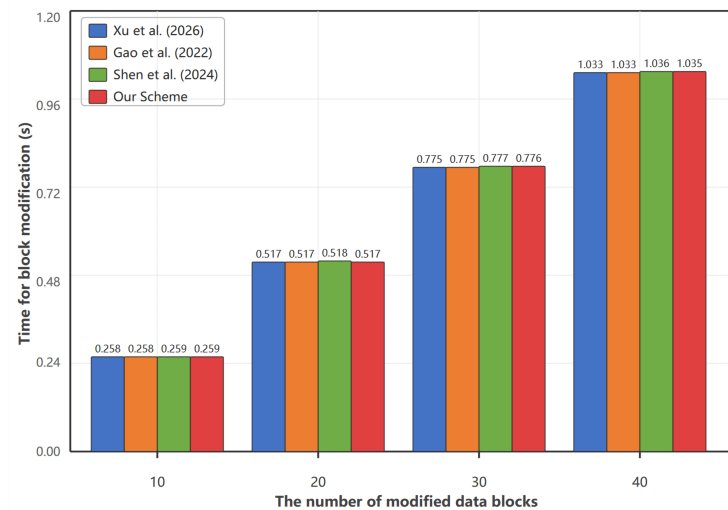


Figure 4. Block modification [7–9].

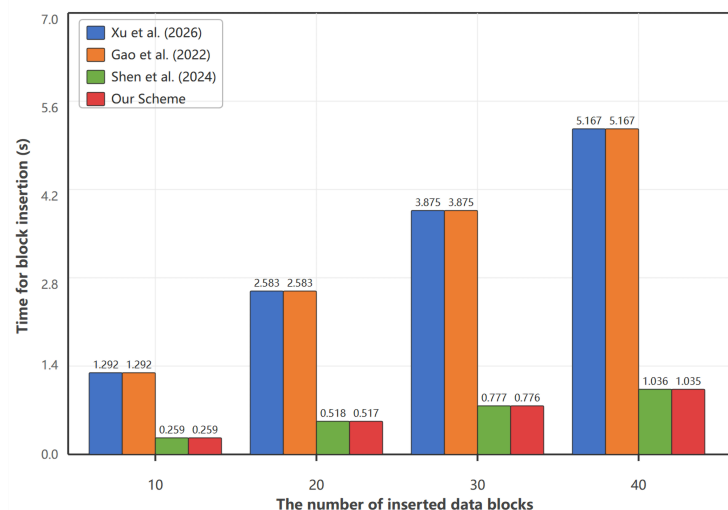


Figure 5. Block insertion [7–9].

For block deletion, as depicted in Figure 6, our scheme achieves efficiency comparable to that of Shen et al. [9], while both significantly outperform those of Xu et al. [7] and Gao et al. [8]. The time cost of our scheme remains nearly constant at approximately 0.050 s across different numbers of deleted blocks, which is comparable to that of Shen et al. [9] and substantially lower than those of Xu et al. [7] (from 0.258 s to 0.181 s) and Gao et al. [8] (from 0.258 s to 0.181 s). It is worth noting that the deletion time reported by Xu et al. [7] and Gao et al. [8] decreases slightly as the number of deleted blocks increases, which may be attributed to the reduced metadata management overhead when fewer blocks remain.

Overall, the experimental results demonstrate that our scheme achieves computational efficiency comparable to or slightly better than existing schemes in the auditing process, and significantly outperforms them in block insertion and deletion. This confirms the efficiency of our dynamic hash list mechanism. Therefore, our scheme not only attains the efficient block-level updates reported by Shen et al.'s scheme, but also additionally

preserves the intrinsic association between keywords and EMRs from the TPA, making it more suitable for privacy-sensitive medical cloud storage scenarios.

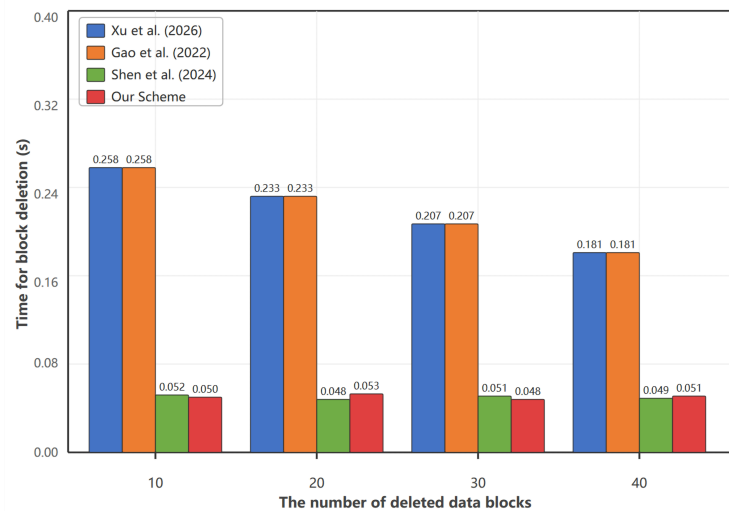


Figure 6. Block deletion [7–9].

Storage overhead: In our scheme, the HIS administrator maintains a dynamic hash list $\{\mathcal{H}\mathcal{L}_i\}_{i=1}^n$ for each EMR $\hat{M}_i$ to support efficient block-level updates. The storage overhead introduced by these dynamic hash lists is $n \cdot t \cdot |\mathbb{G}_1|$ bits, where n is the number of EMRs, t is the number of blocks per EMR, and $|\mathbb{G}_1|$ is the bit length of an element in $\mathbb{G}_1$. For example, when $n = 100$, $t = 1000$, and $|\mathbb{G}_1| = 512$ bits, the total storage overhead is approximately 51.2 Mb (6.4 MB). Although this introduces additional storage on the HIS administrator side, it enables efficient block-level updates by avoiding the recomputation of authenticators for subsequent blocks. Compared to the significant computational savings in dynamic updates, this storage overhead is acceptable in practice, especially considering that modern hospital storage systems can easily accommodate such metadata.

Note: The above experiments were conducted on a representative hardware platform. We acknowledge that real-world hospital environments involve diverse hardware and network conditions. However, we argue that our scheme is practical for medical cloud scenarios because its core cryptographic operations are platform-independent. Moreover, the dynamic hash list mechanism significantly reduces computational and communication overheads, which is particularly beneficial for resource-constrained and bandwidth-limited hospital settings. A comprehensive evaluation under heterogeneous hardware and network conditions is left as future work.

7. Conclusions

Remote data integrity auditing realizes integrity checking of cloud data without local data downloading, and it is indispensable for cloud computing security. This paper constructs a keyword-based cloud medical storage auditing scheme that supports privacy protection and dynamic data updates. In our scheme, a TPA performs keyword-based integrity verification without obtaining any sensitive information of the EMRs. Furthermore, the scheme enables efficient block-level updates. Security analysis proves the security and reliability of our scheme, and the experimental results also indicate that its efficiency achieves a practical and feasible level.

Nevertheless, we acknowledge several limitations of the current scheme: it relies on the assumption that the HIS administrator is trusted, and it does not consider collusion attacks between the CS and TPA. To address these issues, a promising direction is to

introduce blockchain technology, which we leave as an important focus of our future work. Meanwhile, we plan to extend our scheme to multi-user scenarios where multiple hospitals share the same cloud storage. In such settings, the computational overhead of the verification process maintains constant costs even as the number of hospitals increases. Furthermore, we will consider formal privacy guarantees and comprehensive evaluations in large-scale deployment environments to further validate the practicality and scalability of our scheme.

Author Contributions: Conceptualization, M.H.; methodology, M.H.; software, S.C.; validation, M.H. and S.C.; formal analysis, M.H.; investigation, M.H.; resources, M.H.; data curation, X.Z.; writing—original draft preparation, M.H.; writing—review and editing, S.C., B.Y. and X.Z.; visualization, S.C. and X.Z.; supervision, B.Y.; project administration, M.H. and B.Y.; funding acquisition, M.H. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by the Innovation Capability Support Program of Shaanxi (grant no. 2025QCY-KXJ-168) and the Innovative Research Project for Graduate Students of Baoji University of Arts and Sciences (grant no. YJSCX26YB33).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: All original results presented in this study are fully contained in the manuscript. Further questions may be addressed to the corresponding author.

Acknowledgments: The authors are grateful to the School of Mathematics and Information Science, Baoji University of Arts and Sciences, for offering access to its experimental platform and computational facilities.

Conflicts of Interest: Bo Yang is employed at Koal Software Co., Ltd. The authors declared no conflict of interest.

References

1. Konda, B.; Kasula, V.K.; Yenugula, M.; Yadulla, A.R.; Addula, S.R. Homomorphic encryption and federated attribute-based multi-factor access control for secure cloud services in integrated space-ground information networks. *Int. J. Commun. Inf. Technol.* **2022**, *3*, 33–40. [[CrossRef](#)]
2. Liu, Z.P.; Ren, L.L.; Li, R.L.; Liu, Q.N.; Zhao, Y.G. ID-based sanitizable signature data integrity auditing scheme with privacy-preserving. *Comput. Secur.* **2022**, *121*, 102858. [[CrossRef](#)]
3. Hu, X.R.; Chang, J.Y.; Ahmad, T.; Zhang, F.N.; Zhang, Y.C. Identity-Based Integrity Auditing Scheme With Sensitive Information Hiding for Proxy-Server-Assisted Cloud Storage Applications. *IEEE Internet Things J.* **2025**, *12*, 6673–6684. [[CrossRef](#)]
4. Eliazar, A.; Brown, J.T.; Cinamon, S.; Kantarcioglu, M.; Malin, B. Re-identification risk for common privacy preserving patient matching strategies when shared with de-identified demographics. *J. Am. Med. Inform. Assoc.* **2026**, *33*, 336–346. [[PubMed](#)]
5. Peng, Y.K.; Ming, Y.; Liu, H.; Wang, C.H. Storage-Efficient Certificateless Public Auditing Scheme With Data Dynamics in the Cloud. *IEEE Internet Things J.* **2026**, *13*, 21856–21870. [[CrossRef](#)]
6. Xue, J.; Luo, S.; Deng, Q.; Shi, L.; Zhang, X.; Wang, H. KA: Keyword-based auditing with frequency hiding and retrieval reliability for smart government. *J. Syst. Archit.* **2023**, *138*, 102856. [[CrossRef](#)]
7. Xu, J.; Luo, S.; Deng, Q.; Shi, L.; Zhang, X.; Liang, H. DKCIA-B: A dynamic keyword-based cloud data integrity auditing framework with backtracking support. *J. Netw. Comput. Appl.* **2026**, *252*, 104524.
8. Gao, X.; Yu, J.; Chang, Y.; Wang, H.Q.; Fan, J.X. Checking only when it is necessary: Enabling integrity auditing based on the keyword with sensitive information privacy for encrypted cloud data. *IEEE Trans. Dependable Secur. Comput.* **2022**, *19*, 3774–3789. [[CrossRef](#)]
9. Shen, W.T.; Gai, C.; Yu, J.; Su, Y. Keyword-based remote data integrity auditing supporting full data dynamics. *IEEE Trans. Serv. Comput.* **2024**, *17*, 2516–2529. [[CrossRef](#)]
10. Ateniese, G.; Burns, R.; Curtmola, R.; Herring, J.; Kissner, L.; Peterson, Z.; Song, D. Provable Data Possession at Untrusted Stores. In Proceedings of the 14th ACM Conference on Computer and Communications Security, Alexandria, VA, USA, 28 October 2007; pp. 598–609.

11. Wang, Q.; Wang, C.; Li, J.; Ren, K.; Lou, W. Enabling Public Verifiability and Data Dynamics for Storage Security in Cloud Computing. In Proceedings of the European Symposium on Research in Computer Security, Saint-Malo, France, 21 September 2009; pp. 355–370.
12. Worku, S.G.; Xu, C.; Zhao, J. Cloud data auditing with designated verifier. *Front. Comput. Sci.* **2014**, *8*, 503–512. [[CrossRef](#)]
13. Shao, B.; Zhang, L.; Bian, G. Incentive Public Auditing Scheme with Identity-Based Designated Verifier in Cloud. *Electronics* **2023**, *12*, 1308. [[CrossRef](#)]
14. Zhao, M.; Chen, H. Identity-Based Provable Data Possession with Designated Verifier from Lattices for Cloud Computing. *Entropy* **2025**, *27*, 753. [[CrossRef](#)] [[PubMed](#)]
15. Yu, J.; Ren, K.; Wang, C.; Varadharajan, V. Enabling cloud storage auditing with key-exposure resistance. *IEEE Trans. Inf. Forensics Secur.* **2015**, *10*, 1167–1179. [[CrossRef](#)]
16. Yu, J.; Wang, H.Q. Strong Key-Exposure Resilient Auditing for Secure Cloud Storage. *IEEE Trans. Inf. Forensics Secur.* **2017**, *12*, 1931–1940. [[CrossRef](#)]
17. Yuan, H.; Chen, X.; Wang, J. Blockchain-based Public Auditing and Secure Deduplication with Fair Arbitration. *Inf. Sci.* **2020**, *541*, 409–425. [[CrossRef](#)]
18. Yu, J.; Shen, W.; Zhang, X. Cloud storage auditing and data sharing with data deduplication and private information protection for cloud-based EMR. *Comput. Secur.* **2024**, *144*, 103932. [[CrossRef](#)]
19. Xie, M.; Zhao, Q.; Hong, H. A Blockchain-Based Proxy Oriented Cloud Storage Public Audit Scheme for Low-Performance Terminal Devices. In Proceedings of the International Conference on Algorithms and Architectures for Parallel Processing, Online, 3 December 2021; pp. 676–692.
20. Zhang, H.; Zhang, X.; Zhang, Y. Blockchain-Based Proxy-Oriented Data Integrity Checking Mechanism in Cloud-Assisted Intelligent Transportation Systems. *IEEE Trans. Intell. Transp. Syst.* **2024**, *25*, 17366–17381. [[CrossRef](#)]
21. Ling, K.; Gao, J.; Chang, J.; Meng, R. Assistant-Based Integrity Auditing Scheme With Privacy Protection Function for Cloud Storage. *IEEE Internet Things J.* **2025**, *13*, 882–894. [[CrossRef](#)]
22. Wang, C.; Wang, Q.; Ren, K.; Lou, W. Privacy-Preserving Public Auditing for Data Storage Security in Cloud Computing. In Proceedings of the 2010 Proceedings IEEE INFOCOM, San Diego, CA, USA 14 March 2010; pp. 1–9.
23. Yang, Y.; Chen, Y.; Chen, F.; Chen, J. Identity-based cloud storage auditing for data sharing with access control of sensitive information. *IEEE Internet Things J.* **2021**, *9*, 10434–10445. [[CrossRef](#)]
24. Guan, X.; Chang, J.; Zhang, W. Secure data sharing scheme with privacy-preserving and certificateless integrity auditing in cloud storage. *Comput. Commun.* **2024**, *224*, 285–301. [[CrossRef](#)]
25. Zhang, Q.; Qian, S.; Cui, J.; Zhong, H.; Wang, F. Blockchain-based privacy-preserving deduplication and integrity auditing in cloud storage. *IEEE Trans. Comput.* **2025**, *74*, 1717–1729. [[CrossRef](#)]
26. Wu, J.; Bian, Z.; Gao, H.; Wang, Y. A Blockchain-Based Secure Data Transaction and Privacy Preservation Scheme in IoT System. *Sensors* **2025**, *25*, 4854. [[CrossRef](#)] [[PubMed](#)]
27. Erway, C.C.; Küpçü, A.; Papamanthou, C.; Tamassia, R. Dynamic provable data possession. *ACM Trans. Inf. Syst. Secur. (TISSEC)* **2015**, *17*, 1–29. [[CrossRef](#)]
28. Liu, Z.; Wang, S.; Duan, S.; Ren, L.; Wei, J. Dynamic data integrity auditing based on hierarchical merkle hash tree in cloud storage. *Electronics* **2023**, *12*, 717. [[CrossRef](#)]
29. Wang, C.; Sun, Y.; Liu, B.; Xue, L.; Guan, X. Blockchain-based dynamic cloud data integrity auditing via non-leaf node sampling of rank-based Merkle hash tree. *IEEE Trans. Netw. Sci. Eng.* **2024**, *11*, 3931–3942. [[CrossRef](#)]
30. Zheng, X.; Shen, W.; Su, Y. DIADD: Secure deduplication and efficient data integrity auditing with data dynamics for cloud storage. *IEEE Trans. Netw. Serv. Manag.* **2025**, *22*, 299–316. [[CrossRef](#)]
31. Wang, X.; Zhang, X.; Miao, Y.; Xue, J. Enabling anonymous authorized auditing over keyword-based searchable ciphertexts in cloud storage systems. *IEEE Trans. Serv. Comput.* **2023**, *16*, 4220–4232. [[CrossRef](#)]
32. Yu, H.; Ma, Q.; Zhu, Y.; Cui, Y. Multi-keyword Searchable Data Auditing for Cloud-Based Machine Learning. In Proceedings of the International Conference on Machine Learning for Cyber Security, Hangzhou, China, 27–29 December 2024; pp. 118–132.
33. Miao, K.; Gai, J.; Yu, L. Collaborative and Searchable Integrity Auditing for Multi-Copy Data in Decentralized Storage. *IEEE Trans. Dependable Secur. Comput.* **2025**, *22*, 7585–7599. [[CrossRef](#)]
34. Sengupta, B.; Ruj, S. Keyword-Based Delegable Proofs of Storage. In Proceedings of the International Conference on Information Security Practice and Experience, Tokyo, Japan, 6 September 2018; pp. 282–298.
35. Zhang, Y.; Wang, X.A.; Jiang, W. An Efficient and Secure Data Audit Scheme for Cloud-Based EHRs with Recoverable and Batch Auditing. *Comput. Mater. Contin.* **2025**, *83*, 1533–1553. [[CrossRef](#)]

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.