# MediChainAudit: Encrypted Medical Data Auditing on Blockchain with Fuzzy Deduplication and Secure Sharing

Jie Zhang, *Student Member, IEEE*, Xiaohong Li, *Member, IEEE*, Ruitao Feng, *Member, IEEE*, Shanshan Xu, Yongyang Lv, Zhe Hou and Guangdong Bai, *Member, IEEE*

***Abstract*—Cloud storage increasingly uses deduplication to reduce costs, while users require strong confidentiality and integrity for outsourced data—especially in healthcare, where sensitive, redundant data is frequently shared under strict privacy constraints. Although Message-Locked Encryption (MLE) supports encrypted deduplication, existing methods focus on exact duplicates and assume trusted auditors, making them unsuitable for medical use. We propose MediChainAudit, a blockchain-based system for medical clouds that integrates: (i) an enhanced encryption mode enabling *fuzzy* (similarity-aware) block-level deduplication, (ii) a decentralized public audit framework removing the need for trusted third parties, and (iii) a secure key-exchange and sharing protocol. Using a hybrid storage model—on-chain integrity tags and off-chain ciphertext—the system minimizes blockchain overhead while supporting verifiable secure deletion. We formalize security goals including PRV-CDA confidentiality, integrity, ownership consistency, and deduplication correctness, providing proofs under standard cryptographic assumptions. To our knowledge, MediChainAudit is the first blockchain-based solution to integrate fuzzy deduplication auditing, secure sharing, and side-channel attack defense with proven PRV-CDA security.**

***Index Terms*—Deduplication and Integrity Audit, Fuzzy Message-Locked Encryption, Privacy-preserving Data Transmission Protocol, Cloud Security, Blockchain, PRV-CDA security.**

## I. Introduction

Data-driven healthcare is fundamentally collaborative [1–3]. Advances in precision medicine rely on large-scale analysis of medical data across institutions [4, 5]. However, this necessity clashes with the paramount obligation to protect patient *confidentiality* [6, 7], making **encryption** before storage or transmission a non-negotiable standard [8].

This encryption mandate creates a practical dilemma for managing medical data, which is both highly sensitive and massively redundant [9–11]. Electronic health records (EHRs) exhibit significant *semantic similarity* across patients and over time [12–15], making **deduplication** an economic necessity rather than an optimization [16]. Standard encryption renders this redundancy unusable [17], forcing institutions into an untenable choice: incur unsustainable storage costs or compromise data security.

This tension manifests in crude and damaging real-world workarounds: institutions are forced to implement **lossy data retention policies**[1], (e.g., irrevocably deleting data older than a fixed period) that erase invaluable longitudinal research data, or resort to **highly insecure plaintext storage and highly restrictive physically-isolated storage**[2], rendering collaborative analysis exceptionally challenging.

**Unique Challenges in Medical Scenarios.** Secure deduplication cryptography, particularly Message-Locked Encryption (MLE) [18], emerges as the canonical solution in general cloud settings [8, 14, 17, 19–24]. Its promise is seductive: enable storage savings directly on ciphertexts [10, 16]. **Yet, a chasm exists between this cryptographic promise and clinical reality.** Our engagement with medical institutions and analysis of open medical datasets [25–27] reveal that off-the-shelf schemes fail catastrophically in medical contexts for two fundamental reasons:

- **Semantic Inefficiency & Side-Channel Risk:** MLE targets exact duplicates, making it profoundly inefficient for the semantic similarity prevalent in medical data [4, 9, 28, 29]. Our experiments in Section VII-B2 quantify this inefficiency on both synthetic [26, 27] and real-world clinical dataset [25]. While *fuzzy* deduplication techniques can bridge this efficiency gap, they introduce a severe security risk: the deterministic link between plaintext similarity and encryption keys makes shared ciphertexts vulnerable to frequency analysis [30–34] and brute-force attacks [20, 35, 36], especially against the low-entropy fields common in medical records. Creating a potent side channel exactly where collaboration (sharing) is intended.
- **Lack of Verifiable Audit in Cross-User Deduplication:** The core value of deduplication—storage savings across multiple users—vanishes if participants cannot trust its execution [16, 37–39]. In a multi-institutional setting, there must be a **verifiable, tamper-proof record** to prove that deduplication was performed correctly, data integrity is maintained, and ownership is unambiguous [20–22, 33, 35]. The absence of such an audit mechanism erodes the trust

This work is supported in part by the Beijing–Tianjin–Hebei Natural Science Foundation Joint Cooperation Program under Grant 25JJJJC0034, in part by the National Natural Science Foundation of China under Grant 62262073, 62332005, in part by the National Key Research and Development Program of China under Grant 2023YFB3107103.

Jie Zhang, Xiaohong Li, and Yongyang Lv are with the College of Intelligence and Computing, Tianjin University, Tianjin, China. (e-mail: {jackzhang, xiaohongli, lvyongyang}@tju.edu.cn).

Shanshan Xu is with the School of Geographic Sciences, East China Normal University, Shanghai, China. (e-mail: s.xu.ecnu@gmail.com).

Ruitao Feng is with the Faculty of Science and Engineering, Southern Cross University, Australia (e-mail: ruitao.feng@scu.edu.au).

Zhe Hou is with the School of Information and Communication Technology, Griffith University, Nathan, Australia. (e-mail: z.hou@griffith.edu.au).

Guangdong Bai is with the Department of Computer Science, City University of Hong Kong, Hong Kong, China. (e-mail: baiguangdong@gmail.com).

Jie Zhang and Xiaohong Li are contributed equally to this work.

Ruitao Feng and Guangdong Bai are the corresponding authors.

[1] *Medical Record Management Regulations (2013 Edition)* National Health Commission of China.

[2] Observed at several hospitals in Singapore, Australia and China, but we cannot mention names.

necessary for collaboration and leaves no evidence for compliance or dispute resolution.

These fundamental conflicts give rise to a distinct and hitherto underexplored challenge: the **Medical Secure Deduplication Auditing (MSDA)** problem. It demands a solution that simultaneously enables (i) storage-efficient collaboration on encrypted medical data, (ii) resistance to side-channel attacks induced by fuzzy deduplication, and (iii) tamper-proof audit trails without reliance on a trusted third party. To date, no existing system adequately satisfies this holistic set of requirements.

To solve the MSDA problem, we propose MediChainAudit, to the best of our knowledge, constitutes the first systematic solution explicitly designed for medical secure deduplication auditing. MediChainAudit is built upon three core technical innovations, each addressing one facet of the MSDA problem:

- **Fuzzy Message-Locked Encryption (FMLE):** We innovate upon the MLE paradigm by replacing its cryptographic hash with a similarity-aware hash for key derivation. This enables deduplication of semantically similar data blocks while producing *non-identical* ciphertexts for similar blocks, providing inherent resistance to plaintext-guessing attacks. Our evaluation demonstrates FMLE achieves significantly higher storage reduction **(99.90% on MIMIC-III [25], 97.61% on 100-patient dataset [26] and 95.53% on 30K-patient dataset [27])** compared to MLE (84.04%, 90.02% and 83.75% respectively).
- **Privacy-preserving Data Transmission Protocol (PDTP):** To eliminate the side-channel risk inherent in sharing fuzzy ciphertexts, PDTP securely transmits only the minimal sufficient secret (fuzzy hashes) between authorized parties using authenticated key exchange. This ensures that any intercepted ciphertext remains perpetually unintelligible to adversaries, directly mitigating the second flaw in medical data sharing.
- **Blockchain-based Decentralized Audit:** We design and embed a lightweight blockchain audit layer that maintains an immutable, cryptographically signed ledger of all data operations (upload, deduplication, sharing, access). This component provides publicly verifiable proof of correct deduplication and data ownership, enabling compliance auditing and dispute resolution without any trusted third party.

Together, these three components form the backbone of MediChainAudit. We have implemented the complete system and evaluated it on both synthetic and real-world medical datasets. Experimental results demonstrate that MediChainAudit successfully reconciles the conflicting goals of storage efficiency, security, and auditability: it achieves near-optimal deduplication efficiency, provides robust defense against side-channel attacks, and introduces manageable overhead suitable for production medical environments.

**In summary, our contributions are:**

- The first formalization of the **MSDA** problem, explicitly identifying the unique security, efficiency, and auditability requirements that are unmet in current medical data sharing practices.
- The design of two novel cryptographic primitives: **FMLE** for secure and efficient fuzzy deduplication, and **PDTP** for side-channel-resistant data sharing.
- MediChainAudit: the first system that systematically addresses the MSDA problem by holistically combining FMLE, PDTP, and a blockchain-based audit layer. We provide a full-fledged implementation and validate its practical feasibility through extensive experiments on real clinical datasets.
- Open-source release of our implementation and datasets at https://github.com/JackAugust/MediChainAudit.git to facilitate reproducibility and further research.

TABLE I
FUNCTIONAL COMPARISON WITH REPRESENTATIVE WORKS

| Scheme | Third-party auditor | Deduplication granularity | Security reduction | T1 | T2 | T3 | T4 | T5 | T6 | T7 |
|---|---|---|---|---|---|---|---|---|---|---|
| [33] | cloud | block-level | CR | - | ✓ | ✓ | ✓ | - | ✓ | - |
| [36] | cloud | file-level | CR | - | ✓ | ✓ | ✓ | ✓ | ✓ | - |
| [40] | cloud | file-level | DL | - | ✓ | ✓ | - | ✓ | ✓ | - |
| [41] | cloud | file-level | CR | - | ✓ | ✓ | ✓ | - | ✓ | - |
| [42] | cloud | file-level | CR | - | ✓ | ✓ | ✓ | - | ✓ | - |
| [17] | blockchain | file-level | DL | ✓ | ✓ | - | - | ✓ | - | - |
| [8] | blockchain | block-level | DL | ✓ | ✓ | - | - | ✓ | ✓ | - |
| [22] | blockchain | file-level | DL, CDH | ✓ | ✓ | - | - | - | - | - |
| [23] | blockchain | block-level | ECDLP | ✓ | ✓ | - | - | - | - | - |
| [20] | blockchain | block-level | DL, CDH | ✓ | ✓ | - | - | - | - | - |
| Our | blockchain | block-level | ECDLP | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ |

DL: Discrete Logarithm; CDH: Computational Diffie-Hellman; CR: Collusion Resistance; ECDLP: Elliptic Curve Discrete Logarithm Problem. **T1:** Data integrity audit; **T2:** Security deduplication; **T3:** Fuzzy support; **T4:** Tag consistency audit; **T5:** Privacy protection; **T6:** Side-channel attack defense; **T7:** Secure transmission protocol.

## II. RELATED WORK

### A. Blockchain-based Auditing for Encrypted Deduplication Schemes

The verifiability of Proof of Retrievability (PoR) and data integrity in outsourced environments is a core concern of cloud security [43]. Since Armknecht *et al.* [44] introduced outsourcing the audit task to a third party, ensuring the trustworthiness of the auditor itself became a key challenge [45]. Blockchain, with its decentralization, immutability, and programmability, has emerged as ideal infrastructure to replace traditional TPAs. For instance, Lin *et al.* [37] pioneered a fully decentralized, TPA-free public auditing framework; Wang *et al.* [38] designed a smart contract-based arbitration protocol for multi-cloud auditing; Miao *et al.* [39] designed smart contracts to replace the TPA and achieve multitask auditing.

Conversely, to balance privacy with storage efficiency, Deduplication Schemes for Encrypted Data (DSED) have become common in cloud storage, yet their auditing remains inadequately addressed [16, 21, 33]. Researchers are exploring the use of blockchain technology to achieve trustworthy deduplication auditing. Vivekrabinson *et al.* [10] and Zhang *et al.* [35] proposed using blockchain for deduplication auditing, employing smart contracts to defend against collusion between CSPs and users. Zhang *et al.* [9], Li *et al.* [23], and Tian *et al.* [20] stored data tags and deduplication records on-chain to achieve transparent and verifiable auditing; Miao *et al.* [22] optimized key derivation mechanisms to reduce verification

overhead; Song *et al.* [8] and Zhang *et al.* [17] introduced enhanced cryptographic primitives to protect both file content and user identity privacy during auditing.

### B. Auditing Challenges and Limitations for Medical Data

Despite the promising performance of the aforementioned schemes in general scenarios, they are not directly suitable for medical data. Research on auditing for medical clouds is still in its infancy. Zhang *et al.* [14] and Vivekrabinson *et al.* [10] proposed a ciphertext deduplication scheme for Electronic Medical Records (EMRs); Fu *et al.* [16] designed a tiered encryption deduplication protocol; Xiao *et al.* [11] attempted to introduce blockchain for an auditing strategy. However, these works fail to fully consider the particularities of medical data, exhibiting two critical shortcomings:

First, **medical data exhibits semantic rather than identical redundancy**. The fundamental conflict between the similarity-based redundancy of medical data and the exact-match assumption of MLE significantly reduces deduplication efficiency. Existing fuzzy deduplication methods often rely on feature extraction (e.g., perceptual hashing [40, 41]), cosine similarity [29], or Hamming distance [36, 42] for similarity judgment. However, these methods have not been effectively validated or optimized for medical data. Our work leverages Locality-Sensitive Hashing (LSH) to propose a similarity-preserving auditing mechanism compatible with EHRs.

Second, **MLE's deterministic encryption is vulnerable to side-channel attacks**, a particularly critical issue given the strong privacy requirements of medical data. On one hand, ciphertext frequency distribution can leak statistical patterns of the plaintext [30, 32, 33, 46]. On the other hand, the duplicate-checking process can expose file existence information [21, 31, 34, 45]. To counter frequency analysis, we adopt MinHash and randomization strategies [30, 33, 36, 43]. Against duplicate-check leaks, we design a verifiable yet privacy-preserving auditing protocol allowing the server to perform deduplication verification without learning specific file content.

### C. Positioning Within the Research Landscape

To clearly delineate the contribution of MediChainAudit, it is instructive to contrast it with two prevalent strands of related work. First, **general healthcare blockchain frameworks** (e.g., MedRec [15], MedShare [38]) primarily leverage blockchain for decentralized data governance, addressing *interoperability, patient consent, and data provenance*. However, they operate at the data-object level and do not tackle the internal *storage inefficiency* of encrypted data, which is a critical cost factor. Second, **secure deduplication schemes** (e.g., ESDedup [11], HealthDep [14]), focus on storage efficiency but lack audit mechanisms for the secure sharing.

Our system is designed to bridge the gap between these two paradigms by directly addressing the **MSDA (Medical Secure Deduplication Auditing)** problem. As the functional comparison in Table I illustrates, we evaluate prior works across capabilities essential for medical data management: (1) data integrity auditing, (2) secure deduplication, (3) fuzzy support, (4) tag consistency audit, (5) privacy protection, (6) side-channel attack defense, and (7) secure transmission protocol. Comparative analysis shows that although existing secure deduplication schemes have made progress in achieving fuzzy deduplication or leveraging blockchain for trusted integrity auditing, they predominantly focus on exact duplicates and fail to address the unique requirements of medical cloud environments—holistically integrating fuzzy block-level deduplication, scalable integrity and consistency auditing, side-channel attack resilience, and a secure sharing protocol—features indispensable for practical and secure medical data management in the cloud. Our system, MediChainAudit, is designed to bridge this gap.

## III. Preliminary

### A. Notations

**Shorthands.** We denote by $s \xleftarrow{\$} \mathcal{S}$ the uniform random selection and assignment of an element from set $\mathcal{S}$ to variable $s$, where $|\mathcal{S}|$ denotes the number of elements in the set $\mathcal{S}$ . The symbol $\|$ represents string concatenation, while $\oplus$ denotes the exclusive OR (XOR) operation. The truth values are expressed as $\top$ (verum) and $\bot$ (falsum).

**Cryptographic Parameters.** Let $\lambda$ be the security parameter. The scalars lie in a field $\mathbb{F}_p$ with prime order $p$, which is integers modulo prime $p$, $i.e.$, $\mathbb{Z}_p$. Its size is exponential in $\lambda$.

**Group Notations.** We denote group elements in upper case ($e.g.$, group generator $G$ and group elements $X, Y$) and their secret scalar in lower case ($e.g.$, $x$ for $X := x \cdot G$ and $y$ for $Y := y \cdot G$).

**Function Notations.** Let hash function $\mathsf{H} : \{0,1\}^* \to \{0,1\}^\lambda$ that maps to $\mathbb{F}_p$, hash function $\mathsf{H}_2 : \{0,1\}^\lambda \to \{0,1\}^{2\lambda}$. Let $o \leftarrow \boldsymbol{SC}(\cdot)$ denote that $o$ is an output of the smart contract $\boldsymbol{SC}(\cdot)$.

### B. AES and Message-Locked Encryption Algorithm

With its high performance and strong encryption and decryption security, Advanced Encryption Standard (AES) [47] is widely used in today's information systems, including cloud and edge scenarios. AES consists of several encryption modes. Among these, AES-CTR (Counter Mode) is used by introducing a random initialization vector $IV \in \{0,1\}^{128}$ to avoid generating the same ciphertext for the same data under the same key. The algorithm is described as: Let $M := \{m_1 \parallel \cdots \parallel m_n\}$ be the plaintext segmented into 128-bit blocks. The ciphertext $C := \{c_1 \parallel \cdots \parallel c_n\}$ is generated through $c_i := m_i \oplus \mathsf{AES}(key, IV + i)$, and the plaintext $M$ is recovered by $m_i := c_i \oplus \mathsf{AES}(key, IV + i)$.

MLE [18] is a special type of symmetric encryption, in which encryption/decryption keys are extracted from messages themselves, meaning that the same encryption result can be obtained for the same data. This feature supports the cloud for secure duplicate ciphertext removal. From an MLE scheme, the key $k := \mathsf{H}(M)$ is derived from a message $M$, then ciphertext $C := \mathsf{Enc}(k, M)$ is generated by the encryption algorithm, where the tag $T := \mathsf{H}(C)$ is used by the server to detect duplicates. The message $M := \mathsf{Dec}(k, C)$ is recovered by the decryption algorithm. MLE satisfies pseudo-random value

chosen-distribution attacks (PRV-CDA), where encryptions of two unpredictable messages should be indistinguishable. In our paper, we choose the CTR mode of the AES algorithm to implement the MLE scheme and make an algorithmic improvement, to obtain higher efficiency and accuracy.

### C. Locality Sensitive Hashing Function

Locality sensitive hashing (LSH) [48] constructs a specific hash function to approximate the relationship between content and its hash with a certain probability, so that similar content is mapped into the same bucket and yields the same hash value [36]. Taking MinHash-LSH [49] as an example, there exists a hash function $h \in \mathbf{LSH}$, and for any two sets $A, B$, if their Jaccard similarity is less than a specific value such as $r$, the corresponding hashes $h(A)$ and $h(B)$ have a high probability $p$ of being the same, where Jaccard similarity is calculated as follows:

$$J(A, B) = |A \cap B|/|A \cup B|$$

In this paper, we set the Jaccard similarity threshold for MinHash at 0.4 [42] as the criterion for determining similarity.

### D. Edwards-curve Digital Signature Algorithm

EdDSA [50] constitutes a provably secure signature scheme with efficient signing and verification operations. Among the two standardized Edwards curves supported by EdDSA: Ed25519 (128-bit security level) and Ed448 (224-bit security level), our implementation adopts Ed25519 due to its optimal balance between cryptographic strength and computational efficiency for typical application scenarios. The EdDSA consists of the following three algorithms, $\mathsf{KeyGen}(\cdot)$, $\mathsf{Sig}(\cdot)$, and $\mathsf{Verf}(\cdot)$, which are detailed as follows:

- $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{KeyGen}(1^\lambda, ID_i)$. Choose a secret key $\mathsf{sk}_i \xleftarrow{\$} \{0,1\}^\lambda$ and compute $k_1 \parallel \cdots \parallel k_{(2\lambda)} \leftarrow \mathsf{H}_2(\mathsf{sk}_i)$, then set $k_1 = k_2 = k_\lambda = 0, k_{\lambda-1} = 1$ and use the updated vector $(k_1, \cdots, k_\lambda)$ to define a secret value $v \in \mathbb{Z}_p$ i.e., $v := \sum_{i=1}^{\lambda} k_i \cdot 2^i \mod p$. Finally, compute the public key $\mathsf{pk}_i := v \cdot G$.
- $\sigma \leftarrow \mathsf{Sig}(\mathsf{pk}_i, \mathsf{sk}_i, M)$. Compute $k_1 \parallel \cdots \parallel k_{2\lambda} \leftarrow \mathsf{H}_2(\mathsf{sk}_i)$ and set $r \leftarrow \mathsf{H}_2(k_{\lambda+1} \parallel \cdots \parallel k_{2\lambda} \parallel M) \mod p$, then compute $R := r \cdot G$, $c := \mathsf{H}(R \parallel \mathsf{pk}_i \parallel M)$, and finally the signature $\sigma := (R, s)$ where $s := (r + c \cdot v) \mod p$.
- $\{\top/\bot\} \leftarrow \mathsf{Verf}(\mathsf{pk}_i, M, \sigma)$. Parse $\sigma := (R, s)$ then compute the hash value $c' := \mathsf{H}(R \parallel \mathsf{pk}_i \parallel M)$. Output $\top$ if $2^3 \cdot s \cdot G = 2^3 \cdot R + 2^3 \cdot c' \cdot \mathsf{pk}_i$ holds; otherwise output $\bot$.

Inspired by [51, 52], our paper uses EdDSA with the Ed25519 curve, providing a security level of $\lambda = 128$, and instantiates $\mathsf{H}_2$ with SHA256.

## IV. Problem Formulation and System Architecture

### A. The MSDA Problem

The **Medical Secure Deduplication Auditing (MSDA)** problem emerges when multiple healthcare institutions outsource encrypted medical data—characterized by high *semantic similarity*—to an honest-but-curious cloud, while requiring cross-user deduplication, secure sharing, and verifiable audit without a trusted third party. Formally, MSDA demands a solution that simultaneously satisfies four requirements:

- **R1 Confidential & Efficient Deduplication:** Enable the CSP to detect and eliminate semantically redundant ciphertexts, while providing PRV-CDA security [18] against plaintext-guessing and frequency analysis.
- **R2 Side-Channel Resilience:** Prevent adversaries from inferring plaintext similarity via network traffic or on-chain metadata during ciphertext sharing.
- **R3 Verifiable Audit Trail:** Record every data operation in a tamper-proof, publicly verifiable ledger, enabling non-repudiable compliance and dispute resolution without a centralized auditor.
- **R4 Practical Overhead:** Incur acceptable cryptographic and storage costs for large-scale medical archives (e.g., terabyte-scale, thousands of patients).

No existing system meets all four requirements: MLE [18] fails R1 (semantic inefficiency) and R2 (side-channel risk); fuzzy deduplication [33, 36] improves R1 but exacerbates R2 and lacks R3; blockchain-based auditing frameworks [37, 38] provide R3 but ignore R1 and R2. MSDA therefore represents a distinct, underexplored challenge at the intersection of cryptography, storage, and healthcare compliance.

### B. System and Threat Model

*1) System Model:* Our system involves four entities:

- **Data Owners (DOs)**: Healthcare entities that own sensitive medical data. They execute client-side FMLE encryption and the PDTP protocol.
- **Data Consumers (DCs)**: Authorized entities that access shared data; a DC may also be a DO.
- **Cloud Service Provider (CSP)**: Provides storage for encrypted data, performs deduplication checks using client-supplied fuzzy tags, and facilitates the communication channel for PDTP. It is *honest-but-curious*.
- **Blockchain Network**: A permissioned blockchain that replaces the conventional trusted third-party auditor. It maintains an immutable ledger and executes smart contracts ($\boldsymbol{Upload}(\cdot)$, $\boldsymbol{Share}(\cdot)$, $\boldsymbol{Access}(\cdot)$, $\boldsymbol{Audit}(\cdot)$) to enable decentralized auditing.

*2) Threat Model:* We consider the following adversaries; their capabilities are summarized below, while the concrete impact on each security goal is analysed in Section VI (cryptographic proofs), Section VI-C (side-channel leakage), and Section VI-E (network-level attacks).

- **Malicious CSP**: The Cloud Service Provider is *honest-but-curious* and may (i) tamper with or substitute stored ciphertexts, and (ii) eavesdrop on stored data and on network traffic (including PDTP exchanges) to infer plaintext information through frequency analysis.
- **External Network Adversary**: An active Man-in-the-Middle (MITM) that can eavesdrop on and modify all communications between users and the CSP, harvesting ciphertexts and protocol messages for offline analysis.

- **Blockchain Node Corruption**: A subset of validator nodes in the permissioned blockchain may be Byzantine. We assume a PBFT consensus [53] that tolerates up to $f$ faulty nodes among $3f + 1$; audit immutability holds within this threshold. Beyond it, clients can still detect misbehaviour via local consistency checks (Section VI-E).
- **Side-Channel Adversary**: An adversary that attempts to learn the distribution of fuzzy tags $\mathcal{T}_f$ (e.g., by repeated upload queries or by observing the Cuckoo filter) and may observe on-chain audit tags $\mathcal{T}_a$. Tag transmission is encrypted by PDTP, so the adversary cannot link tags to users (Section VI-C).
- **Network-Level Attacks**: The system faces standard DDoS [54], Sybil [55], and collusion attacks [56]. We assume the blockchain uses a Certificate Authority (CA) that binds each node to a verified real-world identity (Sybil-resistant). DDoS is partially mitigated by rate-limiting and architectural decoupling; stronger collusion resistance is discussed in Section VI-E.

### C. Design Goals

To address the MSDA requirements (R1–R4) under the threat model defined in Section IV-B, MediChainAudit pursues the following design goals:

- **G1 Confidentiality & Side-Channel Resilience**: Encrypted medical data must remain secret from the CSP and external adversaries. The system shall resist plaintext-guessing and frequency-analysis attacks (PRV-CDA security [18]), and eliminate side channels induced by sharing fuzzy ciphertexts [33, 36]. **(R1, R2)**
- **G2 Data Integrity**: Any unauthorized modification, deletion, or substitution of stored ciphertexts must be detectable with high probability by any legitimate user [20, 23]. **(R3)**
- **G3 Auditability & Non-Repudiation**: All data operations (upload, deduplication, share, access) shall be permanently logged on the blockchain in a tamper-proof, publicly verifiable manner, providing cryptographic evidence for compliance and dispute resolution [23, 37]. **(R3)**
- **G4 Storage & Computational Efficiency**: The protocols must incur minimal overhead to be practical for large-scale medical archives. Storage efficiency shall be achieved via secure cross-user deduplication of semantically similar data [33, 36]. **(R1, R4)**

### D. The MediChainAudit Architecture

As shown in Fig. 1, MediChainAudit involves four entities: **DOs**, **DCs**, the **CSP**, and a permissioned **blockchain**. The blockchain replaces the traditional TPA via smart contracts $\boldsymbol{Upload}(\cdot)$, $\boldsymbol{Share}(\cdot)$, $\boldsymbol{Access}(\cdot)$, and $\boldsymbol{Audit}(\cdot)$, providing decentralized, tamper-proof audit trails.

*1) Data Upload:* A DO encrypts a file using FMLE (in Section V-A1), generating fuzzy tags (for similarity detection) and audit tags (for integrity verification), then invokes $\boldsymbol{Upload}(\cdot)$. This contract triggers $\boldsymbol{Audit}(\cdot)$ to perform *deduplication audit* (fuzzy screening + exact tag matching) and *integrity audit* (challenge-response). If both succeed, the CSP retains the ciphertext and the DO deletes its local copy; all audit metadata (e.g., hash $C_{in}$) are immutably recorded on-chain. Detailed procedures are given in Section V-A2 and Section V-C.

*2) Data Sharing and Access:* To share data, the DO invokes $\boldsymbol{Share}(\cdot)$, marking the data as "shared". A DC calls $\boldsymbol{Access}(\cdot)$, which triggers PDTP (in Section V-B1) between DO and DC. PDTP securely transfers the minimal secret—the fuzzy tag set $\mathcal{T}_f$—via authenticated key exchange, eliminating sharing-induced side channels. After obtaining $\mathcal{T}_f$, the DC downloads the ciphertext from the CSP and invokes $\boldsymbol{Audit}(\cdot)$ again for *consistency audit* (matching the ciphertext against on-chain $C_{in}$) and *key validity audit* (verifying the decryption key against the audit tags). Only after both verifications succeed can the DC decrypt the data. These audits are elaborated in Section V-B2 and Section V-C.

*3) CSP & Blockchain Roles:* The CSP acts solely as a storage backend for FMLE ciphertexts and a communication channel for PDTP; it never accesses plaintext or keys. All smart contracts run on the blockchain, generating immutable transactions (e.g., 'TxUpload', 'TxShare', 'TxAccess') that serve as non-repudiable evidence for compliance and dispute resolution.

In summary, MediChainAudit integrates FMLE and PDTP with a blockchain audit layer, addressing the MSDA problem (R1–R4) and satisfying design goals G1–G4 without any centralized trusted entity. Cryptographic details and security analysis follow in Section V.

## V. The MediChainAudit Framework

This section presents the detailed construction of the MediChainAudit framework. We first formalize the core cryptographic components—Fuzzy Message-Locked Encryption (FMLE) and the Privacy-preserving Data Transmission Protocol (PDTP)—and then describe how they are integrated into the system's data lifecycle. We further illustrate how these primitives interact with blockchain smart contracts to form a cohesive and auditable system that effectively addresses the Medical Secure Data Auditing (MSDA) problem.

### A. Integrated Construction of FMLE and System Data Flow

We integrate the description of the FMLE scheme with its usage during the data upload process. This combined presentation clarifies how encryption, deduplication, and integrity assurance are achieved in practice.

*1) Fuzzy Message-Locked Encryption (FMLE) in Data Upload:* The FMLE scheme is a tuple of probabilistic polynomial-time algorithms (FMLE.KeyGen, FMLE.Enc, FMLE.Dec) that extends the standard MLE notion [18] to handle semantic similarity.

- $\mathsf{FMLE.KeyGen}(\tilde{\mathcal{T}}) \to key$: On input the raw fuzzy tag set $\tilde{\mathcal{T}} = (\tilde{t}_1^f, \ldots, \tilde{t}_n^f)$, this algorithm outputs a symmetric key $key = \mathsf{H}_2(\tilde{t}_1^f \oplus \cdots \oplus \tilde{t}_n^f)$.
- $\mathsf{FMLE.Enc}(key, F) \to (C, \mathcal{T}_f, \mathcal{T}_a)$: The algorithm outputs the ciphertext $C$, the public (hashed) fuzzy tag set $\mathcal{T}_f = (\mathsf{H}_2(\tilde{t}_1^f), \ldots, \mathsf{H}_2(\tilde{t}_n^f))$, and the audit tag set $\mathcal{T}_a$, the encryption algorithm works as follows:

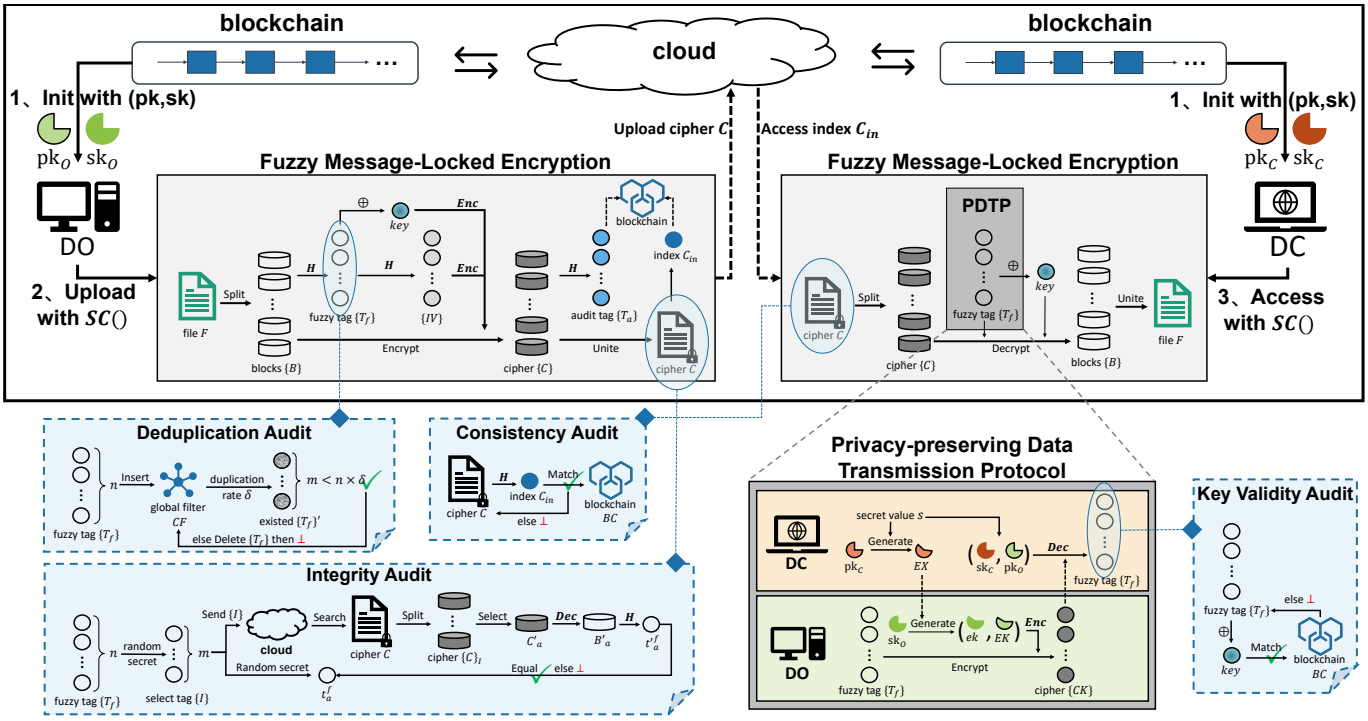


Fig. 1. The MediChainAudit framework with FMLE and PDTP.

– **Blocking**: Split $F$ into $n$ fixed-size blocks $F = (b_1, \ldots, b_n)$ (e.g., $|b_i| = 4$KB [57][3]).
– **Fuzzy Tag Generation**: For each block $b_i$, compute a raw fuzzy tag $\tilde{t}_i^f \leftarrow \mathsf{MinHash}(b_i)$. The *public* fuzzy tag is then defined as $t_i^f \leftarrow \mathsf{H}_2(\tilde{t}_i^f)$, and $\mathcal{T}_f = (t_1^f, \ldots, t_n^f)$.
– **Encryption**: For each block $b_i$, derive a tag-derived initialization vector $IV_i \leftarrow \mathsf{H}(\tilde{t}_i^f)$, the encryption key is derived from the raw tags as $key \leftarrow \mathsf{H}_2(\oplus_{i=1}^n \tilde{t}_i^f)$. Encrypt the block using AES in CTR mode: $c_i \leftarrow \mathsf{Enc}(key, IV_i, b_i)$. The ciphertext is $C = (c_1 \parallel \ldots \parallel c_n)$.
– **Audit Tag Generation**: For each ciphertext block $c_i$, compute a secure *audit tag* $\tau_i \leftarrow \mathsf{H}(c_i)$, and $\tau_0 \leftarrow \mathsf{H}(key)$. Let $\mathcal{T}_a = (\tau_0, \tau_1, \ldots, \tau_n)$. These tags are used for integrity verification and deduplication on the blockchain.

- $\mathsf{FMLE.Dec}(key, \tilde{\mathcal{T}}_f, C) \rightarrow F$: On input $key$, the *raw* fuzzy tag set $\tilde{\mathcal{T}}_f = (\tilde{t}_1^f, \ldots, \tilde{t}_n^f)$, and ciphertext $C$, parse $C$ into blocks $(c_1, \ldots, c_n)$. For each block $c_i$, derive $IV_i \leftarrow \mathsf{H}(\tilde{t}_i^f)$ and decrypt $b_i \leftarrow \mathsf{Dec}(key, IV_i, c_i)$. Reconstruct the file $F = (b_1 \parallel \ldots \parallel b_n)$.

**Integration with Upload Process.** Before data upload, all users must register with the blockchain-based CA. For a user $i$ with unique identifier $ID_i$, the CA verifies the uniqueness of $ID_i$ and generates an EdDSA key pair:

$$(pk_i, sk_i) \leftarrow \mathsf{KeyGen}(1^\lambda, ID_i) \tag{1}$$

To upload a local medical data $Obj$, the Data Owner (DO) performs FMLE.Enc to partition $Obj$ into $n$ fixed-size blocks $Obj = (b_1, \ldots, b_n)$ with $|b_i| = 4$KB. For each block $b_i$, a raw fuzzy tag is computed using MinHash-LSH $\tilde{t}_i^f \leftarrow \mathsf{MinHash}(b_i)$. The public (hashed) fuzzy tag is $t_i^f \leftarrow \mathsf{H}_2(\tilde{t}_i^f)$. The encryption key is derived from the raw tags: $key \leftarrow \mathsf{H}_2(\tilde{t}_1^f \oplus \cdots \oplus \tilde{t}_n^f)$, and the tag-derived IV is $IV_i \leftarrow \mathsf{H}(\tilde{t}_i^f)$. Each block is encrypted using AES in CTR mode $\mathsf{Enc}(\cdot)$ with a tag-derived initialization vector: $c_i \leftarrow \mathsf{Enc}(key, IV_i, b_i)$, where $IV_i \leftarrow \mathsf{H}(\tilde{t}_i^f)$. The ciphertext is $\mathcal{C} = (c_1 \parallel \ldots \parallel c_n)$, and for each ciphertext block, an audit tag $\tau_i \leftarrow \mathsf{H}(c_i)$ is computed for integrity verification, with $\mathcal{T}_a = (\tau_0, \tau_1, \ldots, \tau_n)$. The fuzzy tag set $\mathcal{T}_f = (t_1^f, \ldots, t_n^f)$ is used for deduplication. This tight integration demonstrates how FMLE directly enables privacy-preserving deduplication and prepares data for auditable storage.

$$(\mathcal{C}, \mathcal{T}_f, \mathcal{T}_a) \leftarrow \mathsf{FMLE.Enc}(key, Obj) \tag{2}$$

**Security Properties.** FMLE provides two crucial properties: Fuzzy Deduplication and Confidentiality, that breaks the deterministic link between plaintext similarity and ciphertext. Semantically similar blocks generate identical fuzzy tags $t_i^f$ with high probability, thereby enabling fuzzy deduplication and preventing plaintext information leakage merely by verifying the tags during upload. Since $IV_i$ varies per block, *distinct* plaintext blocks (similar or not) yield independent ciphertexts and thus independent audit tags $\tau_i$. This prevents an adversary from inferring plaintext relationships by observing the public $\mathcal{T}_a$ on the blockchain, effectively mitigating this side-channel.

*2) Deduplication Audit during Upload:* Before uploading, the $\boldsymbol{Upload}(\cdot)$ smart contract invokes the $\boldsymbol{Audit}(\cdot)$ to perform deduplication checking. This process utilizes a two-tiered tagging system: the fuzzy tag set $\mathcal{T}_f$ for efficient similarity estimation, and the audit tag set $\mathcal{T}_a$ for cryptographically exact deduplication decisions.

[3]According to *Ng et al.*'s research, the best chunking efficiency can be achieved by adopting a block size of 4KB.

**Phase 1: Efficient Similarity Screening.** For a file $Obj$ encrypted as $(\mathcal{C}, \mathcal{T}_f, \mathcal{T}_a) \leftarrow \mathsf{FMLE.Enc}(key, Obj)$, the fuzzy tag set $\mathcal{T}_f$ is checked against a global, memory-efficient *cuckoo filter* ($CF$). This filter [58] provides a fast probabilistic membership test. The system calculates the preliminary duplication rate $\delta$ as:

$$\delta = \frac{|\{t_i^f \in \mathcal{T}_f : t_i^f \in CF\}|}{|\mathcal{T}_f|}. \tag{3}$$

A configurable threshold (e.g., $\delta \geq 0.1$) allows the system to reject uploads with excessively high preliminary duplication rates, guarding against the filter's false positives and unnecessary processing.

**Phase 2: Exact Deduplication via Ciphertext Audit Tags.** Crucially, a high $\delta$ or even $\delta = 1$ (all fuzzy tags matched) does not conclude the deduplication. The **audit tag set** $\mathcal{T}_a$, derived from the ciphertext blocks ($\tau_i = \mathsf{H}(c_i)$), serves as the unique fingerprint of the *actual encrypted data* $\mathcal{C}$. The final deduplication decision is based on an exact match of $\mathcal{T}_a$ against the corresponding set stored on the blockchain ($\mathcal{T}_a^*$).

- **Upload and Record:** If $\mathcal{T}_a$ is not found on-chain or does not match the existing $\mathcal{T}_a^*$, the ciphertext $\mathcal{C}$ is uploaded to the CSP, and a new record with $\mathcal{T}_a$ is immutably stored on the blockchain.
- **Ownership Update Only:** An upload is bypassed **only if** a perfect match $\mathcal{T}_a = \mathcal{T}_a^*$ is verified on-chain. This proves the identical ciphertext already exists, and only the new user's ownership is appended to the existing record.

This design ensures **data fidelity**. Semantically similar plaintext blocks share fuzzy tags ($t_i^f$) but, due to unique initialization vectors ($IV_i$), yield distinct ciphertexts and thus distinct audit tags ($\tau_i$). Therefore, the system never incorrectly merges different medical records or updates. The audit guarantees that users always retrieve the precise data they uploaded, addressing the extreme case where all blocks are similar but not identical to existing data. The complete procedure is detailed in Algorithm 1.

This deduplication process ensures storage efficiency (**G4: Efficiency**) while preventing redundant uploads. The use of MinHash-LSH enables semantic deduplication, addressing the semantic redundancy mismatch in medical data.

*3) Integrity Verification and Blockchain Recording:* Upon successful deduplication audit, the ciphertext $\mathcal{C}'$ is uploaded to the CSP for the first time, the ciphertext $\mathcal{C}$ still exists locally. Then $\boldsymbol{Audit}(\cdot)$ will be called to generate the following challenges to complete the integrity audit.

- $\mathsf{chal} \leftarrow \boldsymbol{Audit}.\mathsf{Chal}^{(\lambda)}(ID, \mathcal{C})$: User $ID$ computes the challenge $\mathsf{chal} := \mathsf{H}_2(\mathcal{C})$.
- $\mathsf{proof} \leftarrow \boldsymbol{Audit}.\mathsf{Proof}^{(\lambda)}(\mathcal{C}')$: The CSP computes a proof $\mathsf{proof} := \mathsf{H}_2(\mathcal{C}')$ for the stored ciphertext $\mathcal{C}'$.
- $\{\top/\bot\} \leftarrow \boldsymbol{Audit}.\mathsf{Verf}(\mathsf{chal}, \mathsf{proof})$: The verifier outputs $\top$ if $\mathsf{chal} = \mathsf{proof}$; otherwise outputs $\bot$.

**Notice** that due to the public nature of records $Tx_{up}(C_{in})$ in the blockchain, data integrity is not directly verified at this point by the above challenges, i.e., if $\boldsymbol{Audit}(\cdot)$ requires the CSP to verify the integrity of $\mathcal{C}$, in the $\boldsymbol{Audit}.\mathsf{Proof}$ phase the CSP can ensure that it passes the integrity audit by simply returning the public $Tx_{up}(C_{in})$ as the proof rather than to calculate $\mathsf{H}_2(\mathcal{C}')$. The continuous integrity monitoring is described in Section V-C.

**Algorithm 1** Deduplication Audit

**Input:** User $ID$'s data $Obj$, key $key$, global Cuckoo Filter $CF$, duplication threshold $\delta$
**Output:** Audit result $\top/\bot$

$(\mathcal{C}, \mathcal{T}_f, \mathcal{T}_a) \leftarrow \mathsf{FMLE.Enc}(key, Obj)$ ▷ Encrypt data
$\mathcal{T} \leftarrow \{\mathsf{H}(t_i^f) | t_i^f \in \mathcal{T}_f\}$ ▷ Generate hashed tag set
$n \leftarrow 0, \mathcal{T}' = \mathsf{nil}$ ▷ Initialize counter $n$ and tags $\mathcal{T}'$
**for** each $T_i \in \mathcal{T}$ **do**
  **if** $T_i \in CF$ **then**
    $n \leftarrow n + 1$, $record \leftarrow$ Blockchain.Search$(T_i)$ ▷ Get existing record by querying blockchain
    **if** $\mathcal{T} = record.\mathcal{T}$ **and** $\mathcal{T}_a = record.\mathcal{T}_a$ **then**
      $record.Owner \leftarrow record.Owner \cup \{ID\}$
      **return** $\top$ ▷ Ownership updated, data extended
    **else if** $\mathcal{T} \subset record.\mathcal{T}$ **then**
      **return** $\bot$ ▷ Data already exists
    **else if** $record.\mathcal{T} \subset \mathcal{T}$ **then**
      $CF \leftarrow CF \cup (\mathcal{T} - record.\mathcal{T})$ ▷ Add extra tags
      $record \leftarrow record.\{Owner \cup \{ID\}, \mathcal{T}, \sigma, Now\}$ ▷ Blockchain.Update $record$'s ownership, tag set,signature and timestamp
      **return** $\top$ ▷ Data extended
    **else**
      **if** $n \geq \delta \times |\mathcal{T}|$ **then**
        $CF \leftarrow CF \setminus \mathcal{T}'$ ▷ Revoke upload
        **return** $\bot$ ▷ Audit failure
      **end if**
    **end if**
  **else**
    $CF \leftarrow CF \cup \{T_i\}$ ▷ Add new tag to filter
    $\mathcal{T}' \leftarrow \mathcal{T}' \cup \{T_i\}$
  **end if**
**end for**
Blockchain.Add $record$ with $\mathcal{T}'$,$\mathcal{T}_a$, and user $i$
**return** $\top$ ▷ Upload completed

After passing the integrity audit, allowing the DO to safely delete the local copy of $\mathcal{C}$ and retain only the key set $\mathcal{K}$ : $\{key, t_1^f, \ldots, t_n^f\}$. The $\boldsymbol{Upload}(\cdot)$ then generates a signature and records the transaction on the blockchain:

$$\sigma \leftarrow \mathsf{Sign}(sk_i, \mathcal{T}_a \parallel \mathsf{timestamp}) \tag{4}$$

$$Tx_{up}(C_{in}) \leftarrow \{\mathcal{T}_a, \sigma, ID, \mathsf{Upload}, \mathcal{S}(C_{in}), \mathsf{timestamp}\} \tag{5}$$

where $C_{in} = \mathsf{H}_2(\mathcal{C})$ is the ciphertext index and $\mathcal{S}(C_{in}) = (\top, \bot)$ indicates the data exists ($\top$) but is not yet shared ($\bot$), signature $\sigma$ serves as a cryptographic ownership proof, immutably linking the uploader $ID$ to the data index $C_{in}$. The ownership list is updated if necessary.

This upload process, integrating deduplication and integrity audits, counters data tampering and loss by malicious CSPs (**G2: Integrity**), while achieving storage savings via secure cross-user deduplication (**G4: Efficiency**). FMLE provides semantic security against frequency analysis and inherently resists sharing-induced side channels (**G1: Confidentiality &**

**Side-Channel Resilience**), and the on-chain recording of audit tags creates an immutable, non-repudiable audit trail (**G3: Auditability**).

### *B. Integrated Description of PDTP and Data Sharing*

We now describe the Privacy-preserving Data Transmission Protocol (PDTP) in the context of data sharing, making explicit how secure key exchange enables authorized access.

*1) Privacy-preserving Data Transmission Protocol (PDTP) in Data Sharing:* PDTP is an authenticated key exchange protocol designed to securely transfer the fuzzy tag set $\mathcal{T}_f$, which is both necessary and sufficient for decryption. It operates between a Data Owner ($DO$) and a Data Consumer ($DC$), both possessing EdDSA key pairs [50] $(\mathsf{sk}_O, \mathsf{pk}_O)$ and $(\mathsf{sk}_C, \mathsf{pk}_C)$, respectively.

- **Request**. $DC$ generates an ephemeral secret $s \xleftarrow{\$} \mathbb{Z}_p$, computes $T \leftarrow s \cdot G$ (where $G$ is a generator on an elliptic curve), and signs the request: $\sigma_C \leftarrow \mathsf{Sign}(\mathsf{sk}_C, T \parallel \mathsf{pk}_O)$. It sends $(T, \sigma_C)$ to $DO$.
- **Response**. $DO$ verifies $\sigma_C$ against $\mathsf{pk}_C$ and the message $T \parallel \mathsf{pk}_O$. If valid, it computes the shared secret $SK \leftarrow \mathsf{H}_2(\mathsf{sk}_O \cdot T)$. It then encrypts the fuzzy tag set $\mathcal{T}_f$: $\mathcal{C}_\mathcal{T} \leftarrow \mathsf{Enc}(SK, \mathcal{T}_f)$. Finally, it signs the response: $\sigma_O \leftarrow \mathsf{Sign}(\mathsf{sk}_O, \mathcal{C}_\mathcal{T})$ and sends $(\mathcal{C}_\mathcal{T}, \sigma_O)$ to $DC$.
- **Completion**. $DC$ verifies $\sigma_O$ and computes the same shared secret $SK \leftarrow \mathsf{H}_2(s \cdot \mathsf{pk}_O)$. It decrypts the tag set: $\mathcal{T}_f \leftarrow \mathsf{Dec}(SK, \mathcal{C}_\mathcal{T})$.

**Integration with Sharing and Access.** Upon receiving $\mathcal{T}_f$ via PDTP, the $DC$ can subsequently access and decrypt the corresponding ciphertext blocks from storage. The protocol ensures that the sensitive $\mathcal{T}_f$ is transmitted confidentially and with integrity, mitigating sharing-induced side channels. This illustrates how PDTP is essential for secure and authorized data access.

*2) Data Sharing and Access with PDTP:* The data sharing and access phase in MediChainAudit addresses the peril of shared ciphertexts through the PDTP protocol (Section V-B1) and multi-layered auditing mechanisms.

**Phase 1: Sharing Initiation.** To share data with index $C_{in}$, any user in the ownership list (including the original uploader or users added through deduplication) can invoke the $\boldsymbol{Share}(\cdot)$. The contract verifies that the caller is in the ownership list and updates the data state: $\mathcal{S}(C_{in}) \leftarrow (\top, \top)$. This state change indicates the data exists and is now shared. The shared state is immutable for all owners; once set, it remains shared forever.

**Phase 2: Privacy-preserving Data Transmission Protocol.** When a DC requests access to shared data, the parties engage in the PDTP protocol to securely transfer the decryption key set $\mathcal{K}$, which is the minimal sufficient secret for decryption. This protocol directly addresses the sharing-induced side channel risk. The PDTP protocol proceeds as follows:

**Step 1 -** *DC initiates the request.* The DC generates an ephemeral secret $s \xleftarrow{\$} \mathbb{Z}_p$ and computes $EX := s \cdot \mathsf{pk}_C$, then compute the signature $\sigma_C \leftarrow \mathsf{Sign}(\mathsf{pk}_C, \mathsf{sk}_C, (EX \parallel C_{in}))$, and sends the request $\{(EX \parallel C_{in}), \sigma_C)$ to the DO.

**Step 2 -** *DO responds to requests.* Upon receipt of the request, The DO verifies $\sigma_C$ by running $\mathsf{Verf}(\mathsf{pk}_C, (EX \parallel C_{in}), \sigma_C)$, If valid, the DO computes the shared secret: $EK := \mathsf{H}_2(\mathsf{sk}_O \cdot EX)$ and $ek := \mathsf{H}(\mathsf{sk}_O \cdot EX)$, and for the local key set $\mathcal{K} : \{key, \tilde{t}_1^f, \ldots, \tilde{t}_n^f\}$, encrypts the $t_i^f$ locally to get $ct_i := \tilde{t}_i^f \oplus \mathsf{H}_2(\mathsf{AES}(EK, ek))$, then the ciphertext of key set $\mathcal{CK} := \{ct_1, \cdots, ct_n\}$ and signs the response $\sigma_O \leftarrow \mathsf{Sign}(\mathsf{pk}_O, \mathsf{sk}_O, \mathcal{CK})$. The DO sends $(\mathcal{CK}, \sigma_O)$ to the DC. **Note** that for the encryption of $\tilde{t}_i^f$, we do not exactly use the AES, exactly, the order of arrival of $ct_i$ can vary due to network fluctuations, we choose to "$\oplus$" fixed $\mathsf{AES}(EK, ek)$ to reduce the likelihood of decryption failure. Our security proves that it is impossible to decrypt $\tilde{t}_i^f$ from $ct_i$ without knowing $EK$ and $ek$, thus guaranteeing the security of $\tilde{t}_i^f$.

**Step 3 -** *DC verifies DO response.* The DC verifies $\sigma_O$ and computes the same shared secret $EK' := \mathsf{H}_2(s \cdot \mathsf{sk}_C \cdot \mathsf{pk}_O)$ and $ek' := \mathsf{H}(s \cdot \mathsf{sk}_C \cdot \mathsf{pk}_O)$, then the tag set $t_i' := ct_i \oplus \mathsf{H}_2(\mathsf{AES}(EK', ek'))$ is decrypted locally for each $ct_i \in \mathcal{CK}$ to get $\mathcal{K}' := \{t_i', \cdots, t_n'\}$.

By transmitting the sensitive $\mathcal{T}_f$ confidentially via authenticated key exchange and eliminating the sharing-induced side channel, PDTP directly fulfills **G1 (Confidentiality & Side-Channel Resilience)**.

### *C. Key Validity & Continuous Integrity & Consistency Audit*

With the core components and their operational roles established, we now present the auditing mechanisms that ensure long-term security. The blockchain-based audit mechanism continuously verifies data integrity and key validity. The smart contract $\mathcal{SC}$ periodically issues challenges to storage providers. For a random set of block indices, the provider must return the corresponding audit tags $\tau_i$ along with a cryptographic proof. The contract verifies the proof against the on-chain commitments, ensuring that the data remains unaltered and accessible.

*1) Key Validity Audit:* After receiving the decryption keys $\mathcal{CK}$, the DC invokes the $\boldsymbol{Audit}(\cdot)$ to verify the validity of the key set $\mathcal{K}'$ before use.

- $\mathsf{chal} \leftarrow \boldsymbol{Audit}.\mathsf{Chal}^{(\lambda)}(ID, \mathcal{K}')$: The requester $ID$ parses $\mathcal{K}' := (t_1', \cdots, t_n')$, computes $\forall t_i' \in \mathcal{K}', T_i' := \mathsf{H}_2(t_i')$ where $i \in [1, n]$, and $key' := \mathsf{H}_2(\oplus_{i=1}^n t_i')$, then outputs $\mathsf{chal} := \mathsf{H}(key')$.
- $\mathsf{proof} \leftarrow \boldsymbol{Audit}.\mathsf{Proof}^{(\lambda)}(C_{in})$: Retrieve the latest record $Tx_{up}(C_{in})$ from the blockchain and return $\mathcal{T}_a$ as $\mathsf{proof} := \mathcal{T}_a$.
- $\{\top/\bot\} \leftarrow \boldsymbol{Audit}.\mathsf{Verf}(\mathsf{chal}, \mathsf{proof})$: Output $\top$ if $\mathsf{chal} \subseteq \mathsf{proof}$ holds; otherwise output $\bot$.

This audit prevents malicious CSPs from providing incorrect keys, addressing **G2 (Integrity)** by ensuring the received keys match the original data.

*2) Continuous Integrity Audit:* DOs can periodically initiate integrity challenges to ensure ongoing data protection. The challenge-response protocol works as follows:

- $\mathsf{chal} \leftarrow \boldsymbol{Audit}.\mathsf{Chal}^{(m)}(ID, \mathcal{K})$: The owner $ID$ of ciphertext $\mathcal{C}$ parses $\mathcal{K} : \{key, \tilde{t}_1^f, \ldots, \tilde{t}_n^f\}$, randomly selects $\mathcal{I} \xleftarrow{\$} \{0,1\}^\lambda \mod n$ where $||\mathcal{I}|| = m$ and $\forall i, j \in \mathcal{I}, i \neq j$,

randomly selects $a \xleftarrow{\$} \mathcal{I}$, retrieves $\tilde{t}_a^f$ from $\mathcal{K}$, and computes $\mathsf{chal} := (\mathcal{I}, \tilde{t}_a^f \cdot \mathsf{pk}_O)$.
- $\mathsf{proof} \leftarrow \boldsymbol{Audit}.\mathsf{Proof}^{(m)}(\mathsf{chal}, \mathcal{C})$: The CSP parses $\mathsf{chal} := (\mathcal{I}, \tilde{t}_a^f \cdot \mathsf{pk}_O)$ and $\mathcal{C} := (c_1 \parallel \cdots \parallel c_n)$, then selects $\forall i \in \mathcal{I}$, $c_i \in \mathcal{C}$ to form $\mathsf{proof} := \{c_i\}_{i \in \mathcal{I}, c_i \in \mathcal{C}}$.
- $\{\top/\bot\} \leftarrow \boldsymbol{Audit}.\mathsf{Verf}(\mathsf{chal}, \mathsf{proof})$: The DO parses $\mathsf{proof} := \{c_i\}_{i \in \mathcal{I}, c_i \in \mathcal{C}}$ to get $c_a \in \mathsf{proof}$ where $a \in \mathcal{I}$, retrieves $\tilde{t}_a^f$ from $\mathcal{K}$, computes $b_a := c_a \oplus \mathsf{Dec}(key, \tilde{t}_a^f)$, and outputs $\top$ if $\mathsf{chal} = (\mathcal{I}, \mathsf{H}(b_a) \cdot \mathsf{pk}_O)$ holds; otherwise outputs $\bot$.

This audit addresses the threat of data tampering and substitution by malicious CSP, contributing to **G2 (Integrity)**.

*3) Consistency Audit:* After downloading the ciphertext $\mathcal{C}'$ from the CSP, the DO performs a consistency audit to detect any tampering or substitution:
- $\mathsf{chal} \leftarrow \boldsymbol{Audit}.\mathsf{Chal}^{(\lambda)}(ID, \mathcal{C})$: DO $ID$ computes $\mathsf{chal} := \mathsf{H}_2(\mathcal{C})$.
- $\mathsf{proof} \leftarrow \boldsymbol{Audit}.\mathsf{Proof}^{(\lambda)}(\mathcal{C})$: The CSP returns the $\mathcal{C}$ index stored in the cloud as $\mathsf{proof} := C_{in}$.
- $\{\top/\bot\} \leftarrow \boldsymbol{Audit}.\mathsf{Verf}(\mathsf{chal}, \mathsf{proof})$: Output $\top$ if $\mathsf{chal} = \mathsf{proof}$ holds; otherwise output $\bot$.

This audit addresses the threat of data tampering by malicious CSPs, contributing to **G2 (Integrity)**.

**Supplementary Remarks.** Although the consistency audit only compares public hashes, its effectiveness is enforced by system-wide mechanisms: $\mathcal{SC}$ penalizes CSPs that repeatedly fail audits (e.g., slashing deposits or terminating service agreements). The on-chain commitment $C_{in}$ immutably records the expected ciphertext hash; a CSP storing only the hash cannot pass subsequent **Continuous Integrity Audits**, which demand actual ciphertext blocks. Thus, hash-based consistency checks, random block audits, and blockchain-enforced penalties collectively incentivize CSPs to retain complete and correct data.

## VI. Security Analysis

In this section, we provide a comprehensive security analysis of MediChainAudit under the threat model defined in Section IV-B. We demonstrate that the system achieves its design goals **G1**–**G3** against all five adversary classes identified in the threat model: malicious CSP, external network adversary, Byzantine blockchain nodes, side-channel adversary, and network-level attackers. Concretely, **G1** (Confidentiality & Side-Channel Resilience) is proved via Theorems 1 and 2 and Section VI-C; **G2** (Data Integrity) is proved via Theorems 3–4; and **G3** (Auditability & Non-Repudiation) is proved via Theorem 5 and the immutable blockchain ledger. Resilience against network-level attacks is discussed in Section VI-E. All adversaries are modelled as probabilistic polynomial-time (PPT) entities that cannot break the underlying cryptographic primitives.

### A. Cryptographic Assumptions

Our security relies on the following standard cryptographic assumptions:
- **AES in CTR mode** is a secure Pseudo-Random Function (used in the FMLE proof) and provides IND-CPA security (used in the PDTP proof) [18, 47].
- The **EdDSA signature scheme** is Existentially Unforgeable under Chosen-Message Attacks (EUF-CMA) [50, 51].
- The cryptographic hash functions $\mathsf{H}$ and $\mathsf{H}_2$ are collision-resistant and can be modelled as Random Oracles (ROM) [36, 49]. Under the ROM, outputs on unseen inputs are unpredictable and uniformly distributed, which is used in the key-randomization and IV-randomization steps of the FMLE proof.
- The **Computational Diffie-Hellman (CDH)** problem is hard in the elliptic curve group used for ECDH key exchange [9, 22].
- The **blockchain** provides an immutable append-only ledger [20, 23].

A function $\epsilon(\lambda)$ is *negligible* in the security parameter $\lambda$ if for every positive polynomial $\mathsf{poly}(\cdot)$, $\epsilon(\lambda) < 1/\mathsf{poly}(\lambda)$ for all sufficiently large $\lambda$.

### B. Confidentiality (G1)

The confidentiality of our system hinges on the joint security of the FMLE encryption scheme and the PDTP transmission protocol. We formally define and prove the confidentiality property against both the malicious CSP and the external network adversary defined in the threat model.

**Theorem 1** (FMLE Confidentiality under PRV-CDA)**.** *The FMLE scheme is PRV-CDA secure with respect to the leakage function*

$$\mathcal{L}(F) = \big(n,\ |B_1|, \ldots, |B_n|,\ \mathcal{T}_f,\ \Delta\big),$$

*where $n$ and $|B_i|$ are the block count and sizes, $\mathcal{T}_f = (\mathsf{H}(\tilde{t}_1^f), \ldots, \mathsf{H}(\tilde{t}_n^f))$ is the public hashed tag set (under ROM revealing only the partition of equal raw tags, not the raw values), and $\Delta$ contains the pairwise XORs of plaintext blocks that share a raw tag inside the same file (cross-file XOR is prevented by file-specific keys). For any PPT adversary $\mathcal{A}$,*

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{prv\text{-}cda}}(\lambda) \le \mathsf{negl}(\lambda).$$

*Proof.* We prove the theorem through a sequence of four games. Let $\mathsf{Win}_i$ be the event that $\mathcal{A}$ distinguishes the ciphertexts of the two challenge files $F_0, F_1$ (satisfying $\mathcal{L}(F_0) = \mathcal{L}(F_1)$) in Game $i$.

1) **Game 0 (Real PRV-CDA game).** This is the original PRV-CDA game [18]. The challenger honestly runs FMLE.KeyGen and FMLE.Enc. By definition,
$$\Pr[\mathsf{Win}_0] = \tfrac{1}{2} + \mathsf{Adv}_{\mathcal{A}}^{\mathsf{prv\text{-}cda}}(\lambda).$$
2) **Game 1 (Random keystream).** For every block $b_i$, the AES-CTR keystream is replaced by a truly random string $R_i \xleftarrow{\$} \{0,1\}^{|b_i|}$. The encryption becomes $c_i \leftarrow R_i \oplus b_i$, with the constraint that if two blocks *within the same file* share the same raw fuzzy tag $\tilde{t}_i^f = \tilde{t}_j^f$ (and hence the same public tag $t_i^f = t_j^f$), then $R_i = R_j$. Any distinguisher between Game 1 and Game 0 yields a PRF adversary against AES-CTR. Hence
$$|\Pr[\mathsf{Win}_1] - \Pr[\mathsf{Win}_0]| \le \mathsf{Adv}_{\mathsf{AES\text{-}CTR}}^{\mathsf{prf}}(\lambda).$$

3) **Game 2 (Random encryption key per file).** The encryption key $key = \mathsf{H}_2(\tilde{t}_1^f \oplus \cdots \oplus \tilde{t}_n^f)$ is replaced by a fresh random key $key' \xleftarrow{\$} \{0,1\}^\lambda$ for each distinct file. The challenger maintains a table $T_{\text{key}}$ indexed by the canonically encoded public tag sequence $\mathsf{Encode}(\mathcal{T}_f)$ (or equivalently by the partition $\mathcal{P}$), guaranteeing consistent key reuse for identical files.

   Under the ROM, $\mathcal{A}$ can distinguish Game 2 from Game 1 only by querying $\mathsf{H}_2$ on the exact XOR aggregate $\oplus_{i=1}^n \tilde{t}_i^f$ of the challenge file. However, the leakage $\mathcal{L}$ provides only the public tags $t_i^f = \mathsf{H}_2(\tilde{t}_i^f)$, which reveal nothing about the raw tags $\tilde{t}_i^f$ under the ROM. The probability of a correct blind guess is therefore bounded by $q_{H_2} \cdot 2^{-\lambda}$, where $q_{H_2}$ is the number of RO queries. Hence

$$|\Pr[\mathsf{Win}_2] - \Pr[\mathsf{Win}_1]| \leq \frac{q_{H_2}}{2^\lambda}.$$

4) **Game 3 (Simulated ciphertext using leakage).** In this final game, the challenger no longer uses the plaintext blocks to produce ciphertexts; instead, it generates them solely from the leakage $\mathcal{L}(F_b)$ and fresh randomness. The simulation proceeds as follows:
   - For the challenge file $F_b$, extract the public tag set $\mathcal{T}_f$ and XOR relations $\Delta$ from $\mathcal{L}(F_b)$.
   - For each equivalence class $C \subseteq \{1, \ldots, n\}$ of blocks sharing the same public tag (i.e., same raw tag):
     - Choose a uniformly random string $c_{i_0} \xleftarrow{\$} \{0,1\}^{|B_{i_0}|}$ for a distinguished index $i_0 \in C$.
     - For every other index $j \in C$, compute $c_j \leftarrow c_{i_0} \oplus \Delta(i_0, j)$, where $\Delta(i_0, j) = b_{i_0} \oplus b_j$ is provided by the leakage.
   - For blocks with unique fuzzy tags (singleton classes), set $c_i \xleftarrow{\$} \{0,1\}^{|B_i|}$ uniformly at random.

   We argue that the distribution of ciphertexts in Game 3 is identical to that in Game 2. In Game 2, for an equivalence class $C$, the common keystream $R_C$ is uniformly random, so $c_{i_0} = R_C \oplus b_{i_0}$ is uniformly random. The relation $c_j = c_{i_0} \oplus (b_{i_0} \oplus b_j)$ holds deterministically. Since the simulator draws $c_{i_0}$ uniformly and reproduces the identical XOR relations, the two views are perfectly indistinguishable. Therefore,

$$\Pr[\mathsf{Win}_3] = \Pr[\mathsf{Win}_2].$$

In Game 3, the ciphertext is a deterministic function of the leakage $\mathcal{L}(F_b)$ and fresh randomness. For two files with $\mathcal{L}(F_0) = \mathcal{L}(F_1)$, the adversary's view is identically distributed, so $\Pr[\mathsf{Win}_3] = \frac{1}{2}$.

Summing the differences across all games:

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{prv\text{-}cda}}(\lambda) \leq \mathsf{Adv}_{\mathsf{AES\text{-}CTR}}^{\mathsf{prf}}(\lambda) + \frac{q_{H_2}}{2^\lambda}.$$

All terms are negligible under the stated cryptographic assumptions. This completes the proof. □

**Theorem 2** (PDTP Secrecy and Authenticity)**.** *The PDTP protocol securely transmits the* raw *fuzzy tag set $\tilde{\mathcal{T}}_f$ (required for decryption): for any PPT adversary $\mathcal{A}$ controlling the network, the advantage in distinguishing $\tilde{\mathcal{T}}_f$ from a random string, or in forging an authenticated message, is bounded by*

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{pdtp}}(\lambda) \leq \mathsf{Adv}_{\mathsf{AES\text{-}CTR}}^{\mathsf{ind\text{-}cpa}}(\lambda) + \mathsf{Adv}_{\mathbb{G}}^{\mathsf{cdh}}(\lambda) + \mathsf{Adv}_{\mathsf{EdDSA}}^{\mathsf{euf\text{-}cma}}(\lambda) + \frac{q_s}{2^\lambda},$$

*where $q_s$ is the number of sessions initiated by $\mathcal{A}$.*

*Proof (Hybrid Argument).* Let $\mathsf{Win}_i$ denote the event that $\mathcal{A}$ distinguishes the session key or forges a valid message in Hybrid $i$.

1) **Hybrid $H_0$: Real protocol.** This is the actual PDTP execution between honest parties.

$$\Pr[\mathsf{Win}_0] = \mathsf{Adv}_{\mathcal{A}}^{\mathsf{pdtp}}(\lambda).$$

2) **Hybrid $H_1$: Replace AES-CTR with random encryption.** In this hybrid, when the DO encrypts the tag set $\tilde{\mathcal{T}}_f$ (and the key confirmation), the AES-CTR output is replaced by a truly random string of the same length, while the same session key $EK$ is still used as input. Any difference between $H_0$ and $H_1$ directly yields an IND-CPA adversary against AES-CTR, because the session key is fresh and unknown to $\mathcal{A}$. Hence

$$|\Pr[\mathsf{Win}_1] - \Pr[\mathsf{Win}_0]| \leq \mathsf{Adv}_{\mathsf{AES\text{-}CTR}}^{\mathsf{ind\text{-}cpa}}(\lambda).$$

3) **Hybrid $H_2$: Replace DH shared secret with random.** Here we replace the shared secret $EK = \mathsf{H}_2(sk_O \cdot pk_C)$ (and the derived $ek$) by independent uniform random strings chosen by the challenger. If $\mathcal{A}$ can distinguish $H_2$ from $H_1$, it must break the CDH assumption on the elliptic curve, because computing $sk_O \cdot pk_C$ from the public keys and the ephemeral exchange is exactly the CDH problem. Therefore

$$|\Pr[\mathsf{Win}_2] - \Pr[\mathsf{Win}_1]| \leq \mathsf{Adv}_{\mathbb{G}}^{\mathsf{cdh}}(\lambda).$$

4) **Hybrid $H_3$: Replace EdDSA signatures with ideal ones.** In this final hybrid, all EdDSA signatures are replaced by an ideal signature oracle that always returns a valid signature for any message, but any forgery attempt on a message not queried to the oracle is detected and rejected. Any discrepancy between $H_3$ and $H_2$ implies a successful forgery against EdDSA, thus

$$|\Pr[\mathsf{Win}_3] - \Pr[\mathsf{Win}_2]| \leq \mathsf{Adv}_{\mathsf{EdDSA}}^{\mathsf{euf\text{-}cma}}(\lambda).$$

In $H_3$, the session key is random and independent of the protocol transcripts, the encryption carries no information, and all signatures are either honestly generated or instantly rejected. The adversary's view is therefore completely independent of the real $\tilde{\mathcal{T}}_f$. Consequently, $\mathcal{A}$ cannot distinguish the true tag set from a random string, nor can it create a valid message that would be accepted. Moreover, the probability that $\mathcal{A}$ succeeds by merely guessing the session key is at most $q_s \cdot 2^{-\lambda}$. Hence $\Pr[\mathsf{Win}_3] \leq q_s/2^\lambda$.

Summing the differences from all hybrids:

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{pdtp}}(\lambda) \leq \mathsf{Adv}_{\mathsf{AES\text{-}CTR}}^{\mathsf{ind\text{-}cpa}}(\lambda) + \mathsf{Adv}_{\mathbb{G}}^{\mathsf{cdh}}(\lambda) + \mathsf{Adv}_{\mathsf{EdDSA}}^{\mathsf{euf\text{-}cma}}(\lambda) + \frac{q_s}{2^\lambda},$$

which is negligible under the stated cryptographic assumptions. □

### C. Side-Channel Resistance Analysis (G1)

We now demonstrate that MediChainAudit satisfies Goal **G1** (Confidentiality & Side-Channel Resilience) under the threat model, covering the side-channel adversary and the external network adversary.

Recall from Theorem 1 that FMLE is PRV-CDA secure with respect to $\mathcal{L}(F) = (n, |B_i|, \mathcal{T}_f, \Delta)$, where $\mathcal{T}_f$ is the public (hashed) fuzzy tag set and $\Delta$ the file-internal XORs of blocks sharing a raw tag. Hence the adversary learns at most the file layout, the equality pattern of raw tags (which may reflect MinHash false positives/negatives), and the XORs within each tag class inside a single file, none of which is plaintext content.

1) **Defense against Frequency Analysis.** Assume an adversary $\mathcal{A}$ observes all on-chain data, stored ciphertexts, and the Cuckoo filter. We argue that the only information $\mathcal{A}$ can extract is exactly the leakage function $\mathcal{L}$ defined above.
   - **Fuzzy tag visibility.** The adversary observes the public hashed tags $\mathcal{T}_f = (\mathsf{H}(\tilde{t}_1^f), \ldots, \mathsf{H}(\tilde{t}_n^f))$. By the ROM, these hashes leak no information about the raw tags $\tilde{t}_i^f$ except their equality pattern (i.e., the partition $\mathcal{P}$). MinHash false positives and false negatives therefore influence the observed partition, but the adversary cannot compute the encryption key $key = \mathsf{H}_2(\oplus_i \tilde{t}_i^f)$ from $\mathcal{T}_f$ alone. Critically, the XOR relations in $\Delta$ are confined to blocks *within the same file*; across different files, the file-specific encryption key prevents any keystream reuse, even for blocks that share the same raw tag.
   - **Tag-distribution learning.** Even with full knowledge of the public tags $\mathcal{T}_f$, the adversary only learns which blocks share a tag, not the raw MinHash values. The raw MinHash function is many-to-one, so even if the adversary could guess a raw tag from its hash (which is infeasible under the ROM), the collision set remains large for practical parameters ($N = 64$, $t = 0.4$). Thus, the adversary's equivocation about the underlying plaintext blocks remains significant.
   - **Encrypted tag transmission.** During sharing, the *raw* fuzzy tags $\tilde{\mathcal{T}}_f$ (necessary for decryption) are transmitted inside the PDTP tunnel. Theorem 2 guarantees that an active network adversary learns nothing about $\tilde{\mathcal{T}}_f$ and cannot inject forged messages. Hence, the linkage between a specific user and the tags she uses remains hidden.
   - **Audit tags.** Audit tags $\tau_i = \mathsf{H}(c_i)$ are derived from ciphertext blocks. Under the ROM, they are *computationally* independent of the fuzzy tags given the ciphertexts. Observing the blockchain thus provides no additional information about $\mathcal{T}_f$ or the plaintext beyond what is already inherent in the ciphertexts, which are in turn bounded by $\mathcal{L}$.
   - **Ciphertext indistinguishability.** The PRV-CDA proof (Game 3) establishes that the ciphertexts are indistinguishable from a distribution depending solely on $\mathcal{L}$. Therefore, frequency analysis on stored ciphertexts provides no advantage beyond what is already explicitly leaked through $\mathcal{L}$.
2) **Defense against Brute-Force Attacks.** An adversary attempting to brute-force the decryption keys or the transmitted raw tags faces the following layered defenses, all formally guaranteed by PDTP (Theorem 2):
   - **Session key secrecy.** Under the ROM and the CDH assumption, the ECDH-derived session key $EK$ is indistinguishable from a random string to any adversary that does not solve CDH.
   - **Encryption.** The raw tag set $\tilde{\mathcal{T}}_f$ is encrypted with AES-CTR, which is IND-CPA secure; without $EK$, the ciphertext reveals no information.
   - **Authentication.** EdDSA signatures (EUF-CMA secure) prevent message tampering and impersonation, blocking the injection of chosen plaintexts for offline brute-force attacks.

   Thus, even an adversary with full network visibility and unbounded offline computation cannot efficiently recover $\tilde{\mathcal{T}}_f$ from the PDTP exchange beyond what is permitted by the computational assumptions.

In conclusion, MediChainAudit structurally limits information leakage to the precisely defined set $\mathcal{L}$: the file layout, the public hashed tag set $\mathcal{T}_f$ (which reveals only the equivalence class partition of the raw tags), and the file-internal XOR relations $\Delta$ for blocks sharing the same tag. Critically, no cross-file correlation exists due to the file-specific key derivation. Combined with the cryptographic shielding of PDTP (which protects the raw tags $\tilde{\mathcal{T}}_f$ during transmission), the system achieves **G1** against both the side-channel adversary and the external network adversary.

### D. Integrity (G2) and Auditability (G3)

**Theorem 3** (Integrity of Storage Audit (ISA))**.** *If* $\mathsf{H}$ *is a collision-resistant hash function, then the integrity audit protocol is sound. For a malicious CSP that modifies a fraction* $\alpha$ *of the* $n$ *blocks, the probability of passing an audit that checks* $m$ *random blocks is:*

$$\Pr[\textit{pass}] \leq (1-\alpha)^m + \mathsf{Adv}_{\mathsf{H}}^{\mathsf{coll}}(\lambda).$$

*Proof.* The CSP passes the audit in two mutually exclusive scenarios: If **no modified blocks are checked,** the probability that all $m$ randomly selected indices are unmodified is at most $(1-\alpha)^m$. If **a modified block is checked but the CSP provides a valid proof,** for a modified block $C_i' \neq C_i$, the CSP must produce a value that hashes to the original commitment $\tau_i = \mathsf{H}(C_i)$. This requires finding a hash collision, the probability of which is bounded by $\mathsf{Adv}_{\mathsf{H}}^{\mathsf{coll}}(\lambda)$. The overall probability is bounded by the union bound of these two events. □

**Theorem 4** (Soundness of Key Validity and Consistency Audits)**.** *The key validity and consistency audits are sound under the collision resistance of* $\mathsf{H}_2$ *and* $\mathsf{H}$*, respectively.*

- ***Key Validity:*** *Passing the audit with an invalid key set* $\mathcal{K}'$ *requires finding a preimage or collision in* $\mathsf{H}_2$ *such that* $\mathsf{H}_2(\mathcal{K}')$ *matches the on-chain commitment. Probability* $\leq \mathsf{Adv}_{\mathsf{H}_2}^{\mathsf{coll}}(\lambda)$.
- ***Consistency:*** *Passing the audit with a modified ciphertext* $\mathcal{C}' \neq \mathcal{C}$ *requires that* $\mathsf{H}_2(\mathcal{C}') = \mathsf{H}_2(\mathcal{C})$. *Probability* $\leq \mathsf{Adv}_{\mathsf{H}_2}^{\mathsf{coll}}(\lambda)$.

*Proof.* Directly follows from the collision resistance property of the hash functions. □

**Theorem 5** (Non-Repudiation and Accountability)**.** *The system provides non-repudiation and accountability if the blockchain is immutable and EdDSA is EUF-CMA secure. The probability that an adversary can deny an action or repudiate a transaction is bounded by:*

$$\Pr[\textit{Break}] \leq \mathsf{Adv}^{\mathsf{euf\text{-}cma}}_{\mathsf{EdDSA}}(\lambda) + \mathsf{Adv}^{\mathsf{coll}}_{\mathsf{BC}}(\lambda),$$

*where* $\mathsf{Adv}^{\mathsf{coll}}_{\mathsf{BC}}(\lambda)$ *is the probability of breaking the blockchain's immutability.*

*Proof.* A successful repudiation requires one of two conditions: 1) forging a signature on a transaction that was never signed by the user (breaking EUF-CMA), or 2) altering the immutable blockchain record of a valid transaction (breaking blockchain immutability). □

### *E. Resistance to Network-Level Attacks*

The threat model (Section IV-B) acknowledges standard network-level threats. We briefly discuss how MediChain-Audit withstands or can be extended to withstand them, addressing the blockchain node corruption and network-level attack classes.

1) **Blockchain Node Corruption.** The threat model (Section IV-B) assumes a PBFT consensus [53] tolerating up to $f$ Byzantine nodes among $3f+1$. Let $\mathcal{B}$ denote the event that more than $f$ validators are corrupted, causing the blockchain to lose its immutability property. Under $\neg\mathcal{B}$ (the regime assumed in Sections VI–VI-C), all security goals **G1–G3** hold in full as proved by Theorems 1, 3, and 5. Under $\mathcal{B}$:
   - **G3** (Non-Repudiation) degrades: on-chain records may be tampered with, so the blockchain alone no longer provides irrefutable evidence.
   - **G2** (Data Integrity) remains intact for any user who retains a local commitment (e.g., $C_{in}$ or $\mathcal{T}_a$), because the client-side consistency audit relies only on collision-resistant hashing, not on consensus.
   - **G1** (Confidentiality) is unaffected, as it derives solely from the cryptographic primitives FMLE and PDTP.

   Thus the system degrades gracefully: even under a consensus failure, confidentiality and detectable integrity are preserved; only the strength of non-repudiation is reduced.
2) **Sybil Attacks.** The permissioned blockchain employs a CA that binds each participating node to a verified real-world identity before issuing credentials [9]. Consequently, an adversary cannot create multiple fake identities to (i) subvert the PBFT consensus [53] (which relies on a fixed set of validators), or (ii) forge ownership claims in the deduplication audit [20], since every on-chain record is signed under a CA-certified key. The binding is cryptographically enforced by the EdDSA signature scheme (EUF-CMA secure), and any attempt to register a duplicate identity is detected by the CA.
3) **DDoS Attacks.** Let $\mathcal{S}_{\text{CSP}}$ and $\mathcal{S}_{\text{BC}}$ denote the service states of the CSP and the blockchain, respectively. The system architecture guarantees $\mathcal{S}_{\text{BC}} \not\Rightarrow \mathcal{S}_{\text{CSP}}$: the blockchain remains operational even when the CSP is under DDoS [54]. Hence the immutable audit log $\{Tx_{up}(C_{in})\}$ stays accessible for later verification of data integrity. On the CSP side, rate limiting bounds the number of upload and sharing requests per identity, and the integrity audit (a single challenge-response round, Theorem 3) is lightweight, minimizing the computational surface for amplification attacks. Service availability may degrade under a sustained DDoS, but stored ciphertext confidentiality and audit trail integrity are unaffected.
4) **Collusion.** Consider a coalition [56] of the CSP and a subset of users that attempts to (i) forge an integrity proof for missing data, or (ii) circumvent deduplication to claim storage savings without cross-user sharing. The integrity audit (Theorem 3) forces the CSP to answer $m$ random challenges; if a fraction $\alpha$ of blocks are missing, the coalition succeeds with probability at most $(1-\alpha)^m$ plus a negligible hash-collision term. Thus, even a fully colluding CSP must store the complete ciphertext to avoid detection. Stronger guarantees (e.g., distributing the audit across $k$ independent verifiers to drive the success probability below $(1-\alpha)^{mk}$) can be achieved without altering the core protocol; we leave such extensions to future work. Our parallel work PriME-Deal [59] provides policy-hiding fair exchange, which would further reduce the incentive for collusion in data sharing.
5) **Compliance Auditing Prior to Encryption.** MediChain-Audit targets secure deduplication and storage auditing; it does not inspect plaintext content. Medical regulations that require content-level compliance checking before encryption can be addressed by an orthogonal cryptographic layer: the data owner produces a zero-knowledge proof $\pi$ (e.g., a zk-SNARK) attesting that the plaintext satisfies a public compliance policy, and the blockchain verifies $\pi$ before recording the upload transaction. This preserves the privacy guarantees of FMLE and PDTP while satisfying regulatory requirements. Furthermore, the system's architecture naturally aligns with data localisation requirements (e.g., General Data Protection Regulation (GDPR), China's Data Security Law): raw encrypted data resides at the source institution's CSP, while only compact audit tags and anonymous keys are recorded on the blockchain; cross-border sharing is thus limited to non-identifiable metadata, substantially reducing the regulatory surface. We leave the concrete design of such a compliance proof system to future work.

In summary, we formally prove that MediChainAudit attains its security objectives: **confidentiality and side-channel resilience (G1)** are guaranteed by PRV-CDA-secure FMLE (Theorem 1), the authenticated key exchange of PDTP (Theorem 2), and the side-channel analysis of Section VI-C; **data integrity (G2)** is ensured through collision-resistant audit protocols (deduplication, integrity, consistency, and key validity audits, Theorems 3–4); and **auditability with non-repudiation (G3)** is provided by blockchain-anchored, tamper-proof logging of all operations (Theorem 5). Resilience against network-level and compliance threats is discussed in Section VI-E. The system remains secure against all adversary classes defined in the threat model.

## VII. Performance Evaluation

This section evaluates the performance of MediChainAudit to answer the following research questions (RQs):

**RQ1:** How does the proposed FMLE scheme compare to standard MLE in terms of storage and time efficiency on medical data?

**RQ2:** What is the computational and storage overhead of the on-chain smart contracts governing the data lifecycle?

**RQ3:** How effective and efficient is the deduplication audit mechanism in detecting data redundancy under different duplication rates?

We conduct experiments from two perspectives: local off-chain cryptographic operations (FMLE encryption and deduplication audit) and cloud-based on-chain smart contract execution. The local experiments were performed on a desktop equipped with an Intel i7-11700 CPU (2.50GHz) and 32GB RAM, running Ubuntu 20.04. The cloud experiments were conducted on a server featuring an Intel Xeon Gold 6226R CPU (2.90GHz) with 60 cores and 180GB of RAM, also running Ubuntu 20.04. The experimental data encompass the publicly available real medical dataset MIMIC-III (Medical Information Mart for Intensive Care III) [25] and synthetic datasets (100 patients [26] and 30K patients [27], generated by Synthetic), as well as other non-public test datasets. Our scheme is publicly available at: https://github.com/JackAugust/MediChainAudit.git. Note that the blockchain smart contract components are excluded due to commercial deployment constraints.

### A. Theoretical Analysis and Comparison

To position MediChainAudit in the literature, we compare its protocol-level costs with two recent fuzzy deduplication schemes: Jiang et al. [36] and Tang et al. [42]. Both baselines are file-level schemes that use different algorithmic foundations (secure sketch with oblivious transfer, and perceptual hashing with public-key cryptography, respectively) and do not incorporate blockchain-based auditing. To enable a fair comparison, we decompose each scheme into the same four-stage lifecycle as our smart contracts: **Upload/Setup**, **Share/Dedup**, **Access/Download**, and **Audit**. We then count the dominant cryptographic operations at each stage using a *unified primitive set*: H (one cryptographic hash operation, e.g., SHA-256, MinHash, perceptual hash), E (one symmetric encryption or decryption, e.g., AES), P (one public-key operation, e.g., pairing evaluation or PKE encrypt/decrypt), and M (one modular multiplication or addition over a large finite field). For Jiang et al. [36], the secure sketch and fuzzy extractor are dominated by hash evaluations and counted as one composite H; the $N$ oblivious transfers in FuzzyPoW each require the sender to encrypt one bit and the receiver to perform one modular addition, hence $N{\cdot}\mathsf{E} + N{\cdot}\mathsf{M}$. For Tang et al. [42], each public-key encryption or decryption counts as one P; the zero-knowledge proof is conservatively counted as two P (prover + verifier); Hamming distance and Pearson correlation are constant-time arithmetic and omitted. The detailed step-by-step mapping is available in our open-source repository.

Table II presents the resulting operation counts per stage. The storage and communication rows apply solely to our scheme because the baselines do not employ blockchain-based audit layers. All user-initiated stages require an EdDSA signature for the blockchain transaction ($\approx 2\mathsf{H} + 1\mathsf{M}$). In the **Share/Dedup** stage, the user only signs the deduplication request; the fuzzy tags $\mathcal{T}_f$ used for similarity detection are already computed during the **Upload** stage and are not regenerated.

TABLE II
Theoretical cost comparison per protocol stage and storage/communication overhead of MediChainAudit. $n$ = number of 4 KB blocks (ours) or ciphertext bit-length (Jiang et al.); $|\lambda|$ = security parameter length; $|G|$ = group element size. Storage and communication rows are shown only for our scheme; the baselines do not include on-chain auditing.

| | **Computational Cost (per stage)** | | |
|---|---|---|---|
| **Stage** | **MediChainAudit** | **Jiang et al. [36]** | **Tang et al. [42]** |
| Upload / Setup | $(2n+4)\mathsf{H} + n\mathsf{E}$ | $4\mathsf{H} + 1\mathsf{E}$ | $3\mathsf{H} + 4\mathsf{P}$ |
| Share / Dedup | $2\mathsf{H} + 1\mathsf{M}$ | $N{\cdot}\mathsf{E} + N{\cdot}\mathsf{M}$ | $8\mathsf{P} + 1\mathsf{H}$ |
| Access / Download | $n\mathsf{E} + 6\mathsf{H}$ | $1\mathsf{E}$ | $1\mathsf{E}$ |
| Audit | $1\mathsf{H}$ (on-chain) | — | — |
| | **Storage Overhead (ours)** | | |
| **Contract** | **Local (User)** | **CSP** | **Blockchain** |
| $\boldsymbol{Upload}(\cdot)$ | $(n+2)\|\lambda\|$ | $n{\cdot}4\text{KB} + 2\|\lambda\|$ | $(2n+7)\|\lambda\| + \|G\|$ |
| $\boldsymbol{Share}(\cdot)$ | $(n+2)\|\lambda\|$ | $n{\cdot}4\text{KB} + 2\|\lambda\|$ | $(2n+7)\|\lambda\| + \|G\|$ |
| $\boldsymbol{Access}(\cdot)$ | $n\|\lambda\|$ | $n{\cdot}4\text{KB} + 2\|\lambda\|$ | $5\|\lambda\| + \|G\|$ |
| $\boldsymbol{Audit}(\cdot)$ | — | — | $\|\lambda\|$ (per proof) |
| | **Communication Overhead (ours)** | | |
| $\boldsymbol{Upload}(\cdot)$ | $n{\cdot}4\text{KB} + 2(n{+}1)\|\lambda\|$ | $2\|\lambda\|$ | 1bit (Tx) |
| $\boldsymbol{Share}(\cdot)$ | $4\|\lambda\|$ | — | 1bit (Tx) |
| $\boldsymbol{Access}(\cdot)$ | $7\|\lambda\| + 2\|G\|$ | $3\|G\| + \|\lambda\|$ | 1bit (Tx) |
| $\boldsymbol{Audit}(\cdot)$ | $2\|\lambda\|$ (initial) | $2\|\lambda\|$ (initial) | 1bit (Tx) |

*1) Key observations:* The table highlights three fundamental differences. First, in **Upload/Setup** all schemes incur comparable hash and symmetric costs; our additional $n\mathsf{E}$ operations are the price for block-level fuzzy deduplication. Second, **Share/Dedup** reveals the dominant bottlenecks: Jiang et al. [36] suffer $O(N)$ oblivious transfers (linear in file size), while Tang et al. [42] pay a constant but heavy $8\mathsf{P}$ per attempt. Our scheme avoids both extremes by offloading deduplication to the CSP and requiring only an EdDSA signature from the user. Third, **Audit** is unique to our design, a single on-chain hash ($1\mathsf{H}$) delivers publicly verifiable accountability, a qualitative advantage absent in both baselines. The lower part of the table further shows that our on-chain storage and communication overheads are compact: the bulk of data resides at the CSP, while subsequent operations (share, access, audit) cost only a few hundred bytes and one blockchain transaction.

*2) Cost illustration based on measured micro-benchmarks:* To provide intuition for the asymptotic differences in Table II, we instantiate the unified primitives with concrete timings measured on our local testbed (Intel i7-11700, 2.50 GHz, Ubuntu 20.04).

- H (SHA-256 or MinHash): $\approx 0.05$ ms
- E (AES-256-CTR on 4 KB): $\approx 0.02$ ms
- M (modular multiplication): $< 0.001$ ms

For public-key operations, we use the BN254 curve (Type-3 asymmetric pairing [60]) as a representative 128-bit security level. Although our own scheme employs EdDSA (not pair-

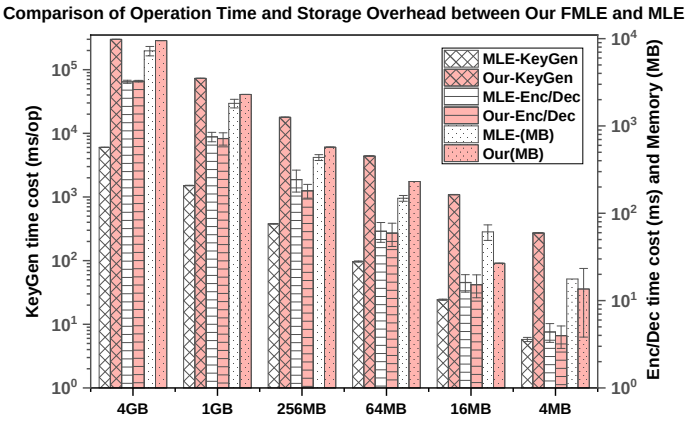


Fig. 2. Performance comparison between MLE and FMLE.

TABLE III
PERFORMANCE COMPARISON: MLE VS. FMLE ON SYNTHETIC MEDICAL DATA (100 PATIENTS [26] AND 30K PATIENTS [27]) AND COMPREHENSIVE CLINICAL DATA [25]

| Scheme | Original Size | Deduplicated Size | Size Reduction (%) | Compression Ratio | Ave. Processing Time (s) |
|---|---|---|---|---|---|
| MLE | 39.1MB in [26] | 6.37MB ± 6.58 | 90.02 ± 7.35 | 16.82 ± 16.26 | 0.131 ± 0.233 |
| FMLE | | **1.48MB ± 3.28** | **97.61 ± 5.24** | **68.49 ± 111.73** | 0.259 ± 0.416 |
| MLE | 6.35GB in [27] | 2.87GB ± 2.05 | 83.75 ± 8.00 | 8.15 ± 5.31 | 0.095 ± 0.052 |
| FMLE | | **0.36GB ±0.38** | **95.53 ± 7.41** | **42.63 ± 59.02** | 0.181 ± 0.084 |
| MLE | 43.3GB in [25] | 12.57GB ± 13.20 | 84.04 ± 8.59 | 8.90 ± 7.03 | 0.264 ± 0.443 |
| FMLE | | **6.88GB ±8.52** | **99.90 ± 1.10** | **16.14 ± 117.51** | 0.268 ± 0.398 |

ings), the P counts in Tang et al. [42] are dominated by pairing evaluations; we therefore adopt the measured $e(\mathbb{G}_1, \mathbb{G}_2)$ time on BN254 ($\approx$ 15.3 ms) and the $\mathbb{G}$ exponentiation time ($\approx$ 0.57 ms) as concrete references. These numbers serve solely for comparative illustration and do not constitute a claim that our scheme requires pairings.

Under these estimates, the Share/Dedup stage for a 64 MB file ($N \approx 5{\times}10^8$ bits) would cost Jiang et al. [36] roughly $N{\cdot}(0.02 + 0.001) \approx 1.05{\times}10^7$ ms ($\approx$ 2.9 hours) due to the linear OT bottleneck, whereas Tang et al. [42] incur $8{\times}15.3 = 122.4$ ms in pairing operations alone, and our scheme requires only $2{\times}0.05{+}0.001 \approx 0.1$ ms for the EdDSA signature. This stark contrast illustrates why the asymptotic differences in Table II translate into qualitatively different deployment profiles.

### B. Off-chain Performance

*1) Raw Performance Comparison:* To answer **RQ1**, we implemented MLE with AES-CTR and SHA-256 as a baseline [18, 20] and our proposed FMLE based on MinHash for similarity-aware key derivation. For the MinHash-LSH used in FMLE, we instantiate $N = 64$ hash functions and set the Jaccard similarity threshold to $t = 0.4$ following medical data deduplication practice [42]. The band parameters $(k, l)$ are optimized to minimize false positive and false negative probabilities under the constraint $k \cdot l \leq N$, resulting in $k = 8$, $l = 8$. The corresponding false positive probability is below $2.3 \times 10^{-5}$, and the false negative probability is approximately 0.12. These probabilities affect only the efficiency and precision of deduplication; they do not impact the cryptographic security of FMLE, whose proof (Theorem 1) explicitly includes the actual tag multiset $\mathcal{T}_f$ within the leakage function $\mathcal{L}$. We evaluated their performance on synthetically generated datasets (4MB to 4GB) with a fixed 10% duplication rate, measuring the average time cost in key generation (KeyGen), encryption and decryption (Enc/Dec), as well as memory usage.

As shown in Fig. 2, our FMLE scheme (red part) has a significantly higher cost, dominated by the **key generation** phase (MinHash computation), which constitutes over 99% of the total time for large files (e.g., 4m59s for 4GB). This is the price of enabling fuzzy deduplication. However, the actual **encryption and decryption times** for FMLE are comparable to MLE (e.g., 3.25s vs. 2.21s for encrypting and decrypting 4GB). This indicates that the symmetric encryption itself is not the bottleneck. Considering that the key generation cost might seem prohibitive, the Section VII-B2 will show that the high key generation cost of FMLE is justified by its superior deduplication capability.

*2) Performance Comparison on Public Medical Dataset:* We evaluated the performance of MLE and FMLE on synthetic medical datasets of different scales: a smaller cohort of 100 patient EHRs [26] and a larger cohort of 30,000 patient EHRs [27], both generated by Synthea. These datasets exhibit the high semantic similarity characteristic of real-world medical records. To further validate the effectiveness of FMLE on real clinical data, we introduced the widely used real-world medical dataset MIMIC-III [25]. This database contains de-identified health-related data from over 40,000 patients admitted to the intensive care units of Beth Israel Deaconess Medical Center between 2001 and 2012 [4]. Characterized by its large scale, public availability, and rich clinical information, it provides a data foundation for method evaluation that closely mirrors real-world application scenarios.

As shown in Table III, FMLE consistently demonstrates superior deduplication efficacy across both scales. On the 100-patient dataset, FMLE achieved a mean size reduction of **97.61%**, significantly outperforming MLE's **90.02%** and resulting in a **4.1x** higher compression ratio (68.49 vs. 16.82). This performance advantage is even more pronounced on the large-scale 30K-patient dataset, where FMLE achieved a **95.53%** size reduction compared to MLE's **83.75%**, representing a **5.2x** improvement in compression ratio (42.63 vs. 8.15). On the real-world clinical database, FMLE again exhibits outstanding performance, achieving a size reduction of **99.90%**, substantially higher than MLE's **84.04%**, with a compression ratio increased by **1.8**× (16.14 vs. 8.90). These results further confirm the robustness and efficiency of FMLE on real, complex medical data.

This storage efficiency, enabled by FMLE's ability to detect semantically similar blocks, incurs a predictable computational cost. On synthetic datasets, FMLE requires approximately **1.98x** (small) and **1.91x** (large) more processing time than

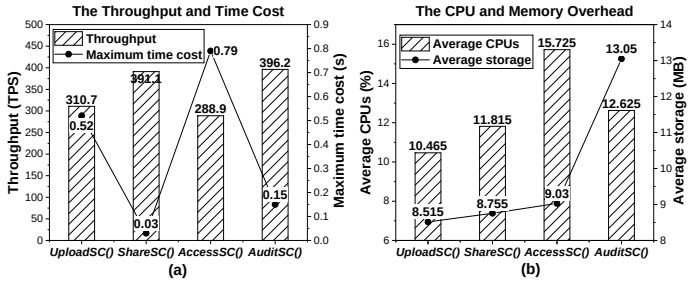


Fig. 3. Performance Analysis of Our Smart Contract (Fig. a shows Throughput and Latency, Fig. b shows CPU and Memory Consumption of Peer Nodes)

TABLE IV
DEDUPLICATION AUDIT RESULTS ON SYNTHETIC DATA WITH VARYING DUPLICATION RATES

| Size | $\delta$ | Total Time (ms) | Key Gen. (ms) | Enc. (ms) | Memory (MB) | Dup. Count | Stop here |
|---|---|---|---|---|---|---|---|
| 64 MB | 0.1 | 4673 | 4502 | 171 | 99.59 | 1639 | **14745** |
| | 0.2 | 4688 | 4525 | 162 | 85.11 | 3277 | **13107** |
| | 0.3 | 4705 | 4550 | 155 | 85.81 | 4916 | **11468** |
| | 0.4 | 4767 | 4615 | 150 | 88.33 | 6554 | **9830** |
| | 0.5 | 4749 | 4610 | 138 | 114.63 | 8192 | **8977** |
| 128 MB | 0.1 | 10314 | 9973 | 341 | 223.02 | 3277 | **29491** |
| | 0.2 | 10308 | 9984 | 323 | 202.69 | 6554 | **26214** |
| | 0.3 | 10284 | 9971 | 313 | 226.05 | 9831 | **22937** |
| | 0.4 | 10134 | 9843 | 290 | 223.75 | 13108 | **19660** |
| | 0.5 | 10143 | 9870 | 272 | 182.56 | 16384 | **17948** |

MLE due to its similarity-aware hashing. Notably, on the real-world MIMIC-III dataset, both methods show comparable processing times (0.268s vs. 0.264s). Given the substantial storage savings achieved—especially valuable for large-scale medical archives—this trade-off remains highly favorable.

### C. On-chain Performance

To answer **RQ2**, we analyze the resource consumption of our smart contracts deployed on a permissioned Hyperledger Fabric network (v2.2)[4], where validator peers are authenticated by PKI and elected from trusted member nodes, and use the practical BFT consensus protocol. We developed the contracts in Go 1.18.8 and deployed them across three Docker-based peer nodes. Performance metrics were gathered using Caliper v0.4.0[5].

Fig. 3 presents the empirical results. $\boldsymbol{Upload}(\cdot)$ and $\boldsymbol{Access}(\cdot)$ contracts exhibit higher latency and lower throughput (Fig. 3a) due to data transfer and complex cryptographic operations (e.g., PDTP's authenticated key exchange). $\boldsymbol{Share}(\cdot)$ and $\boldsymbol{Audit}(\cdot)$ show higher throughput as they involve minimal data transfer. Average CPU and memory consumption of peer nodes (Fig. 3b) align with theoretical predictions. $\boldsymbol{Access}(\cdot)$ incurs the highest CPU load due to digital signature generation and verification within the PDTP protocol. Storage overhead is highest for $\boldsymbol{Audit}(\cdot)$, stemming from maintaining continuous reading and writing of records on the chain. Overall, the on-chain performance is efficient and practical for managing medical data audit trails. The resource consumption is rational and stable across nodes.

### D. Effectiveness of Deduplication Audit

To answer **RQ3**, we evaluate the effectiveness and efficiency of the deduplication audit mechanism described in Section V-A1. We generated synthetic 64MB and 128MB datasets (common sizes for medical records like MRIs or genomic data [4]) with controlled duplication rates ($\delta = 0.1$ to $0.5$) and executed the $\boldsymbol{Audit}(\cdot)$ contract. Table IV shows the results. The audit successfully terminates early once the number of detected duplicate blocks exceeds the threshold ($0.1$), demonstrating efficiency. For instance, with $\delta = 0.3$, the audit for the 64MB dataset stopped after checking only 11,468 out of 16,384 blocks (bolded), having found 1,639 duplicates. This early termination leads to a significant reduction in audit time (1014.13ms vs. 4538.78ms for full audit at $\delta = 0.1$). A known minor false positive rate of the maintained Cuckoo filter [58] causes the detected duplicates to occasionally slightly exceed the expected value ($\delta \times$ total blocks). However, by adjusting the threshold $\delta$, the system can achieve a desired balance between audit accuracy and performance. The results confirm that the mechanism is highly effective and efficient for identifying redundant data in storage. More importantly, this optimization makes the performance penalty manageable for the target scenario. For a 128MB file with $\delta = 0.1$, FMLE with early termination takes approximately 10.314s total, which is acceptable for the batch upload processes common in healthcare systems, given the enormous storage savings it enables (as shown in Table IV).

This demonstrates that the audit mechanism is highly effective and efficient for its intended purpose of encrypting highly redundant medical data, successfully answering **RQ3**.

### E. Summary

The experimental results provide positive answers to our research questions:

Answer to **RQ1**: FMLE significantly outperforms MLE in storage efficiency on medical data (such as 99.90% vs. 84.04% size reduction) with a reasonable computational overhead, making it practical for medical archives.

Answer to **RQ2**: The on-chain smart contracts introduce manageable and rational overhead. Complex protocols like $\boldsymbol{Access}(\cdot)$ (PDTP) incur higher but justified latency due to strong security guarantees.

Answer to **RQ3**: The deduplication audit mechanism is effective and efficient, enabling early termination upon detecting sufficient duplicates, which drastically reduces audit time, especially for highly redundant data.

Overall, MediChainAudit achieves its design goals: confidential storage and sharing with side-channel resilience (**G1**), detectable data integrity (**G2**), auditable and non-repudiable operations (**G3**), and practical storage and computational efficiency (**G4**). The system is feasible for real-world deployment in healthcare environments.

## VIII. CONCLUSION AND FUTURE WORK

This paper formalizes the Medical Secure Deduplication Auditing problem and presents MediChainAudit, the first

[4]https://github.com/hyperledger/fabric.git
[5]https://github.com/hyperledger/caliper.git

system to reconcile encrypted fuzzy deduplication with verifiable audit in collaborative healthcare. Our Fuzzy Message-Locked Encryption achieves near-optimal storage reduction by exploiting semantic similarity; the Privacy-preserving Data Transmission Protocol eliminates side-channel risks during ciphertext sharing; a blockchain-anchored audit layer provides tamper-proof, non-repudiable logs for all operations. Security proofs and extensive experiments on real clinical datasets demonstrate that MediChainAudit delivers strong security guarantees with practical overhead, enabling privacy-preserving medical research at scale.

**Future Work.** A direct extension is medical imaging (MRI/CT), where massive volumes and high redundancy call for tailored similarity metrics and privacy analysis. We are also planning a large-scale pilot with clinical partners to incorporate compliance features (e.g., consent revocation logging, automated retention timers), optimize blockchain deployment costs, and adopt mainstream blockchain frameworks for broader reproducibility and real-world usability. To further strengthen resilience against the network-level threats, we intend to explore distributed audit verification across multiple independent verifiers, which would exponentially reduce the success probability of collusion attacks, and to integrate zero-knowledge compliance proofs as a pre-encryption step, enabling regulatory content auditing without compromising the privacy guarantees of FMLE and PDTP.

## REFERENCES


[1] J. D. Wall, E. W. Stawiski, A. Ratan *et al.*, "The genomeasia 100k project enables genetic discoveries across asia," *Nature*, vol. 576, no. 7785, pp. 106–111, Dec 2019.

[2] Y. Lv, X. Li, Y. Wang *et al.*, "Cross-chain sharing of personal health records: Heterogeneous and interoperable blockchains," in *IEEE BIBM, Lisbon, Portugal, December 3-6, 2024*. IEEE, 2024, pp. 3588–3591.

[3] M. Zhang, X. Li, J. Zhang *et al.*, "Swiftguard: Enhanced privacy and efficiency in blockchain-based fine-grained access control for cross-domain healthcare collaboration," in *28th CSCWD, Compiegne, France, May 5-7, 2025*. IEEE, 2025, pp. 1863–1868.

[4] A. E. Johnson, T. J. Pollard, L. Shen *et al.*, "Mimic-iii, a freely accessible critical care database," *Scientific data*, vol. 3, no. 1, pp. 1–9, 2016.

[5] H. Mao, J. Zhang, Y. Zhang *et al.*, "A cross-domain data sharing scheme based on federated blockchain," in *19th TASE, Limassol, Cyprus, July 14-16, 2025*, vol. 15841. Springer, 2025, pp. 285–302.

[6] A. S. Kesselheim and M. M. Mello, "Confidentiality laws and secrecy in medical research: improving public access to data on drug safety," *Health Affairs*, vol. 26, no. 2, pp. 483–491, 2007.

[7] W. M. Association *et al.*, "World medical association declaration of helsinki: ethical principles for medical research involving human participants," *Jama*, vol. 333, no. 1, pp. 71–74, 2025.

[8] M. Song, Z. Hua, Y. Zheng *et al.*, "Blockchain-based deduplication and integrity auditing over encrypted cloud storage," *IEEE Trans. Dependable Secur. Comput.*, vol. 20, no. 6, pp. 4928–4945, 2023.

[9] J. Zhang, X. Li, R. Feng *et al.*, "From isolation to integration: A reputation-backed auditable model for cohort data sharing," *IEEE Trans. Dependable Secur. Comput.*, vol. 23, no. 1, pp. 637–654, 2026.

[10] K. Vivekrabinson, K. Ragavan, P. J. Thilaga *et al.*, "Secure cloud-based electronic health records: Cross-patient block-level deduplication with blockchain auditing," *J. Medical Syst.*, vol. 48, no. 1, p. 33, 2024.

[11] L. Xiao, B. Zou, C. Zhu *et al.*, "Esdedup: An efficient and secure deduplication scheme based on data similarity and blockchain for cloud-assisted medical storage systems," *J. Supercomput.*, vol. 79, no. 3, pp. 2932–2960, 2023.

[12] Y. Lv, X. Li, K. Chen *et al.*, "Medexchain: Enabling secure and efficient PHR sharing across heterogeneous blockchains," *IEEE Internet Things J.*, vol. 12, no. 16, pp. 32 932–32 950, 2025.

[13] F. A. Satti, M. Hussain, S. I. Ali *et al.*, "A semantic sequence similarity based approach for extracting medical entities from clinical conversations," *Inf. Process. Manag.*, vol. 60, no. 2, p. 103213, 2023.

[14] Y. Zhang, C. Xu, H. Li *et al.*, "Healthdep: An efficient and secure deduplication scheme for cloud-assisted ehealth systems," *IEEE Trans. Ind. Informatics*, vol. 14, no. 9, pp. 4101–4112, 2018.

[15] A. Azaria, A. Ekblaw, T. Vieira *et al.*, "Medrec: Using blockchain for medical data access and permission management," in *2nd OBD, Vienna, Austria, August 22-24, 2016*, I. Awan and M. Younas, Eds. IEEE Computer Society, 2016, pp. 25–30.

[16] Y. Fu, N. Xiao, T. Chen *et al.*, "Fog-to-multicloud cooperative ehealth data management with application-aware secure deduplication," *IEEE Trans. Dependable Secur. Comput.*, vol. 19, no. 5, pp. 3136–3148, 2022.

[17] Q. Zhang, S. Qian, J. Cui *et al.*, "Blockchain-based privacy-preserving deduplication and integrity auditing in cloud storage," *IEEE Trans. Computers*, vol. 74, no. 5, pp. 1717–1729, 2025.

[18] M. Bellare, S. Keelveedhi, and T. Ristenpart, "Message-locked encryption and secure deduplication," in *32nd EUROCRYPT, Athens, Greece, May 26-30, 2013.*, vol. 7881. Springer, 2013, pp. 296–312.

[19] R. Chen, Y. Mu, G. Yang *et al.*, "BL-MLE: block-level message-locked encryption for secure large file deduplication," *IEEE Trans. Inf. Forensics Secur.*, vol. 10, no. 12, pp. 2643–2652, 2015.

[20] G. Tian, Y. Hu, J. Wei *et al.*, "Blockchain-based secure deduplication and shared auditing in decentralized storage," *IEEE Trans. Dependable Secur. Comput.*, vol. 19, no. 6, pp. 3941–3954, 2022.

[21] S. Jiang, J. Liu, Y. Zhou *et al.*, "Fvc-dedup: A secure report deduplication scheme in a fog-assisted vehicular crowdsensing system," *IEEE Trans. Dependable Secur. Comput.*, vol. 19, no. 4, pp. 2727–2740, 2022.

[22] Y. Miao, K. Gai, L. Zhu *et al.*, "Blockchain-based shared data integrity auditing and deduplication," *IEEE Trans. Dependable Secur. Comput.*, vol. 21, no. 4, pp. 3688–3703, 2024.

[23] S. Li, C. Xu, Y. Zhang *et al.*, "Blockchain-based transparent integrity auditing and encrypted deduplication for cloud storage," *IEEE Trans. Serv. Comput.*, vol. 16, no. 1, pp. 134–146, 2023.

[24] J. Zhao, Z. Yang, J. Li *et al.*, "Encrypted data reduction: Removing redundancy from encrypted data in outsourced storage," *ACM Trans. Storage*, vol. 20, no. 4, pp. 27:1–27:30, 2024.

[25] A. Johnson, T. Pollard, and R. Mark, "MIMIC-III Clinical Database," *PhysioNet*, Sep. 2016, version 1.4. [Online]. Available: https://doi.org/10.13026/C2XW26

[26] A. Chen, "Medical records of 30K Synthea synthetic patients," 2022. [Online]. Available: https://doi.org/10.7910/DVN/BWDKXS

[27] A. J. Chen, "Medical records of 100 synthea patients," 2022. [Online]. Available: https://doi.org/10.7910/DVN/VBEKZO

[28] Y. Gao, L. Chen, J. Han *et al.*, "Similarity-based secure deduplication for iiot cloud management system," *IEEE Trans. Dependable Secur. Comput.*, vol. 21, no. 4, pp. 42–56, 2024.

[29] C. Gan, Y. Hu, L. Zhao *et al.*, "Enabling encrypted delta compression for outsourced storage systems via preserving similarity," in *41st ICCD, Washington, DC, USA, November 6-8, 2023*. IEEE, 2023, pp. 231–238.

[30] J. Li, P. P. C. Lee, C. Tan *et al.*, "Information leakage in encrypted deduplication via frequency analysis: Attacks and defenses," *ACM Trans. Storage*, vol. 16, no. 1, pp. 4:1–4:30, 2020.

[31] Y. Zhang, Y. Mao, M. Xu *et al.*, "Towards thwarting template side-channel attacks in secure cloud deduplications," *IEEE Trans. Dependable Secur. Comput.*, vol. 18, no. 3, pp. 1008–1018, 2021.

[32] Z. Yang, J. Li, Y. Ren *et al.*, "Tunable encrypted deduplication with attack-resilient key management," *ACM Trans. Storage*, vol. 18, no. 4, pp. 32:1–32:38, 2022.

[33] Z. Tang, S. Zeng, S. Han *et al.*, "Hdefc: Hierarchical secure fuzzy deduplication based on fog computing," *IEEE Internet Things J.*, vol. 12, no. 9, pp. 12 658–12 668, 2025.

[34] J. Zhang, X. Li, M. Zhang *et al.*, "QAE-BAC: achieving quantifiable anonymity and efficiency in blockchain-based access control with attribute," *IEEE Internet Things J.*, pp. 1–12, 2026.

[35] G. Zhang, P. Chen, and Y. Liao, "Blockchain-based secure and verifiable deduplication scheme for cloud-assisted internet of things," *IEEE Internet Things J.*, vol. 11, no. 8, pp. 13 995–14 006, 2024.

[36] T. Jiang, X. Yuan, Y. Chen *et al.*, "Fuzzydedup: Secure fuzzy deduplication for cloud storage," *IEEE Trans. Dependable Secur. Comput.*, vol. 20, no. 3, pp. 2466–2483, 2023.

[37] Y. Lin, J. Li, S. Kimura *et al.*, "Blockchain based public auditing outsourcing for cloud storage," in *27th ICPADS, China, Dec 14-16, 2021*. IEEE, 2021, pp. 482–489.

[38] M. Wang, Y. Guo, C. Zhang *et al.*, "Medshare: A privacy-preserving medical data sharing system by using blockchain," *IEEE Trans. Serv. Comput.*, vol. 16, no. 1, pp. 438–451, 2023.

[39] Y. Miao, K. Gai, Y. Tan *et al.*, "Blockchain-assisted searchable integrity auditing for large-scale similarity data with arbitration," *IEEE Trans. Dependable Secur. Comput.*, vol. 22, no. 6, pp. 6012–6027, 2025.

[40] C. Zhang, Y. Miao, Q. Xie *et al.*, "Privacy-preserving deduplication of sensor compressed data in distributed fog computing," *IEEE Trans. Parallel Distributed Syst.*, vol. 33, no. 10, pp. 4176–4191, 2022.
[41] S. Cheng, Z. Tang, S. Zeng *et al.*, "Pfdup: Practical fuzzy deduplication for encrypted multimedia data," *J. Ind. Inf. Integr.*, p. 100613, 2024.
[42] Z. Tang, S. Zeng, S. Han *et al.*, "Fuzzy deduplication: Color-aware deduplication for multi-media data," *IEEE Trans. Serv. Comput.*, vol. 17, no. 5, pp. 2459–2472, 2024.
[43] J. Zhang, X. Li, M. Zheng *et al.*, "Verifuzzy: A dynamic verifiable fuzzy search service framework for encrypted cloud data," *IEEE Trans. Serv. Comput.*, vol. 19, no. 1, pp. 780–793, 2026.
[44] F. Armknecht, J. Bohli, G. O. Karame *et al.*, "Outsourced proofs of retrievability," in *ACM SIGSAC Conference on CCS, AZ, USA, November 3-7, 2014*. ACM, 2014, pp. 831–843.
[45] X. Yu, H. Bai, Z. Yan *et al.*, "Veridedup: A verifiable cloud data deduplication scheme with integrity and duplication proof," *IEEE Trans. Dependable Secur. Comput.*, vol. 20, no. 1, pp. 680–694, 2023.
[46] X. Fan, J. Zhang, and X. Li, "A blockchain-based trust and reputation model resilient to forgery attacks for data sharing scenarios (RWS-BTRM)," *Int. J. Inf. Sec.*, vol. 24, no. 5, p. 205, 2025.
[47] J. Daemen and V. Rijmen, "Aes proposal: Rijndael," 1999.
[48] P. Indyk and R. Motwani, "Approximate nearest neighbors: Towards removing the curse of dimensionality," in *30th STOC, Dallas, Texas, USA, May 23-26, 1998*. ACM, 1998, pp. 604–613.
[49] A. Z. Broder, "On the resemblance and containment of documents," in *Compression and Complexity of SEQUENCES, Positano, Amalfitan Coast, Italy, June 11-13, 1997*. IEEE, 1997, pp. 21–29.
[50] D. J. Bernstein, N. Duif, T. Lange *et al.*, "High-speed high-security signatures," *J. Cryptogr. Eng.*, vol. 2, no. 2, pp. 77–89, 2012.
[51] Q. Feng, K. Yang, M. Ma *et al.*, "Efficient multi-party eddsa signature with identifiable aborts and its applications to blockchain," *IEEE Trans. Inf. Forensics Secur.*, vol. 18, pp. 1937–1950, 2023.
[52] H. W. H. Wong, J. P. K. Ma, H. H. F. Yin *et al.*, "How (not) to build threshold eddsa," in *26th Research in Attacks, Intrusions and Defenses, RAID, Hong Kong, China, Oct 16-18*. ACM, 2023, pp. 123–134.
[53] M. Castro and B. Liskov, "Practical byzantine fault tolerance," in *3rd USENIX (OSDI), New Orleans, Louisiana, USA, February 22-25, 1999*. USENIX Association, 1999, pp. 173–186.
[54] S. Purohit, R. Neupane, N. R. Bhamidipati *et al.*, "Cyber threat intelligence sharing for co-operative defense in multi-domain entities," *IEEE Trans. Dependable Secur. Comput.*, vol. 20, no. 5, pp. 4273–4290, 2023.
[55] J. R. Douceur, "The sybil attack," in *IPTPS 2002, Cambridge, MA, USA, March 7-8, 2002, Revised Papers*, vol. 2429. Springer, 2002, pp. 251–260.
[56] M. Rezvani, A. Ignjatovic, E. Bertino *et al.*, "Secure data aggregation technique for wireless sensor networks in the presence of collusion attacks," *IEEE Trans. Dependable Secur. Comput.*, vol. 12, no. 1, pp. 98–110, 2015.
[57] C. Ng and P. P. C. Lee, "Revdedup: a reverse deduplication storage system optimized for reads to latest backups," in *Asia-Pacific Workshop on Systems, Singapore, July 29-30, 2013*. ACM, 2013, pp. 15:1–15:7.
[58] B. Fan, D. G. Andersen, M. Kaminsky *et al.*, "Cuckoo filter: Practically better than bloom," in *10th CoNEXT, Sydney, Australia, Dec 2-5*. ACM, 2014, pp. 75–88.
[59] J. Zhang, X. Li, S. Xu *et al.*, "PriME-Deal: privacy-preserving bilateral data trading with efficient matchmaking and auditable fair exchange on blockchain," *CoRR*, 2026. [Online]. Available: https://doi.org/10.48550/ARXIV.2606.11539
[60] D. F. Aranha, K. Karabina, P. Longa *et al.*, "Faster explicit formulas for computing pairings over ordinary curves," in *30th EUROCRYPT, Tallinn, Estonia, May 15-19, 2011*. Springer, 2011, pp. 48–68.

## Biography Section

**Jie Zhang** (Student Member, IEEE) received the M.S. degree in computer application technology from the Yunnan Normal University in 2022. He is currently working toward the Ph.D. degree in computer science and technology with the Tianjin University. His research interests include efficient and secure data sharing within blockchain systems.

**Xiaohong Li** (Member, IEEE) received the Ph.D. degree in computer application technology from Tianjin University in 2005. She is currently a Full Tenured Professor with the Department of Cyber Security, College of Intelligence and Computing, Tianjin University. Her research interests include knowledge engineering, trusted computing, and security software engineering.

**Ruitao Feng** is a Lecturer at Southern Cross University, Australia. He received the Ph.D. degree from the Nanyang Technological University. His research centers on security and quality assurance in software-enabled systems, particularly AI4Sec & SE. This encompasses learning-based intrusion/anomaly detection, malicious behavior recognition for malware, and code vulnerability detection.

**Shanshan Xu** received the M.S. degree in computer application technology from the Yunnan Normal University in 2022. She is currently working toward the Ph.D. degree in physical geography with the School of Geographic Sciences, East China Normal University. Her research interests include Atmospheric vapor, machine learning and model building.

**Yongyang Lv** is an assistant professor at Tianjin University. He received his doctorate from Tianjin University in 2025. His research focuses on data security, AI security and network security.

**Zhé Hóu** is a Senior Lecturer at Griffith University, Australia. He obtained his Ph.D. degree from the Australian National University in 2015. His research mainly focuses on automated reasoning, formal methods, AI, quantum computing and blockchain.

**Guangdong Bai** (Member, IEEE) received the B.S. and M.S. degrees in computer science from Peking University in 2008 and 2011, respectively, and the Ph.D. degree in computer science from the National University of Singapore in 2015. He is currently an Associate Professor with the Department of Computer Science, City University of Hong Kong. His research interests include cyber security, software engineering, and machine learning.