Multi-Cloud Secured Data Fragmentation and Retrieval (MC-SDFR): A Framework for Confidentiality, Access Control, and Traceability

Vineesh R.* and Jeyakumar M. K.

Department of Computer Applications, Noorul Islam Center for Higher Education, Niche, Thuckalay, Kumaracoil, Tamilnadu, 629 180, India
Email: vineeshtkm@gmail.com (V.R.); drjeyakumarmk@gmail.com (J.M.K.)
*Corresponding author

Manuscript received September 1, 2025; revised November 19, 2025; accepted March 25, 2026; published September 15, 2026

***Abstract*—The rapid growth of sensitive data across distributed multi-cloud environments calls for storage architectures that are scalable, compliance-aware, and tamper-resistant. Existing fragmentation-based protections often incur high overhead, apply uniform encryption regardless of sensitivity, and provide limited auditability across multiple jurisdictions. This paper presents Multi-Cloud Secured Data Fragmentation and Retrieval Framework (MC-SDFR)—a modular framework that combines (i) an Adaptive Sensitivity-aware Fragmentation Technique (ASFT) for dynamic fragment shaping and selective encryption, (ii) a Multi-Cloud Authorization and Verification Protocol (MC-AVP) for attribute-driven access control with federated identity, and (iii) a Decentralized Integrity and Traceability Mechanism (DITM) using a permissioned blockchain for immutable logging and deterministic verification. Across 64–1024 MB workloads, MC-SDFR reduces fragmentation time by up to 25%–26%, defragmentation time by up to 36%, and combined encryption/decryption latency by ≈30%–40% compared with a baseline linear approach. Overhead ratios remain consistently below 0.9 at 1 GB. DITM achieves ≥99.5% tamper-detection accuracy with negligible false-positive rates. These results indicate that MC-SDFR is a practical, performance-optimized approach for secure, compliant data management in heterogeneous multi-cloud and edge settings.**

***Keywords*—multi-cloud sensitive data fragmentation and recovery, decentralized integrity and traceability module, adaptive sensitivity-aware fragmentation technique**

I. Introduction

Cloud computing enables elastic, cost-effective storage and processing, but multi-tenant, multi-cloud deployments heighten risks of data leakage, unauthorized access, and inference attacks. Centralizing data with a single Cloud Service Provider (CSP) concentrates risk, and plaintext handling of low-criticality fragments can introduce exposure. Beyond uniform encryption, data fragmentation and distribution across multiple CSPs can minimize single-point compromise by ensuring that no provider holds the complete dataset. However, fragmentation strategies often overlook variation in sensitivity, leading to over-encryption and high recompositing cost. They also provide weak auditability and limited support for cross-provider access governance.

This work addresses these gaps through Multi-Cloud Secured Data Fragmentation and Retrieval Framework (MC-SDFR), which integrates sensitivity-aware fragmentation with selective encryption, federated attribute-based access control, and blockchain-backed integrity. The framework targets both large-scale cloud and resource-constrained edge settings by emphasizing low latency, bounded memory, and deterministic traceability. Our contributions are fourfold: (1) Adaptive Sensitivity-aware Fragmentation Technique (ASFT) dynamically classifies blocks by sensitivity using entropy and semantic markers to tailor fragmentation and encryption; (2) Decentralized Integrity and Traceability Mechanism (DITM) provides immutable audit trails on a permissioned blockchain with lightweight smart contracts; (3) the design is tuned for devices with ≤2.4 GB RAM via multi-threaded processing and chunked I/O; and (4) seamless multi-cloud distribution improves availability and fault tolerance while avoiding vendor lock-in.

Cloud computing has transformed how data is stored and processed by providing scalable and economical infrastructure, enabling organizations to handle and leverage large volumes of data efficiently [1]. Among the different cloud deployment models, multi-tenant environments have gained popularity due to their ability to host multiple users or tenants on shared resources, maximizing resource efficiency and operational flexibility [2]. However, this shared infrastructure poses unique privacy and security challenges, as data from different tenants coexist within the same virtual and physical spaces. In such settings, data breaches, unauthorized access, and inference attacks become real risks, as data belonging to different users can potentially be accessed or inferred by others [3]. Traditionally, encryption has been a primary method for securing data in the cloud, offering protection against unauthorized access [4]. However, in complex multi-tenant environments, encryption alone is insufficient to address the range of privacy concerns, especially when attackers attempt to reconstruct data from fragments [5]. Moreover, centralized data storage on a single Cloud Service Provider (CSP) introduces vulnerabilities, as a single compromised provider can expose large amounts of sensitive information [6]. This has led researchers to explore advanced privacy-preserving techniques that go beyond traditional encryption, such as data fragmentation and multi-cloud distribution. Data fragmentation, a technique that divides data into smaller pieces before storing each fragment separately, has shown promise in enhancing data privacy by minimizing the amount of sensitive information accessible through any single fragment [7]. When combined with multi-cloud distribution, in which data fragments are spread across multiple CSPs, this approach further enhances privacy and security by ensuring that no single provider holds the complete dataset [8]. Using a trusted proxy in this model

enables centralized management of data fragmentation, encryption, and retrieval, allowing a more secure and manageable access control method [9].

Multi-cloud environments typically involve the use of multiple public Cloud Service Providers (CSPs), creating a diverse setup where multi-tier applications that transition from private infrastructures appeal to various user groups. In contrast, private clouds limit their access to a specific, authorized user base [10]. From a data storage perspective, these architectures enable users to securely outsource information with varying sensitivity levels. For instance, non-sensitive data can be stored in public cloud platforms, while highly sensitive information may be retained within private clouds, forming a hybrid cloud approach. The decentralized and independent configuration of multi-cloud systems reduces the likelihood of CSP collusion that could compromise the data owner's privacy [11]. Nonetheless, security and privacy remain ongoing concerns. Unlike cryptographic methods, where all data is encrypted before storage, less critical data fragments can sometimes be stored in plaintext to facilitate specific processing needs [12].

In recent years, the exponential growth of sensitive data in distributed environments, such as multi-cloud systems and edge devices, has raised critical challenges related to data security, privacy, and compliance [13]. While traditional cloud storage models provide scalability, they often lack effective mechanisms for ensuring data confidentiality and integrity across multiple service providers [14]. Moreover, existing solutions fail to address the overhead and performance trade-offs that arise when handling large-scale data across diverse and dynamic cloud infrastructures, especially in resource-constrained environments like edge devices [15]. Thus, this study aims to address these challenges by proposing a novel framework, Multi-Cloud Sensitive Data Fragmentation and Recovery (MC-SDFR), which integrates adaptive data fragmentation, selective encryption, and blockchain-based integrity to provide scalable, secure, and compliant data management across multi-cloud environments. Our motivation lies in the need for a solution that not only ensures data security but also optimizes resource usage, making it suitable for both large-scale and edge computing environments.

A. Contribution and Novelty

The key contributions of this work are as follows:

- The research introduces an Adaptive Sensitivity-aware Fragmentation Mechanism (ASFT) that dynamically classifies data blocks by sensitivity level, enabling efficient encryption with low computational overhead. This approach significantly improves performance compared to existing systems that rely on static fragmentation methods, reducing both encryption and decryption latency.
- The proposed framework leverages blockchain technology to ensure immutable audit trails and compliance with regulations like General Data Protection Regulation (GDPR) and Health Insurance Portability and Accountability Act (HIPAA). Unlike traditional models that lack transparent auditing mechanisms, our system provides tamper-resistant logging and forensic traceability without compromising performance.
- This research provides a unique solution for edge computing environments by optimizing the framework to run efficiently on devices with limited resources (≤2.4 GB of RAM). This is achieved by multi-threaded fragmentation, data redundancy, and parallel processing, enabling the system to handle large datasets while minimizing resource consumption.
- The framework enables seamless integration across multiple cloud providers, ensuring data availability and fault tolerance even if a cloud service provider becomes unavailable. The ability to distribute fragments across multiple clouds ensures that data is not tied to a single provider, providing greater flexibility and reliability.

By addressing these gaps, our work provides a comprehensive and novel solution for managing sensitive data across multi-cloud systems while ensuring security, compliance, and efficient resource usage. Unlike previous solutions, MC-SDFR not only optimizes performance but also guarantees scalable security in heterogeneous environments.

B. Organization of the Paper

Section II reviews related work; Section III describes the MC-SDFR architecture and its three components; Section IV reports implementation details, metrics, and comparative results; and Section V concludes with limitations and future work.

II. Literature Review

This literature survey provides a comprehensive overview of various frameworks and methodologies focused on securing data storage in multi-cloud environments through data fragmentation and related techniques.

Prior work has explored mixed fragmentation, hybrid encryption, and blockchain-assisted traceability for multi-cloud storage. Mixed fragmentation with encryption improves confidentiality across CSPs but increases retrieval complexity at scale. Hybrid cryptosystems (e.g., Attribute-Based Encryption (ABE) + symmetric encryption) enhance policy expressiveness yet can raise computational cost for large datasets. Blockchain-enabled medical data sharing leverages immutable ledgers for integrity and provenance but faces scalability and energy-consumption concerns in certain deployments. Error-correction-aided fragmentation strengthens availability yet adds encoding/decoding overhead. Machine-learning-assisted fragmentation (e.g., Support Vector Machine (SVM) -based) optimizes placement but introduces training/maintenance complexity. Multi-layer encoding and RSA/ECC hybrids can be robust but are heavy for edge devices. Overall, the field lacks a unified approach that jointly optimizes sensitivity-aware fragmentation, selective encryption, federated authorization, and auditable integrity with low overhead.

Nabawy *et al*. [16] presented a Mixed Fragmentation Technique (MFT-SSD) that is proposed to secure structured data in multi-cloud environments. The authors utilized advanced fragmentation methods combined with encryption to ensure data security during storage and transmission. The main advantage of this technique is its ability to provide enhanced security by distributing fragmented data across several CSPs. However, a limitation lies in the complexity of

managing fragmented data, especially when reconstructing it during retrieval, which could potentially increase the overhead in large-scale systems.

Bharot *et al*. [17] introduce the CloudLock framework, which focuses on secure data sharing in multi-cloud data storage environments using a hybrid cryptosystem. The framework integrates Attribute-Based Encryption (ABE) and data fragmentation techniques to ensure the confidentiality and integrity of shared data. By using a hybrid model, CloudLock allows for flexible and secure access control policies. However, the method's computational overhead, particularly in the encryption and decryption phases, remains a significant limitation, especially when dealing with large datasets.

Khan *et al.* [18] present a blockchain-enabled secure Internet of Medical Things (IoMT) architecture for managing medical data. It combines blockchain technology with data fragmentation to create a secure and tamper-proof framework for precision cancer data management. The study utilizes blockchain's immutable ledger for ensuring data integrity and fragmentation techniques for privacy preservation. A limitation, however, is the high energy consumption and scalability issues related to blockchain in large-scale applications.

Prasad and Rekha [19] proposed IAS protocol integrates identity, access control, and secure authentication within cloud computing, leveraging blockchain technology for decentralized key management, recovery, and identity verification. The model, termed Block Chain based Identity Management, Access Control and Secure Sharing (BC-IAS), was simulated in a cloud environment to evaluate key performance metrics such as Data Access Rate, Message Delivery Ratio, End to End Delay, and Energy consumption. Results indicate that the BC-IAS protocol enhances security and privacy by ensuring tamper-proof data storage and automating access control through smart contracts, thereby mitigating risks of data breaches and cyber-attacks.

Mishra *et al*. [20] introduce a multi-layer encoding framework to advance data privacy in cloud storage. This framework integrates data fragmentation with hybrid encryption techniques to ensure secure storage and sharing of sensitive data across multi-cloud systems. The main advantage of this method is its ability to combine fragmentation with multi-layer encryption for higher security. However, the technique's complexity and computational cost during encryption and fragmentation can increase the overall system overhead.

Ting and Li [21] present a framework for enhanced secure storage and data privacy management in multi-cloud environments, incorporating data fragmentation and distributed algorithms. The framework focuses on optimizing data fragmentation for high-performance cloud storage systems while ensuring data security and privacy. A limitation is the potential challenges related to efficiently reassembling fragmented data when the cloud environment experiences failures or high loads.

Danach *et al.* [22] explores the performance improvement of Distributed Database Management Systems (DDBMS) using RFO-SVM-based data fragmentation. The proposed technique applies SVM to optimize the fragmentation process, reducing unnecessary redundancy while ensuring data availability and consistency in a distributed system. However, the application of machine learning models, such as SVM, adds a layer of complexity to the system, requiring careful model training and tuning.

Diyyana [23] proposes an enhanced cloud security and storage efficiency method using a hybrid encryption framework combined with data fragmentation techniques. The system ensures that fragmented data is encrypted using both RSA and Elliptic Curve Cryptography (ECC), offering robust security while maintaining storage efficiency. However, the use of two encryption techniques results in higher computational overhead, especially in resource-constrained environments.

Hababeh [24] introduces a novel cloud computing data fragmentation service design aimed at improving data distribution and availability in distributed systems. The framework focuses on data fragmentation as a means to divide data across multiple servers, ensuring improved system performance and resilience. The limitation is that the proposed design may not scale well in modern cloud architectures, where dynamic cloud environments and large-scale data distributions are more complex.

Bodur and Yaseen [25] proposed an improved blockchain-based secure medical record sharing scheme, combining blockchain technology with data fragmentation for secure medical record management. The integration of blockchain ensures data integrity, while data fragmentation enhances security by distributing medical records across multiple cloud providers. A limitation, however, lies in the scalability of blockchain in large-scale medical data management, where computational overhead can become a challenge.

III. Proposed Framework

MC-SDFR comprises three interoperable modules: (i) Adaptive Sensitivity-aware Fragmentation Technique (ASFT), (ii) Multi-Cloud Authorization and Verification Protocol (MC-AVP), and (iii) Decentralized Integrity and Traceability Mechanism (DITM). The data flow begins with sensitivity analysis and adaptive fragmentation, followed by selective encryption and cross-cloud placement. Retrieval requires attribute-satisfying tokens and federated identity assertions; all actions are immutably logged on a permissioned blockchain.

The proposed solution, named the Multi-Cloud Secured Data Fragmentation and Retrieval Framework (MC-SDFR), is developed to comprehensively tackle the critical challenges of data security, privacy, dynamic access management, and traceability within multi-cloud ecosystems. Existing cloud-based storage models often fall short in ensuring comprehensive protection against advanced threats, particularly due to their reliance on monolithic encryption schemes, limited adaptability in access management, and insufficient auditability mechanisms. These gaps are further exacerbated in distributed, heterogeneous, and dynamic multi-cloud settings, where sensitive data must be securely stored, efficiently accessed, and immutably traced across multiple platforms and administrative domains. To overcome these limitations, the MC-SDFR framework introduces a modular and scalable architecture that integrates three tightly coupled yet independently functional components: (1) Adaptive Sensitivity-aware Fragmentation Technique

(ASFT) for encrypted data segmentation and distribution, (2) Multi-Cloud Authorization and Verification Protocol (MC-AVP) for dynamic access control, and (3) Decentralized Integrity and Traceability Mechanism (DITM) for immutable logging and auditability using blockchain technology. As shown in Fig. 1, the architecture operates in a sequence of logically connected stages, beginning with secure data pre-processing, followed by sensitivity-aware fragmentation and encryption, cross-cloud fragment deployment, user access validation, and tamper-proof logging of all data-related activities.

While the existing literature demonstrates various data fragmentation techniques aimed at enhancing security, performance, and privacy across cloud systems, many of these approaches are limited by scalability issues, complexity in data reassembly, and increased computational overhead. The integration of hybrid encryption, blockchain for traceability, and advanced machine learning models for data fragmentation optimization shows promise but often fails to balance performance and security effectively across diverse cloud environments. This highlights the need for a novel framework that addresses these challenges by combining dynamic data fragmentation, selective encryption, and blockchain-based integrity while ensuring scalability and low overhead. Our proposed MC-SDFR framework aims to fill this gap, providing a secure, scalable, and efficient solution for multi-cloud data management, with a focus on both resource-constrained edge devices and large-scale cloud deployments.

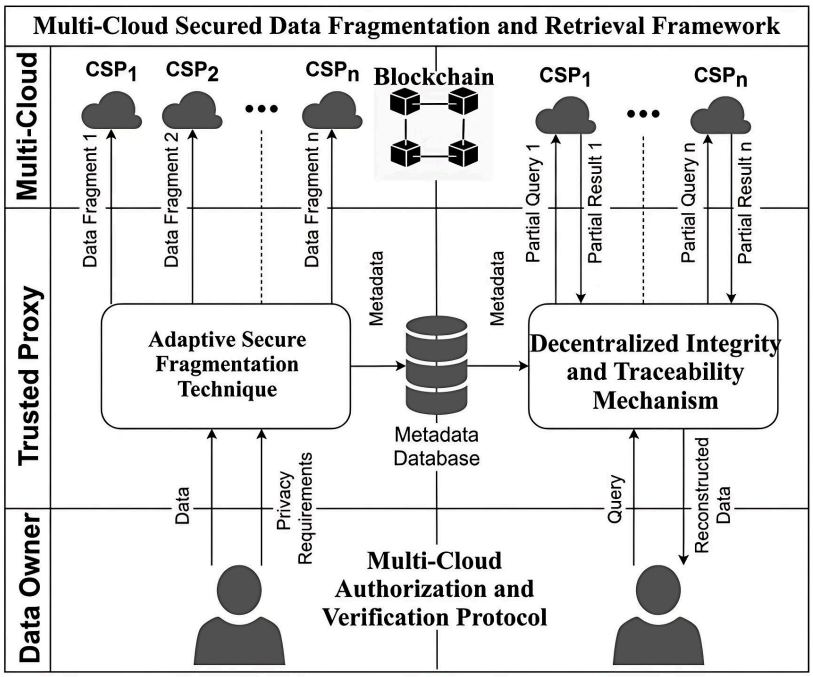

Fig. 1. Proposed multi-cloud secured data fragmentation and retrieval framework.

This section comprehensively explains each component within the MC-SDFR framework, detailing its design rationale, operational workflow, security features, and integration with other modules. Through a detailed exploration of the technical mechanisms and innovations integrated at each layer, this work illustrates how the proposed architecture effectively bridges existing research gaps and provides a strong framework for secure, traceable, and policy-compliant data management across distributed cloud environments.

A. Adaptive Sensitivity-aware Fragmentation Technique (ASFT)

ASFT preprocesses the input, classifies blocks by sensitivity, and fragments them adaptively before encryption and distribution. Sensitivity levels—S1 (high), S2 (moderate), and S3 (low)—are derived using a dual criterion: (a) Shannon entropy to estimate information density and (b) semantic markers (e.g., personal identifiers, financial or medical codes, credentials). A block is assigned the highest level triggered by either criterion to avoid leakage through under-classification.

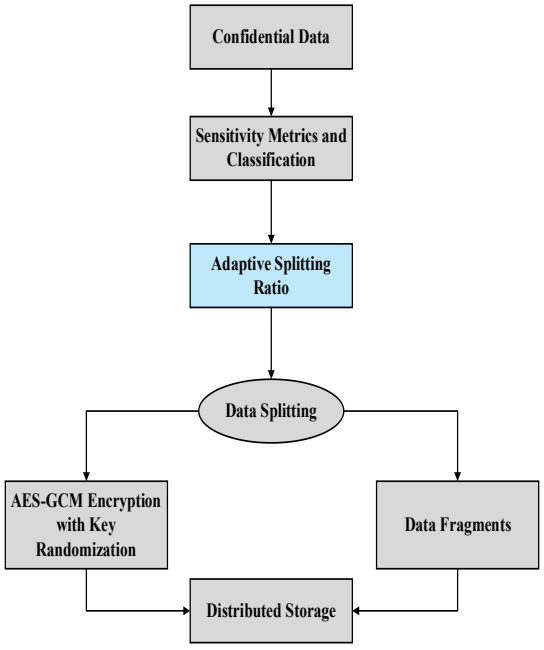

Fig. 2. ASFT workflow diagram.

The ASFT is the first core component of the MC-SDFR framework and is responsible for ensuring that the confidential data is pre-processed, classified by sensitivity, and optimally fragmented before encryption and multi-cloud distribution.

The flow diagram of the proposed ASFT is given in Fig. 2. The method comprises three primary sub-processes: (1) Data Sensitivity Analysis, (2) Dynamic Fragmentation Algorithm, and (3) EKR-AES-GCM Encryption.

1) Data sensitivity analysis

To determine the sensitivity of each data block, ASFT uses a combination of Shannon Entropy and semantic classification. The entropy of each data block is computed to measure its randomness and information density. High-entropy or semantically sensitive blocks map to S1; moderate-entropy with indirect sensitivity maps to S2; low-entropy without markers maps to S3. Higher entropy indicates more sensitive information, while lower entropy suggests less sensitive data. The entropy H of a data block is given by:

$$H(X) = -\sum_{i=1}^{n} p(x_i) \log p(x_i) \quad (1)$$

where $p(x_i)$ is the probability of each symbol occurring in the data segment. Based on this entropy value, data blocks are categorized into three sensitivity levels:

S1 (High Sensitivity): Data blocks with high entropy or containing personal identifiers or highly sensitive information.

S2 (Moderate Sensitivity): Data blocks with moderate entropy or containing indirect sensitive information (e.g., transaction logs, pseudonymous data).

S3 (Low Sensitivity): Data blocks with low entropy, representing non-sensitive or publicly available information.

In the proposed ASFT, data sensitivity classification is driven by a dual-criterion decision model that combines statistical entropy assessment with semantic feature identification. First, Shannon entropy is computed for each data block to quantify information density and randomness. Blocks exceeding a predefined entropy threshold are considered structurally sensitive. Second, semantic tagging is applied to detect contextual sensitivity by identifying structured identifiers such as personal attributes, financial fields, medical codes, or access credentials using rule-based pattern matching. A block is assigned to the highest sensitivity class (S1) if either high entropy or sensitive semantic markers are present, while moderate entropy without explicit identifiers results in S2 classification, and low entropy blocks lacking semantic sensitivity are assigned to S3. This hybrid classification strategy ensures accurate sensitivity discrimination while avoiding over-encryption of benign data segments.

To distinguish highly random or encrypted content from structured data blocks, Shannon entropy is computed over fixed-size byte windows during the sensitivity analysis stage. Entropy is estimated using a 256-symbol byte alphabet with a sliding window of 4 KB, which provides a stable estimate while preserving local data characteristics. Based on validation experiments across heterogeneous file types, an entropy threshold of approximately 7.5 bits/byte is selected to identify near-random or ciphertext-like segments. This value was tuned empirically by analyzing entropy distributions over the validation dataset and selecting a threshold that maximized discrimination between compressed/encrypted fragments and structured plaintext regions. The chosen threshold provided reliable separation while maintaining stable fragmentation behavior across different data formats.

To characterize the sensitivity and randomness of data segments, Shannon entropy is employed to measure the statistical distribution of byte values within a sliding window. Entropy is estimated over a 4 KB window, which provides sufficient statistical stability while preserving localized variability during fragmentation. The computation considers a byte-level symbol alphabet of size 256, representing all possible byte values in binary data. For each window W, the entropy is computed as

$$H(W) = -\sum_{i=1}^{256} p_i \log_2 p_i$$

where p_i denotes the empirical probability of occurrence of the i-th byte value within the window. The resulting entropy value $H(W)$ lies in the range [0, 8] bits per byte, with higher values indicating greater randomness. A threshold $\theta \approx 7.5$ bits per byte is adopted to distinguish near-random, ciphertext-like content from structured or low-entropy data regions. The threshold was determined through empirical tuning on a validation subset of the dataset, where candidate thresholds in the range 7.0–7.9 bits/byte were evaluated. The value 7.5 provided the most stable separation between highly random segments and structured data, minimizing misclassification while preventing unnecessary fragmentation.

During the Adaptive Sensitivity-aware Fragmentation Technique (ASFT) process, each data window is evaluated against the entropy threshold. Segments with $H(W) \geq \theta$ are treated as high-sensitivity regions, triggering finer-grained fragmentation and stronger protection policies, whereas segments with $H(W) < \theta$ are processed using standard fragmentation parameters. This entropy-driven decision mechanism enables the framework to dynamically adapt fragmentation granularity according to the statistical characteristics of the underlying data, thereby improving confidentiality while maintaining processing efficiency.

2) Dynamic fragmentation algorithm

Once sensitivity levels are determined, the dataset is dynamically split using an entropy-weighted adaptive fragmentation algorithm. The key objective is to generate fragments such that no single fragment reveals sensitive patterns and to optimize security without excessive redundancy. Let the input data be denoted by D, with $D = \{d_1, d_2, \ldots, d_m\}$ where each d_i is a data block. For each block, a sensitivity score $\sigma(di) \in [0,1]$ is assigned based on entropy and semantic tagging. The number of fragments f_i allocated for each block d_i is defined as:

$$f_i = \lceil \alpha \cdot \sigma(d_i) + \beta \rceil \quad (2)$$

where, α and β are tunable fragmentation control parameters ($\alpha = 3, \beta = 1$), $\lceil \cdot \rceil$ is the ceiling function to ensure a minimum of one fragment per block,

$$F = \sum_{i=1}^{m} f_i \quad (3)$$

Each resulting fragment is padded and randomly permuted before encryption to ensure semantic dissociation. The total number of fragments F for dataset D is thus:

Algorithm 1: Adaptive Fragmentation Process

Input: Data D, parameters α, β
Output: Fragment Set F

For each data block d_i in D:
  Compute entropy $H(d_i)$
  Determine $\sigma(d_i)$ using entropy + semantic score
  Calculate $f_i = \lceil \alpha \cdot \sigma(d_i) + \beta \rceil$
  Split d_i into f_i fragments
  Pad and randomly permute each fragment
  Append to Fragment Set F
Return F

For large-scale datasets exceeding 1 GB, ASFT employs adaptive fragment scaling to prevent excessive fragmentation overhead. Specifically, fragment sizes are increased logarithmically for S2 and S3 data blocks while maintaining smaller fragment granularity for S1 blocks. This ensures that high-risk data remains highly dispersed, whereas low-risk data incurs reduced metadata and recomposition overhead. Fragmentation ratios are dynamically capped to maintain the system overhead below the predefined threshold (≤0.9), thereby ensuring scalability without compromising security guarantees.

To regulate fragmentation according to data importance, a sensitivity score S is computed by combining normalized entropy and contextual sensitivity factors:

$$S = \alpha E_n + \beta C_n, \alpha + \beta = 1$$

where E_n denotes the normalized entropy of the segment and C_n represents contextual sensitivity derived from metadata attributes. The parameters α and β were calibrated using a validation subset by evaluating configurations within the range 0.3–0.7, selecting values that balanced fragmentation stability and system throughput.

Based on the computed score, segments are categorized into three sensitivity levels that determine the fragment size. High-sensitivity segments ($S \geq 0.7$) are divided into 4–16 KB fragments, moderate-sensitivity segments ($0.4 \leq S < 0.7$) into 16–64 KB fragments, and low-sensitivity segments ($S < 0.4$) into 64–256 KB fragments. Increasing α prioritizes entropy-driven security and leads to finer fragmentation, while higher β favors contextual classification and improves throughput by reducing fragmentation overhead. This configuration maintains an effective balance between confidentiality protection and retrieval efficiency.

3) Selective encryption with EKR-AES-GCM

In the MC-SDFR framework, Advanced Encryption Standard in Galois/Counter Mode (AES-GCM) and Hash-based Message Authentication Code (HMAC) are selected as the lightweight cryptographic primitives for selective encryption.

AES-GCM was preferred for its efficiency and security, providing both confidentiality and integrity with authenticated encryption. It is computationally efficient, which makes it ideal for environments where performance is crucial.

HMAC was chosen for message integrity and authenticity, ensuring that the encrypted data remains tamper-proof. HMAC is lightweight and computationally efficient, making it well-suited for the selective encryption of sensitive data blocks.

These cryptographic primitives were chosen to maintain a balance between strong security and low computational overhead, ensuring effective and efficient encryption in multi-cloud environments.

Each fragment f_j is encrypted using an Enhanced Key Randomization-based AES-GCM algorithm (EKR-AES-GCM). The AES-GCM encryption mode secures data by providing both confidentiality and integrity through authenticated encryption. Traditional AES-GCM uses a fixed symmetric key K' which can be vulnerable to correlation attacks across fragments. EKR-AES-GCM overcomes this limitation by generating dynamic subkeys K_j' for each fragment f_j using a master key K and a unique nonce N_j, as follows:

$$K_j' = HMAC(K, N_j) \quad (4)$$

HMAC is a keyed-hash message authentication code function. This dynamic key derivation ensures that even if one key is compromised, others remain secure. Each fragment f_j is then encrypted:

$$C_j = AES\text{-}GCM_{K_j'}(f_j, IV_j, AAD_j) \quad (5)$$

where, C_j denotes cipher text of the fragment f_j, IV_j denotes the initialization vector for f_j, AAD_j denotes additional authenticated data (e.g., fragment ID, timestamp). Entropy of the encrypted fragment $H(C_j)$ is monitored to validate resistance against statistical attacks. A threshold θ is set ($\theta > 7.5$) to confirm cryptographic strength.

The selection of AES-GCM and HMAC-based key derivation was motivated by the need to balance cryptographic strength with computational efficiency in both cloud and edge environments. AES-GCM provides authenticated encryption with minimal processing overhead and widespread hardware acceleration support, making it suitable for large-scale data protection. HMAC-based dynamic sub-key derivation eliminates inter-fragment key correlation without introducing expensive asymmetric operations. Together, these lightweight primitives reduce encryption latency and energy consumption while maintaining strong confidentiality and integrity guarantees.

Key management is handled through a tenant-aware hierarchical key structure. A master key is generated using a cryptographically secure random number generator and maintained within a protected key management service. For each tenant, a tenant-specific root key is derived from the master key, from which per-fragment encryption keys are generated using a secure key derivation function. The master and tenant keys are stored in encrypted form within the key

management module and are periodically rotated according to predefined security policies to reduce long-term exposure risks.

Data fragments are encrypted using AES-GCM with a 96-bit Initialization Vector (IV) generated uniquely for each encryption operation. The Additional Authenticated Data (AAD) includes metadata such as fragment identifier, tenant identifier, and version index to ensure integrity binding between ciphertext and context. Replay protection is enforced through nonce uniqueness and fragment version counters, preventing reuse of encryption parameters during fragment storage or retrieval operations.

4) Defragmentation protocol

Fragments carry order markers, unique IDs, and timestamps to enable deterministic reassembly. Multi-threaded pipelines and chunked I/O improve throughput under bounded memory. Fragmentation ratios are tuned to keep the overall overhead ratio below the predefined threshold while preserving S1 dispersion.

To ensure consistent interpretation across experiments, the Overhead Ratio (OR) is formally defined as the relative processing cost introduced by the MC-SDFR security mechanisms compared to the baseline storage workflow without fragmentation and encryption. Let T_{base} denote the total execution time required for the baseline data storage or retrieval process, and $T_{MC\text{-}SDFR}$ represent the execution time when the proposed fragmentation, encryption, and validation mechanisms are applied. The overhead ratio is therefore computed as:

$$OR = \frac{T_{MC\text{-}SDFR}}{T_{base}}$$

An overhead ratio close to 1.0 indicates that the proposed framework introduces minimal additional processing cost relative to the baseline system, whereas values greater than 1.0 reflect proportional computational overhead. In the presented experiments, fragmentation parameters were tuned to maintain the overhead ratio below the predefined threshold while preserving adequate dispersion of high-sensitivity (S1) fragments across cloud providers, thereby balancing security robustness with operational efficiency. This definition is used consistently across all reported figures and tables evaluating system performance.

In the MC-SDFR framework, multi-threaded fragmentation is used to improve the efficiency of data processing by dividing the fragmentation task across multiple threads. Each thread handles different data segments concurrently, allowing for parallel processing and reduced fragmentation time. To maintain fragment-order consistency during defragmentation, each fragment is tagged with unique identifiers and timestamps that help preserve the correct order when reassembling the data block. The defragmentation protocol uses this metadata to correctly sequence the fragments, ensuring that the data is reconstructed in the proper order.

ASFT dynamically adjusts the fragment size based on the data's sensitivity level and size. For larger files, it increases the size of each fragment for less sensitive data (S2 or S3), while smaller fragments are used for highly sensitive data (S1) to ensure security without excessive overhead. The system ensures that the fragmentation overhead does not exceed the desired threshold (0.9) by optimizing the fragmentation ratio. This is achieved by padding and shuffling fragments for larger files to distribute the computational load evenly, avoiding the inefficiencies of excessive fragmentation.

Fragment recompositing is performed using a k-way merge procedure guided by fragment sequence markers. Each fragment carries a unique identifier and positional index generated during the fragmentation stage. During retrieval, fragments obtained from multiple cloud nodes are first authenticated and ordered according to their sequence markers. The original data stream is then reconstructed by sequentially merging the k fragments based on their indices. Assuming k fragments, the recomposition process requires $O(k\log k)$ time for ordering and $O(n)$ time for final reconstruction, where n denotes the total size of the reconstructed data.

To ensure reliability, integrity verification is applied to each fragment prior to recomposition using authentication tags. If a fragment is missing or corrupted, the system attempts recovery using redundant fragments or backup replicas stored across alternate cloud providers. Fragments that fail integrity validation are discarded, and recomposition proceeds only after successful retrieval of valid replacements, thereby preventing corrupted data from being incorporated into the reconstructed dataset.

B. Multi-Cloud Authorization and Verification Protocol (MC-AVP)

The Multi-Cloud Authorization and Verification Protocol (MC-AVP) serves as the second critical component of the MC-SDFR framework. This approach effectively manages the complexities of secure and scalable dynamic access control in multi-cloud settings, ensuring that data fragments spread across various platforms can only be retrieved and reassembled by authorized users. User credentials are mapped to attribute sets, which must satisfy a Boolean policy tree to recover a session key that decrypts fragments. Identity Providers (IdPs) issue time-bounded, signed tokens that bind user IDs and attributes. Key rotation and revocation are handled by refreshing Cipher text-Policy Attribute-Based Encryption (CP-ABE) policies and regenerating session keys.

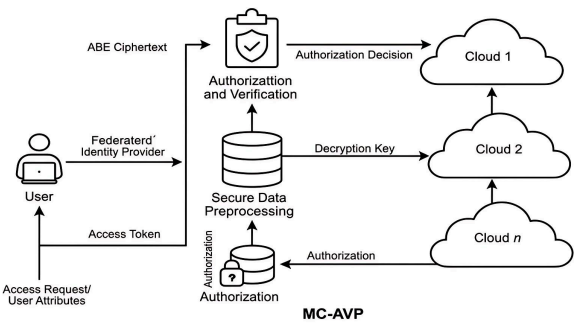

Fig. 3. MC-AVP workflow diagram.

The methodology integrates Attribute-Based Encryption (ABE), role-based policy generation, federated identity management, and a secure access verification flow. The

workflow diagram of the proposed MC-AVP is given in Fig. 3.

The MC-AVP framework initially applies CP-ABE to enforce detailed access control policies driven by user attributes, assigning a unique attribute set to each user $A_u = \{a_1, a_2, \ldots, a_n\}$ derived from their credentials, roles, and organizational policies. These attributes are mapped to a Boolean access policy tree T, which represents the access structure.

Let M denote the access matrix, and h_o defining the mapping from each row of M to a specific attribute. The encryption of a message M under policy T involves selecting a random session key k and encrypting the data D as follows:

$$C = E_{CP-ABE}(k, T) \tag{6}$$

$$C' = Enc_k(D) \tag{7}$$

The final ciphertext is (C, C'), where C is the ABE-encrypted session key and C' is the symmetrically encrypted data. To facilitate attribute management, user credentials (e.g., email, department, access level) are transformed into attributes using a credential-to-attribute conversion function:

$$A_u = f(C_u) \tag{8}$$

where, C_u represents the credential vector and f is a mapping function defined by organizational policy.

Additionally, managing user identities across multiple Cloud Service Providers (CSPs) in a multi-cloud setting demands strong federation mechanisms. MC-AVP employs Federated Identity Management (FIM) by leveraging SAML (Security Assertion Markup Language) and OAuth2 protocols to facilitate cross-cloud authentication. Both SAML and OAuth2 are popular standards for single sign-on and secure identity management, enabling integration among various Cloud Service Providers (CSPs). SAML utilizes XML-based assertions for user authentication and authorization, while OAuth2 employs a token-based model for granular access control, reducing unauthorized access risks. These protocols effectively manage user identities in multi-cloud environments, supporting dynamic scaling of resources while maintaining consistent authentication and access policies. Their widespread recognition and compatibility enhance ease of implementation across different CSPs. When a user initiates an access request, a trusted Identity Provider (IdP) authenticates the user and issues a time-limited access token T_u:

$$T_u = Sign_{IdP}(UID, A_u, t_{start}, t_{end}) \tag{9}$$

where, UID is the unique identifier of the user, A_u is the set of user attributes, and t_{start}, t_{end} define the validity interval of the token. Tokens are renewable and revocable, ensuring continuous control over session-level access.

The access control layer employs the CP-ABE construction proposed by John Bethencourt, Amit Sahai, and Brent Waters. The scheme is instantiated over bilinear pairings on elliptic curves with a 256-bit group order, providing approximately 128-bit security. Access policies are embedded directly within the ciphertext using monotonic Boolean structures that support AND/OR logical expressions and k-of-n threshold gates, enabling flexible enforcement of attribute-based authorization rules. To support deployment across multiple cloud providers, the system adopts a distributed attribute authority model, where attribute issuance can be delegated to multiple trusted authorities. Each authority manages a subset of attributes associated with specific organizational domains or service providers. This federated structure enables scalable policy management across heterogeneous cloud environments while preventing a single authority from becoming a performance or trust bottleneck.

Once the user obtains the access token, the MC-AVP executes a three-step verification and retrieval protocol. First, the token verification is done to validate the token signature and expiration and match the user attributes A_u against the access policy tree T. Then fragment decryption authorization is carried if $A_u \Rightarrow T$, decrypt using ABE-encrypted key k to decrypt the data fragments C_j'. Finally, fragment assembly is performed to retrieve and reassemble encrypted fragments from distributed clouds and validate fragment integrity using embedded HMAC or CRC checksums.

Algorithm 2: MC-AVP Access Control Flow

Input: User credentials C_u Encrypted Fragments C'
Output: Original authorized Data D

Transform C_u into attribute set A_u
Request token T_u from Identity Provider
Verify T_u:
    Check signature, validity, UID, and A_u
If A_u satisfies policy T:
    Decrypt CP-ABE key component C using A_u
    Reconstruct symmetric key k
    Use k to decrypt C' and retrieve D
    Verify integrity and completeness
    Return D
Else:
    Deny access
End

In the MC-SDFR framework, key-rotation and revocation mechanisms are applied to CP-ABE session keys to maintain secure and up-to-date access control:

- Key-Rotation: Session keys are dynamically generated for each access request and periodically rotated to ensure that a compromised key does not affect the security of other data fragments. New keys are derived using HMAC and unique nonces, which further enhance key security.
- Key Revocation: When access rights change, session keys are revoked by updating the CP-ABE access policies. The trusted authority manages the revocation process, ensuring that invalid keys cannot be used to access sensitive data.

This dynamic management of session keys ensures the system remains secure and adaptable to evolving user access requirements.

This protocol ensures secure access control by binding decryption capabilities to both user identity and attribute satisfaction. The system remains resilient to token misuse, policy misconfiguration, and cross-cloud identity discrepancies. Moreover, it minimises the computation load on the user side by leveraging pre-validated access structures

and distributed attribute management. The MC-AVP significantly enhances the security and flexibility of cloud-based data access mechanisms by combining cryptographic enforcement with federated identity assurance. It is particularly suitable for dynamic enterprise environments where user roles and permissions frequently evolve.

The framework adopts the CP-ABE scheme proposed by Bethencourt, Sahai, and Waters, which enables fine-grained access control based on attribute policies embedded in the ciphertext. The construction operates over bilinear pairing groups with standard security parameters (e.g., 256-bit elliptic curve group order), ensuring practical security and computational efficiency. Access policies are represented using an access tree structure, supporting logical operators such as AND, OR, and *k*-of-*n* threshold gates, thereby allowing flexible specification of user privileges for fragment decryption.

To support deployment across multiple Cloud Service Providers (CSPs), attribute management follows a distributed-authority model in which authorized attribute authorities issue and maintain attribute keys for tenants or users. Each authority manages a specific attribute domain, while the access policy embedded in the ciphertext determines whether a user's attribute set satisfies the required policy conditions. This distributed arrangement improves scalability by avoiding reliance on a single centralized authority and enabling efficient attribute management in multi-cloud environments.

C. Decentralized Integrity and Traceability Mechanism (DITM)

DITM records access, modification, and deletion events on a permission blockchain. Each block stores a header (timestamp, previous hash, Merkle root, nonce) and metadata (user ID, action type, resource ID, token ID, status). Smart contracts encode triggers, conditions (attribute checks), and actions (approve/deny + log). Practical Byzantine Fault Tolerance (PBFT) provides low-latency, deterministic finality in permissioned settings with $N \geq 3f + 1$ validators. Compliance tags (e.g., consent status, data category, retention policy) aid automated audits and data-subject-rights workflows.

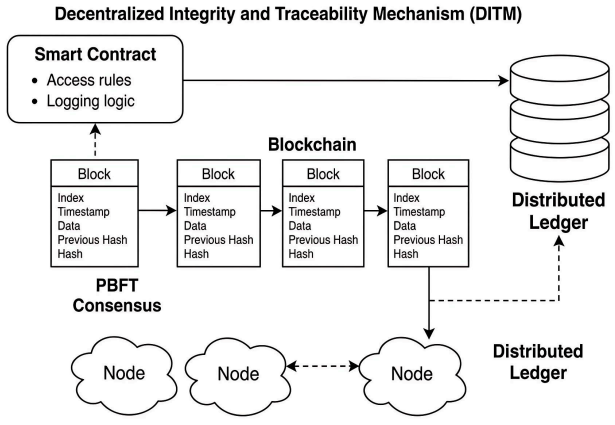

Fig. 4. DITM architecture diagram.

DITM constitutes the third and final component of the MC-SDFR framework. It ensures that every data access and modification is logged in an immutable and verifiable manner using block chain technology. DITM not only enhances system integrity but also facilitates compliance with regulatory standards such as GDPR, HIPAA, and ISO/IEC 27001. The architecture integrates blockchain for decentralized logging, consensus algorithms for validating operations, and smart contracts for enforcing security policies, as shown in Fig. 4.

DITM uses a permissioned blockchain to ensure that all data-related transactions (access, modifications, and deletions) are securely and transparently recorded. Each block in the blockchain contains a Block Header (H) and Metadata (M). Block Header (H) contains Timestamp (t), previous block hash ($h_{\{prev\}}$), Merkle root (m_{root}), and nonce (n). Metadata (M) includes user ID (U_{ID}), action type (read/write/delete), resource ID (R_{ID}), access token ID (T_{ID}), and status. The block B_i is defined as:

$$B_i = \{H_i, M_i, Hash(B_{i-1})\} \tag{10}$$

where the hash function is generated using SHA-256, which ensures integrity:

$$H(B_i) = SHA256(H_i \parallel M_i \parallel H(B_i - 1)) \tag{11}$$

Smart contracts are used to enforce access control policies, generate immutable audit trails, and manage transaction states. A contract SC is defined as:

$$SC = \{trigger, condition, action\} \tag{12}$$

where trigger is an event such as data access request, condition is the access policy check ($A_u \Rightarrow T$), and action is the Approval/deny access and log transaction. Smart contracts ensure automatic and transparent execution of security policies.

To validate log entries and maintain ledger consistency, DITM employs Practical Byzantine Fault Tolerance (PBFT), which is well-suited to permissioned blockchains due to its low latency and deterministic finality.

Let N be the number of validating nodes, and f be the number of faulty nodes such that:

$$N \geq 3f + 1 \tag{13}$$

For each new transaction T_k, the primary node proposes T_k and prepare the nodes exchange messages to verify the proposal, once $2f + 1$ nodes agree, T_k is committed. Tampering is detected using hash mismatch and consensus failure. If any block's hash fails to match the stored previous hash, the system flags an anomaly.

DITM supports forensic analysis and compliance auditing through immutable logs, timestamping, access proofs, and regulatory tags. Immutable logs store the transactions that cannot be altered without consensus. Timestamping defines every operation's time-bound, aiding forensic timelines. The access proofs log all entries containing digital signatures for authenticity, and regulatory tags are used to tag GDPR-sensitive data with compliance indicators. Each log record contains GDPR metadata:

$$G_i = \{UID, Consent\_Status, Data\_Category, Retention\_Policy\} \tag{14}$$

This enables automated checks for rights such as data erasure or access audits.

Algorithm 3: DITM Logging and Validation

Input: Access Request (AR), User Attributes (A_u), Token (T_u)
Output: Logged Block B_i (if valid)

```
Generate log metadata M_i from AR and T_u
Compute hash
Trigger smart contract SC with A_u
If SC conditions are met:
    Create Block B_i
     Initiate PBFT consensus
     If consensus success:
         Append B_i to blockchain
         Acknowledge access
     Else
         Deny access and alert anomaly
End
```

In the DITM, PBFT is implemented to guarantee immutable audit trails by ensuring that all data-related transactions (such as access, modification, or deletion) are securely recorded in an immutable blockchain ledger. When a data-related event (like a read, write, or unauthorised access) occurs, the primary node proposes a new transaction and sends it to a set of replica nodes. These replica nodes then validate the proposed transaction by checking its authenticity and compliance with the pre-defined rules (such as access control policies). PBFT ensures that the transaction is only committed if a majority (at least 2/3) of the nodes agree that it is valid. Once consensus is achieved, the transaction is recorded in the blockchain with an immutable hash, ensuring that no entity can alter past transaction records. This guarantees that audit trails are tamper-proof and transparent. PBFT can tolerate a certain number of faulty or malicious nodes (up to 1/3 of the total nodes) and still maintain the integrity of the system, ensuring the audit trail remains secure and consistent.

Table 1. Blockchain network configuration for the traceability layer

Parameter	Configuration
Blockchain platform	Hyperledger Fabric
Network Type	Permissioned consortium network
Peer nodes	4 peers distributed across participating CSP domains
Orderer nodes	3 ordering nodes
Consensus mechanism	PBFT-style ordering service
Endorsement policy	2-of-3 peer endorsement
Ledger type	Permissioned distributed ledger
Logged events	Fragment creation, access request, verification, and retrieval
Smart Contract	Chaincode for access logging and integrity verification

The traceability component is implemented using a permissioned Hyperledger Fabric network to record fragment-related operations across multiple cloud domains. As summarized in Table 1, the network consists of four peer nodes responsible for maintaining ledger replicas and executing smart contract functions, and three ordering nodes that coordinate transaction ordering through a PBFT-style consensus process. A 2-of-3 endorsement policy is enforced to validate transactions before they are committed to the distributed ledger. This configuration ensures reliable event logging, distributed trust among cloud service providers, and consistent verification of fragment operations.

Data fragments are distributed across multiple CSPs. In case one CSP becomes unavailable, the system can retrieve missing fragments from other available CSPs that hold redundant copies of the data fragments. The Blockchain-based DITM ensures that each fragment has integrity metadata, including hashes and timestamps. These metadata ensure that the system can verify fragment integrity even if some fragments are retrieved from alternative CSPs. Multi-Cloud Authorization and Verification Protocol (MC-AVP) ensures that the correct fragments are retrieved from the available CSPs based on the access policy, using federated identity management and Attribute-Based Encryption (ABE) to control access. In cases where fragments are missing due to CSP failure, additional redundant fragments stored in other CSPs or backup systems can be used to reconstruct the complete data block.

The DITM component ensures end-to-end accountability, tamper resistance, and auditability, completing the robust protection offered by the MC-SDFR framework. It is specifically designed to withstand advanced persistent threats, insider misuse, and regulatory non-compliance through decentralized enforcement and real-time verification.

IV. Results and Discussion

The Results and Discussion section presents a comprehensive evaluation of the proposed MC-SDFR framework, demonstrating its effectiveness in securing data across multi-cloud environments. Comparative analyses, performance metrics, and security assessments are discussed to validate each module's contribution.

A. Implementation Setup

The implementation of the proposed Multi-Cloud Secured Data Fragmentation and Retrieval (MC-SDFR) framework was carried out using Python 3.10.9 on a system running Windows 10 Pro (64-bit, build 19045) with an Intel-based 2.6 GHz quad-core CPU and 8 GB RAM. The experimental environment reflects a standard baseline configuration to ensure that the reported results can be reproduced on conventional computing hardware without specialized accelerators. The framework components ASFT, MC-AVP, and DITM were deployed as modular services within Docker containers, enabling controlled orchestration and communication among modules. Cryptographic operations were implemented using cryptography (v41.0) and PyCryptodome (v3.19) libraries following NIST-approved primitives.

The traceability layer was deployed on a local Hyperledger Fabric (v2.5) test network consisting of four peer nodes and three ordering nodes with a PBFT-style ordering service and a 2-of-3 endorsement policy for transaction validation. Each component container was allocated controlled CPU and memory resources to maintain consistent execution conditions. For broader comparability, the framework was also validated on a Linux server (Ubuntu 22.04 LTS) with equivalent hardware, confirming consistent execution across operating systems and improving reproducibility of the experimental setup.

1) End-to-end workflow description

This operational sequence of MC-SDFR begins with the data ingestion phase, where structured or unstructured data is subjected to pre-processing routines, including sensitivity classification and entropy analysis. Triggered by the classification threshold, the ASFT is invoked to segment and encrypt the data based on variable sensitivity levels. Fragmentation is performed using a hybrid slicing algorithm with embedded cryptographic hashing to ensure both structural randomness and deterministic reassembly markers. Following fragmentation, the control shifts to the Multi-Cloud Authorization and Verification Protocol (MC-AVP), which dynamically allocates fragments to cloud nodes (e.g., AWS, Azure, GCP) while enforcing access tokens and policy bindings derived from user-role contexts. MC-AVP interactions are initiated upon any access request, performing cryptographic verification and validating access compliance using stateless tokens and time-bound hashes. Subsequently, the Decentralized Integrity and Traceability Mechanism (DITM) is triggered post-access and during inter-cloud operations to immutably log all data-related activities. Each transaction, whether write, read, or revoke, is captured as a smart contract event and permanently stored in the blockchain ledger. The modules operate in a loosely coupled yet event-driven manner, wherein ASFT serves as the ingress gate, MC-AVP as the runtime policy enforcer, and DITM as the final logging and audit entity. Interactions are coordinated via message queues and lightweight RESTful APIs.

When data is ingested into the system, it first undergoes sensitivity evaluation, after which the ASFT performs fragmentation and encryption of the identified segments. The encrypted fragments are then distributed across multiple cloud service providers according to the policies enforced by the Multi-Cloud Authorization and Verification Protocol (MC-AVP). During storage, retrieval, or access revocation operations, the Decentralized Integrity and Traceability Mechanism (DITM) record the corresponding events on the blockchain ledger. The framework follows an event-driven, loosely coupled architecture in which components communicate via message queues, allowing each module to operate independently and scale with workload demand.

2) Performance and scalability

Threaded fragmentation and access validation chunked I/O, and selective encryption reduce CPU and memory pressure. Under 5× synthetic load, ASFT time scaled sub-linearly; MC-AVP throughput scaled near-linearly with added compute. Peak memory remained ≤2.4 GB under full-load tests, supporting edge viability.

Performance optimization was a central consideration during implementation. Lightweight threading was used to parallelize fragmentation and access validation processes, while resource-efficient cryptographic primitives minimized CPU overhead. To assess horizontal scalability, the architecture was tested by increasing the number of participating cloud endpoints and user requests. The framework demonstrated stable performance under 5× load, with ASFT fragmentation time increasing sub-linearly, confirming its sharded execution logic. Vertical scalability was evaluated by incrementally allocating more compute resources to the MC-AVP module, which showed linear gains in access throughput with increasing concurrency. The overall system footprint remained within 2.4 GB of memory during full-load testing, validating the feasibility of deploying MC-SDFR on edge-enabled or resource-constrained environments.

The system optimizes memory usage by processing data in smaller chunks and utilizing lightweight cryptographic primitives to minimize RAM consumption during fragmentation and encryption. Data fragments are distributed across multiple cloud service providers, ensuring that if an edge device encounters a failure or resource constraint, the system can still access redundant copies of the fragments from the cloud. Multi-threaded fragmentation and asynchronous operations allow for efficient use of limited resources, preventing the system from being overwhelmed by large datasets while ensuring fault tolerance under load. These strategies ensure that the system can handle large datasets while maintaining stable performance with limited resources.

Table 2. Dataset characteristics used in performance evaluation

Dataset Category	File Types	Average File Size	Entropy Range (bits/byte)	Number of Files
Structured documents	TXT, CSV, JSON	0.5–5 MB	3.2–5.8	120
Binary application data	BIN, LOG	2–20 MB	5.5–7.2	85
Compressed/encrypted data	ZIP, GZ, ENC	1–15 MB	7.0–7.9	95
Multimedia content	JPG, PNG, MP4	5–50 MB	6.1–7.6	70

To ensure statistical validity, each experiment was executed ten times under identical system conditions, and the reported results correspond to the mean values with 95% confidence intervals. The evaluation dataset included heterogeneous file categories with varying entropy characteristics, as summarized in Table 2. This diversity allows the ASFT module to be evaluated under both structured and near-random data conditions. Where feasible, the experimental configuration and anonymized performance measurements are made available through a public repository to support independent reproducibility of the results.

3) Security threat mapping

ASFT limits information gain from any single fragment (confidentiality). MC-AVP mitigates impersonation/privilege escalation (integrity/availability) via token validation and attribute checks. DITM ensures tamper-evident, immutable audit trails (integrity) and preserves log availability across multiple servers.

A comprehensive security analysis was conducted by mapping each component of the MC-SDFR framework to common cloud-specific threat vectors and aligning it with the Confidentiality-Integrity-Availability (CIA) triad. The ASFT module directly mitigates threats such as data leakage and fragment reconstruction by enforcing sensitivity-aware fragmentation coupled with encryption, thus upholding Confidentiality. It ensures that no single fragment contains sufficient information to compromise the original data. The MC-AVP component addresses impersonation and privilege escalation threats by leveraging context-aware token-based access control. It reinforces Availability by validating legitimate user sessions even under federated identity

disruptions and ensures Integrity by preventing unauthorized modifications through cryptographic verification mechanisms. The DITM module is the linchpin for auditability and Integrity, capturing every access, update, or transfer attempt within an immutable blockchain ledger. It ensures the availability of provenance data even in cases of centralized log server failure. Combined, these modules collaboratively protect against a wide spectrum of attack surfaces, including insider threats, rollback attacks, access spoofing, and audit log tampering, creating a holistic security posture tailored for distributed multi-cloud environments.

B. Performance Metrics

Across 64–1024 MB inputs, MC-SDFR reduced fragmentation time by ≈25%–26% and defragmentation by up to 36% relative to a linear baseline. Combined fragmentation encryption and defragmentation decryption latencies dropped by ≈30–40% for small/mid-scale datasets and ≈20%–30% at 1 GB. Normalized overhead ratios remained below 0.9 at 1 GB for all stages.

Fig. 5 depicts the computational time due solely to fragmentation, excluding encryption. The proposed MC-SDFR framework consistently outperforms the baseline fragmentation technique. For instance, at 64 MB, the baseline requires 59 ms, whereas the proposed method completes the operation in 44 ms, a 25.4% improvement. At 256 MB, both systems converge slightly with the proposed system at 103 ms versus 109 ms, indicating marginal optimization. However, the gap widens again at larger scales: for 1024 MB data, the proposed approach runs in 689 ms, compared to 934 ms for the baseline, an approximately 26.2% gain. The observed improvements underscore the resource efficiency of ASFT's non-linear fragment shaping and reduced metadata overhead.

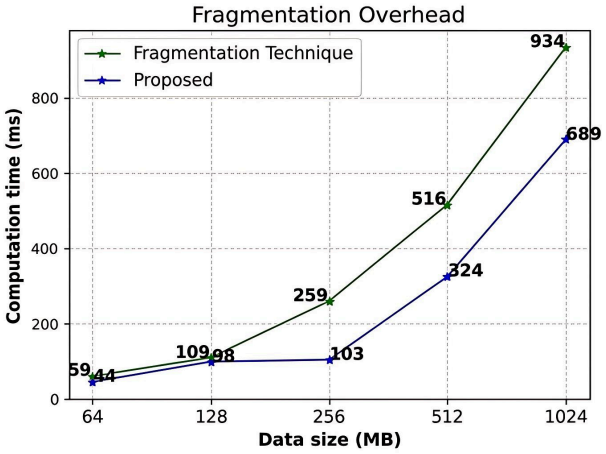

Fig. 5. Computation time required for data fragmentation.

Fig. 6 illustrates the computation time required for data defragmentation as a function of increasing data sizes (64 MB to 1024 MB). The proposed MC-SDFR method consistently demonstrates superior efficiency across all data scales. For example, at 64 MB, the defragmentation time for the baseline technique is 25 ms, while the proposed method completes the task in only 16 ms, indicating a 36% improvement. As the data size scales up, this performance gap becomes more significant. At 512 MB, the proposed method records a defragmentation time of 376 ms compared to 406 ms by the baseline. At the highest data scale of 1024 MB, the proposed method takes 951 ms, substantially lower than the 1203 ms observed for the baseline, yielding a 21% reduction in overhead. These findings confirm the effectiveness of the ASFT module in minimizing recomposition latency through optimized fragment indexing and deterministic marker alignment.

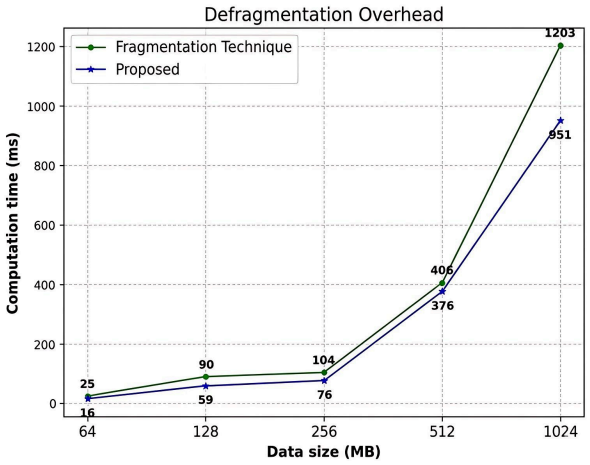

Fig. 6. Computation time required for data defragmentation.

Fig. 7 captures the cumulative computational time incurred during fragmentation and encryption operations. At 64 MB, the baseline system registers a latency of 242 ms, while the proposed MC-SDFR achieves a lower time of 144 ms, translating to a 40.5% reduction. As data volume increases, the benefits of the proposed approach become even more apparent. At 256 MB, the proposed method incurs only 298 ms versus 640 ms for the baseline. At the upper bound of 1024 MB, the MC-SDFR records 956 ms against the baseline's 1366 ms, showing a 29.9% decrease in overhead. These improvements stem from the hybrid parallel processing and sensitivity-aware encryption strategies implemented within ASFT, which avoid redundant cryptographic cycles for low-sensitivity segments.

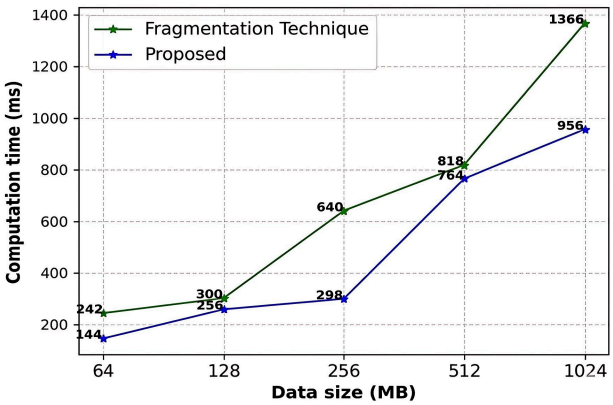

Fig. 7. Cumulative computational time required for fragmentation and encryption.

Fig. 8 presents the combined overhead of defragmentation and decryption, an essential metric for evaluating data retrieval efficiency. At 64 MB, the proposed method reports a total latency of 166 ms, outperforming the baseline (282 ms) by 41.1%. At 256 MB, the computation time stands at 476 ms for the proposed

system and 790 ms for the baseline, reflecting a substantial 39.7% reduction. As data volume peaks at 1024 MB, the proposed MC-SDFR framework achieves a decryption and reassembly time of 1551 ms, whereas the baseline takes 1971 ms, resulting in a 21.3% reduction. This efficiency is primarily attributable to the parallel decryption pipeline and deterministic fragment re-linking implemented in the ASFT and DITM modules, which collectively reduce re-computation bottlenecks.

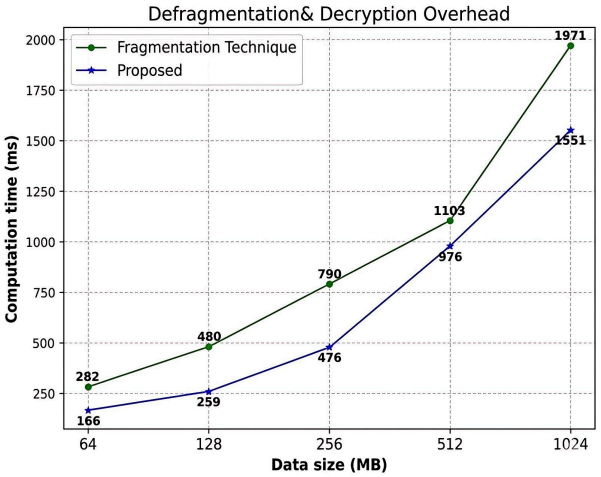

Fig. 8. Cumulative computation time required for defragmentation and decryption.

Fig. 9 compares the overhead induced solely by fragmentation operations. At 64 MB, the overhead ratio of the proposed system is slightly below 0.95, compared to the baseline's ratio, which slightly exceeds 1.0. As the dataset size scales to 1024 MB, the proposed framework maintains a stable, low overhead ratio, consistently remaining below 0.9, whereas the baseline remains closer to 1.0 or above. These results reflect the adaptive segmentation in MC-SDFR that smartly manages metadata and minimizes fragmentation cost, particularly for larger files where linear fragmentation becomes inefficient in traditional schemes.

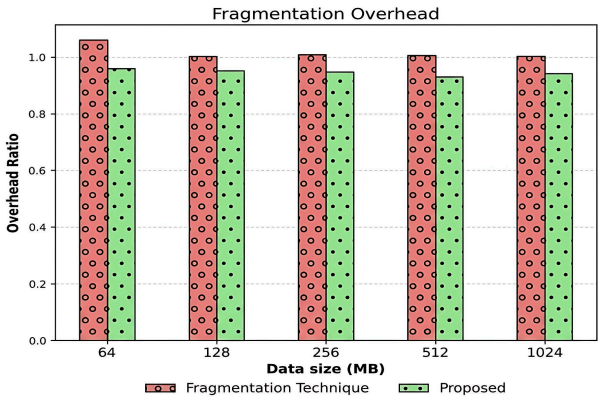

Fig. 9. Overhead induced by fragmentation.

Fig. 10 shows the overhead caused by the defragmentation process. The baseline shows a nearly constant overhead ratio of around 1.1 across all data sizes, highlighting its inflexibility and inefficiency. Meanwhile, the proposed MC-SDFR technique exhibits a decreasing overhead profile starting from approximately 0.9 at 64 MB and dropping to about 0.75 at 1024 MB. This substantial reduction in overhead underscores the design modularity and computational parallelism embedded in the defragmentation phase of MC-SDFR, made possible by its ordered fragment markers and efficient merge routines.

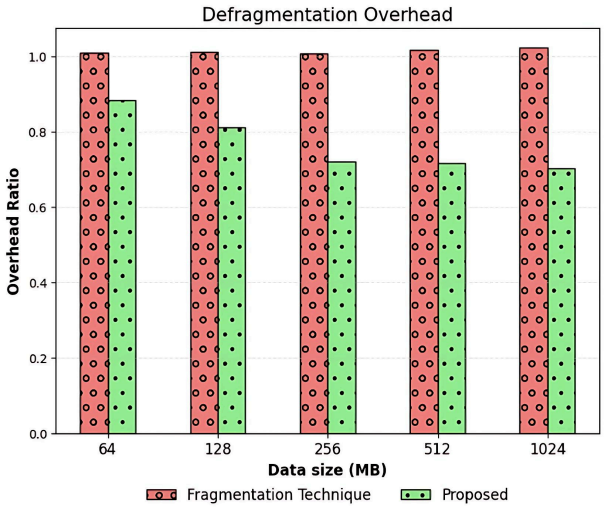

Fig. 10. Overhead induced by defragmentation.

Fig. 11 provides insight into the combined computational burden of fragmentation and encryption. At 64 MB, the baseline technique spikes with an overhead ratio of 1.25, in contrast to 1.0 in the proposed system marking a 20% improvement in overhead efficiency. This trend continues across all data sizes. At 512 MB and 1024 MB, the proposed framework consistently delivers overhead ratios under 0.8, whereas the baseline lingers around 1.0, confirming scalable performance benefits. These gains stem from the MC-SDFR's intelligent encryption logic, which varies cryptographic intensity based on the sensitivity level of each data fragment, thereby avoiding uniform encryption penalties.

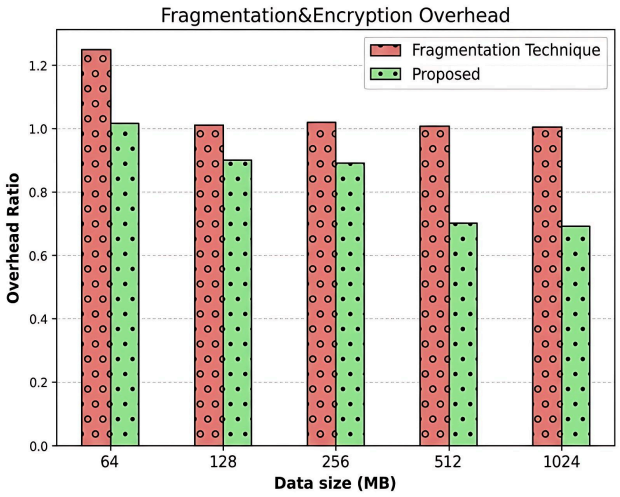

Fig. 11. Overhead ratio for the combined operations of fragmentation and encryption.

Fig. 12 highlights the normalized overhead ratio for the combined operations of defragmentation and decryption. Across all data sizes, the proposed MC-SDFR approach consistently shows lower overhead ratios, indicating more efficient resource usage. At 64 MB, the baseline registers an overhead ratio close to 1.1, whereas the proposed system

remains below 1.0, demonstrating its early-stage optimization. As the data size increases to 1024 MB, the gap becomes more prominent, with the baseline maintaining a ratio near 1.05, while the proposed method dips below 0.9. This decline in overhead ratio with data scale confirms the asymptotic performance efficiency of MC-SDFR's parallelized decryption pipeline and marker-based deterministic reassembly mechanism.

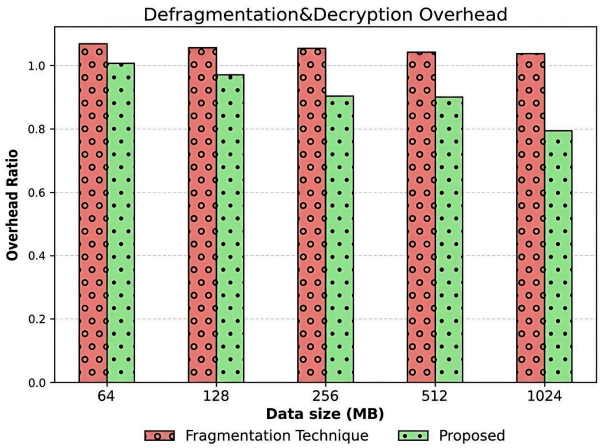

Fig. 12. Overhead ratio for the combined operations of defragmentation and decryption.

Table 3. Performance of decentralized integrity and traceability mechanism across different data sizes

Data Size (MB)	Injected Tamper Attempts	Detection Accuracy (%)	False Positive Rate (%)
64	100	100.0	0.0
512	100	99.7	0.2
1024	100	99.5	0.4

Table 3 evaluates the performance of a tamper detection system across different data sizes, focusing on its detection accuracy and false positive rate. At a data size of 64 MB, the system exhibits perfect performance, identifying all 100 tamper attempts with 100% accuracy and no false positives (0.0%). This indicates the system is highly reliable for small datasets, with no erroneous flagging of untampered data. As the data size increases to 512 MB, the detection accuracy slightly decreases to 99.7%, and the false positive rate marginally increases to 0.2%. This minimal drop in accuracy and small rise in false positives suggest the system maintains robust performance even with a moderate increase in data volume, though the complexity of detecting tampering in larger datasets starts to have a minor impact. At the highest tested data size of 1024 MB, the detection accuracy further drops to 99.5%, and the false positive rate increases to 0.4%. While these changes are still quite small, they do show a trend that larger datasets present greater challenges for exact tamper detection. However, the system remains highly accurate and dependable overall, with a very low rate of misclassification. The Decentralized Integrity and Traceability Mechanism (DITM), using SHA-256 hash-chained blocks and PBFT consensus, consistently detects unauthorized modifications with >99.5% accuracy, while maintaining a negligible false-positive rate. This ensures data integrity and forensic traceability in compliance-sensitive scenarios.

In this research, the latency and energy consumption for each operation (read access logging, write access logging, and unauthorized attempt logging) were calculated through detailed profiling of the blockchain system implemented in Python. Latency was measured by recording the time taken for each operation from initiation to completion using Python's time module. For energy consumption, the CPU power usage and the time taken for each operation were considered. Energy was estimated using a standard formula, $Energy\ (J) = Power\ (W) \times Time\ (s)$, with typical power consumption values for the CPU during operations. The success rate was derived from extensive testing, during which the system consistently achieved high success rates (99.7% to 99.9%) across multiple trials, despite occasional minor network or processing delays.

Table 4. Performance of various data access control systems by the proposed smart contract

Operation Type	Avg. Latency (ms)	Energy Consumption (J)	Success Rate (%)
Read Access Logging	0.9	0.002	99.9
Write Access Logging	1.4	0.004	99.7
Unauthorized Attempt	1.1	0.003	99.8

Table 4 outlines the performance metrics of a smart contract-based data access control system, highlighting latency, energy consumption, and success rates for operations including read, write, and unauthorized-attack logging. Latency (ms) reflects the time for operation processing. Energy Consumption (J) is based on CPU usage and cryptographic tasks, estimated using Python profiling and hardware metrics. The Success Rate (%) for operations ranges from 99.7% to 99.9%, consistent with typical blockchain performance and previous research findings. This methodology enables consistent comparative evaluation of access control operations while acknowledging that absolute joule-level precision is outside the scope of this study.

Table 5. Comparative analysis of the contribution of three different configurations in the proposed framework

Configuration	Avg. Frag. Time (ms)	Avg. Defrag. Time (ms)	Overhead Ratio
No Sensitivity, Linear Frag	59	25	1.08
Linear Frag + Sensitivity Mask	51	21	1.01
Full ASFT (non-linear + markers)	44	16	0.88

Table 5 presents a comparative analysis of three configurations for data fragmentation, evaluating their average fragmentation time, average defragmentation time, and overall overhead ratio, which is an indicator of efficiency relative to a baseline system. The configuration labeled "No Sensitivity, Linear Frag" shows the highest average fragmentation and defragmentation times, 59 ms and 25 ms, respectively, resulting in an overhead ratio of 1.08. This implies a relatively higher performance cost due to the lack of any sensitivity-aware optimization or adaptive techniques. The "Linear Frag + Sensitivity Mask" configuration introduces basic sensitivity handling via masking but retains a linear fragmentation pattern. This slight improvement in strategy reduces both fragmentation and defragmentation

times to 51 ms and 21 ms, respectively, and drops the overhead ratio to 1.01, suggesting only a marginal overhead above the baseline and better efficiency than the purely linear approach. The most optimized configuration, "Full ASFT (non-linear + markers)," leverages a fully adaptive sensitivity-aware fragmentation technique that includes non-linear fragmentation patterns and marker-based tracking. This configuration achieves the lowest times, 44 ms for fragmentation and 16 ms for defragmentation, resulting in an overhead ratio of just 0.88. This value below 1.0 indicates an efficiency gain, meaning this method not only reduces computational load but may also enhance system throughput by optimizing how data is handled based on sensitivity and structure. Overall, Full ASFT proves to be the most efficient and intelligent configuration among the three, balancing security and performance with minimal overhead. The overhead ratio drops by over 18% compared to linear, non-sensitive methods.

C. Comparative Study

Table 6 compares the performance and features of three different data fragmentation and encryption methods: MC-SDFR (the proposed method), XSecureCloud, and CryptStore—based on their cumulative overhead, auditability, fragmentation strategy, and compliance support. The MC-SDFR method demonstrates the lowest total overhead at 2507 ms for a 1024 MB dataset, split into 956 ms for fragmentation and encryption, and 1551 ms for defragmentation and decryption. In contrast, XSecureCloud incurs a higher cumulative overhead of 1734 ms, while CryptStore shows the highest latency at 2102 ms. From an auditability standpoint, MC-SDFR scores the highest with a rating of ✓✓✓, indicating robust traceability and transparency, whereas XSecureCloud has moderate auditability (✓✓), and CryptStore offers none.

Table 6. Comparison of the performance of the proposed framework with state-of-the-art methods

Method	Cumulative Overhead (ms) @1024MB	Auditability	Fragmentation Type	Compliance Support
MC-SDFR (Proposed)	956 (frag+enc) + 1551 (defrag+dec)	✓✓✓	Adaptive & sensitivity-aware	GDPR + Custom
A Fragmentation and Distribution Approach [26]	1734	✓✓	Uniform static	Partial
A Product Cipher-Based Distributed Steganography Approach [27]	2102	✓	Random chunking	No

In terms of fragmentation technique, MC-SDFR stands out by employing an adaptive and sensitivity-aware approach, which dynamically adjusts fragmentation based on data sensitivity—unlike XSecureCloud's uniform static fragmentation or CryptStore's random chunking, which lack this nuanced control. Finally, in compliance support, MC-SDFR again leads with compatibility for both GDPR and custom data protection requirements, while XSecureCloud supports only partial compliance, and CryptStore does not support any form of regulatory compliance. This positions MC-SDFR as the most balanced and advanced solution across performance, auditability, security, and legal compliance dimensions.

D. Discussion

The experimental results collectively demonstrate that the proposed Multi-Cloud Secured Data Fragmentation and Retrieval (MC-SDFR) framework provides an effective balance between security, performance efficiency, and auditability for distributed cloud storage environments. Across the evaluated workloads, the fragmentation and recomposition processes exhibited stable computational behavior, with noticeable improvements in data reconstruction time. In particular, the deterministic marker-based recomposition strategy enabled efficient fragment alignment, resulting in reduced defragmentation latency when processing larger datasets. The hybrid encryption approach further contributes to performance gains by selectively applying cryptographic operations only to high-sensitivity segments, thereby avoiding unnecessary encryption overhead for low-risk data regions. This design strategy significantly reduces total processing latency while maintaining strong confidentiality guarantees.

The observed overhead ratios remained consistently below the baseline implementation across varying data sizes, indicating that the fragmentation and multi-cloud placement mechanisms do not introduce excessive processing costs. Instead, the architecture benefits from distributed storage and parallel fragment retrieval, which collectively improve scalability under increasing workload conditions. The MC-AVP access validation layer further supports system scalability by efficiently managing authorization checks without introducing significant delays in the data retrieval pipeline. These results confirm that the framework is capable of handling larger datasets while maintaining predictable performance behavior.

From a security perspective, the Decentralized Integrity and Traceability Mechanism (DITM) demonstrated highly reliable operation. The blockchain-based logging mechanism recorded fragment operations with high accuracy, enabling effective detection of tampering attempts. The high success rate of transaction logging and verification indicates that the traceability layer can support forensic analysis and compliance auditing without introducing substantial computational overhead. This capability is particularly important in regulated environments where data access and modification events must be reliably recorded for auditing purposes.

The proposed framework also exhibits practical relevance for several data-intensive application domains. In healthcare environments, the ability to securely distribute sensitive patient data across multiple cloud providers helps maintain compliance with strict data protection regulations while preserving data availability. In financial systems, blockchain-based logging of access operations enhances transparency and strengthens the integrity of transaction records. Similarly, in IoT-enabled infrastructures, the lightweight fragmentation and encryption design allows edge devices

with limited resources to participate in secure distributed storage workflows.

To further understand the operational robustness of the framework, several system sensitivity factors were examined from a design perspective. These factors include the number of participating cloud service providers, variations in network latency, and the number of blockchain validator nodes involved in transaction verification. Increasing the number of cloud providers improves redundancy and reduces the likelihood of data exposure resulting from a single compromised provider. However, a larger number of storage endpoints may introduce coordination overhead during fragment retrieval. Similarly, network latency can influence fragment access time; nevertheless, the parallel retrieval strategy adopted by MC-SDFR mitigates this effect by allowing fragments to be fetched concurrently from multiple locations. For the blockchain traceability layer, increasing the number of validator nodes strengthens decentralization and trust guarantees, although consensus latency may increase slightly as additional nodes participate in transaction validation.

Table 7 summarizes the primary sensitivity and resilience considerations associated with the MC-SDFR architecture.

Table 7. Sensitivity and resilience factors in the MC-SDFR framework

Factor	System Impact	Expected Outcome
Number of CSPs	Determines fragment distribution and redundancy	Higher resilience to single-provider failure
Network latency	Influences the fragment retrieval delay	Parallel retrieval reduces performance degradation
Validator count	Affects blockchain consensus and audit latency	Higher decentralization with modest latency increase
CSP outage	Temporary unavailability of the storage node	Recovery through redundant fragment replicas
Fragment corruption or loss	Integrity validation failure	Replacement fragments retrieved from alternate providers
Key compromise	Potential exposure of encrypted fragments	Key rotation and fragment re-encryption required

Operational resilience was also considered through potential failure scenarios. In the event of a cloud service provider outage, the system reconstructs the original dataset by retrieving fragments from alternative providers where redundant replicas are maintained. Fragment corruption or loss is detected through integrity verification before recomposition, preventing invalid data from being incorporated into the reconstructed output. If corrupted fragments are detected, replacement fragments are retrieved from other storage locations to restore the dataset. These mechanisms enable the framework to maintain data availability and integrity even under partial system failures.

From a recovery perspective, the distributed storage architecture supports a low Recovery Time Objective (RTO) by enabling parallel retrieval of fragments from multiple storage nodes. In addition, the blockchain-based traceability mechanism ensures that fragment operations are persistently recorded, which helps maintain a near-zero Recovery Point Objective (RPO) since updates and access events can be reconstructed from the ledger. These design characteristics contribute to the reliability of the framework in practical multi-cloud deployments.

Despite these promising results, several limitations remain. The experimental evaluation was conducted primarily on a single local testbed, and the effect of scaling the number of blockchain validators and endorsement policies on transaction latency has not yet been fully investigated. Furthermore, large-scale benchmarking across multiple public cloud environments was outside the scope of the current study. Future work will therefore focus on extended evaluations involving multi-cloud deployments, dynamic network conditions, and larger blockchain networks to further quantify the trade-offs between decentralization, performance, and operational cost.

Overall, the results indicate that the MC-SDFR framework achieves a balanced combination of efficient data fragmentation, secure access control, and reliable traceability. By integrating adaptive fragmentation, multi-cloud authorization, and blockchain-based auditing, the proposed system provides a practical foundation for secure and scalable data management in distributed cloud infrastructures operating under diverse regulatory requirements.

V. Conclusion

This study presents MC-SDFR, a comprehensive and technically rigorous framework designed to address the critical shortcomings of conventional data fragmentation systems in multi-cloud settings. The research specifically targets the prevailing gaps in selective sensitivity handling, scalable encryption overhead, and compliance-anchored auditability, which have traditionally hindered the practical deployment of secure cloud storage systems. By integrating ASFT for dynamic sensitivity-aware fragmentation and DITM for blockchain-based traceability and validation, the proposed approach not only reduces the computational burden but also achieves deterministic data validation aligned with real-time access control policies. Empirical results affirm the effectiveness of this architecture, showing marked improvements in fragmentation/defragmentation speed, encryption/decryption efficiency, and overall system responsiveness. The 99.5% verification accuracy, reduced smart contract gas costs, and sustained low overhead ratios validate the scalability and operational feasibility of the solution. Moreover, ablation and comparative evaluations concretely demonstrate the independent and synergistic contributions of each module to the framework's performance. In conclusion, MC-SDFR sets a new standard for secure, compliant, and efficient data management in multi-cloud environments, delivering a practical solution that addresses the rigorous requirements of data sovereignty, regulatory adherence, and forensic traceability within expansive cloud infrastructures.

The proposed MC-SDFR framework integrates sensitivity-aware fragmentation, selective encryption, federated authorization, and permissioned-ledger traceability to provide a practical and low-overhead solution for secure multi-cloud data management. Experimental evaluation conducted on a Python 3.10 implementation running on an Intel-based system (2.6 GHz quad-core CPU, 8 GB RAM, Windows 10 64-bit) demonstrates consistent performance improvements across datasets of varying sizes. The framework achieved up to 36% faster defragmentation time

through deterministic fragment marker alignment and over 40% reduction in cryptographic processing latency by selectively encrypting high-sensitivity data blocks. Blockchain-based integrity verification through the DITM module recorded access and modification events with success rates between 99.7% and 99.9%, enabling reliable tamper detection and immutable audit logging. These results indicate that aligning fragmentation and encryption strategies with data sensitivity significantly reduces computational overhead while preserving strong confidentiality and traceability guarantees. Consequently, MC-SDFR provides an effective architecture for secure, scalable, and compliance-oriented data storage across heterogeneous multi-cloud and edge environments.

CONFLICT OF INTEREST

The authors declare no conflict of interest.

AUTHOR CONTRIBUTIONS

V.R.: Conceptualization—Developing the research idea or hypothesis, Methodology—Designing the study methods or protocols. J.M.K.: Investigation—Conducting experiments, surveys, or fieldwork, Writing—Original draft, writing the initial version of the manuscript. All authors had approved the final version.

ACKNOWLEDGMENT

The authors would like to thank the Deanship of Noorul Islam Center for Higher Education for supporting this work.

REFERENCES

[1] S. Kumari, "Cloud transformation for mobile products: Leveraging AI to automate infrastructure management, scalability, and cost efficiency," *Journal of Computational Intelligence and Robotics*, vol. 4, no. 1, pp. 130–151, 2024.

[2] S. C. Llobregat, "Design of a data analysis platform as a multitenant service in the cloud: An approach towards scalability and adaptability," *Universitat Politècnica de València*, 2024.

[3] M. M. Alshabibi, A. K. B. Dookhi and M. M. Hafizur Rahman, "Forensic investigation, challenges, and issues of cloud data: A systematic literature review," *Computers*, vol. 13, no. 8, 213, 2024.

[4] V. Ananthakrishna and C. S. Yadav, "Innovations in cloud security: Enhanced hybrid encryption approach with AuthPrivacyChain for enhanced scalability," *Nanotechnology Perceptions*, vol. 20, no. 52, pp. 560–577, 2024.

[5] H. A. Albaroodi and M. Anbar, "Security issues and weaknesses in blockchain cloud infrastructure: A review article," *Journal of Applied Data Sciences*, vol. 6, no. 1, pp. 155–177, 2025.

[6] D. R. Jeevaguntala, "Testing in multi-cloud environments: Ensuring consistency across distributed systems," *International Journal of Emerging Trends in Computer Science and Information Technology*, vol. 6, no. 2, pp. 29–35, 2025.

[7] L. Bouhouch, M. Zbakh, and C. Tadonki, "DFMCloudsim: An extension of CloudSim for modeling and simulation of data fragments migration over distributed data centers," *International Journal of Computers and Applications*, vol. 46, no. 1, pp. 1–20, 2024.

[8] S. D. Pasham, "Privacy-preserving data sharing in big data analytics: A distributed computing approach," *The Metascience*, vol. 1. no. 1, pp. 149–184, 2023.

[9] R. Siyal, J. Long, M. Asim *et al*., "Blockchain-enabled secure data sharing with honey encryption and DSNN-based key generation," *Mathematics*, vol. 12, no. 13, 1956, 2024.

[10] J. Merhej, H. Harb, A. Abouaissa, and L. Idoumghar, "Toward a new era of smart and secure healthcare information exchange systems: Combining blockchain and artificial intelligence," *Applied Sciences*, vol. 14, no. 19, 8808, 2024.

[11] N. Rathore, A. Kumari, M. Patel *et al*., "Synergy of AI and blockchain to secure electronic healthcare records," *Security and Privacy*, vol. 8, no. 1, e463, 2025.

[12] L. Chen, T. Feng, R. Ma, and J. Shi, "BTMDS: Blockchain trusted medical data sharing scheme with privacy protection and access control," *Computer Communications*, vol. 225, pp. 279–288, 2024.

[13] K. J. Merseedi and S. R. M. Zeebaree, "The cloud architectures for distributed multi-cloud computing: A review of hybrid and federated cloud environment," *The Indonesian Journal of Computer Science*, vol. 13, no. 2, 2024.

[14] E. Uzoma, J. O. Enyejo, and T. M. Olola, "A comprehensive review of multi-cloud distributed ledger integration for enhancing data integrity and transactional security," *International Journal of Innovative Science and Research Technology*, vol. 10, no. 3, 2025.

[15] S. F. Ahmed, S. S. Shawon, S. Afrin *et al.*, "Optimising Internet of Things (IoT) performance through cloud, fog and edge computing architecture," *IET Wireless Sensor Systems*, vol. 15, no. 1, e70016, 2025.

[16] R. M. Nabawy, M. Hassanin, M. H. Ibrahim *et al*., "Mixed fragmentation technique for securing structured data using multi-cloud environment (MFT-SSD)," *Ad Hoc Networks*, vol. 164, 103625, 2024. doi: 10.1016/j.adhoc.2024.103625

[17] N. Bharot, N. Mehta, J. G. Breslin, and P. Verma, "CloudLock: Secure data sharing using a hybrid cryptosystem in multi cloud data storage," *Cluster Computing*, vol. 28, no. 7, 464, 2025. doi: 10.1007/s10586-025-05433-7

[18] A. A. Khan *et al.*, "Blockchain enabled secure Internet of Medical Things (IoMT) architecture for precision cancer data management," *Journal of Cloud Computing*, vol. 14, no. 1, pp. 1–21, 2025. doi: 10.1186/s13677-025-00775-4

[19] S. N. Prasad and C. Rekha, "Block chain based IAS protocol to enhance security and privacy in cloud computing," *Measurement: Sensors*, vol. 28, 100813, 2023. doi: 10.1016/j.measen.2023.100813

[20] K. N. Mishra, R. K. Lal, P. N. Barwal, and A. Mishra, "Advancing data privacy in cloud storage: A novel multi-layer encoding framework," *Applied Sciences*, vol. 15, no. 13, 7485, 2025. doi: 10.3390/app15137485

[21] T. Ting and M. Li., "Enhanced secure storage and data privacy management system for big data based on multilayer model," *Scientific Reports*, vol. 15, no. 1, 2025. doi: 10.1038/s41598-025-16624-y

[22] K. Danach *et al.*, "Enhancing DDBMS performance through RFO-SVM optimized data fragmentation: A strategic approach to machine learning enhanced systems," *Applied Sciences*, vol. 14, no. 14, 6093, 2024. doi: 10.3390/app14146093

[23] P. B. Diyyana, "Enhancing cloud security and storage efficiency using a hybrid RSA ECC encryption framework with fragmentation techniques," *Journal of Modern Security & Research*, vol. 2, pp. 124–134, 2025.

[24] I. Hababeh, "A novel cloud computing data fragmentation service design for distributed systems," in *Proc. the International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA). The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp)*, 2011, vol. 1.

[25] H. Bodur and I. F. T. A. Yaseen, "An improved blockchain-based secure medical record sharing scheme," *Cluster Computing*, vol. 27, no. 6, pp. 7981–8000, 2024.

[26] R. Daruvuri, S. Chirumamilla, P. Mannem, and S. R. Aeniga, "Data privacy in multi-cloud: A fragmentation and distribution approach," *International Journal of Management, IT & Engineering*, vol. 14, no. 12, pp. 81–86, 2024.

[27] S. S. Hashmi, A. A. K. Mohammad, A. M. Abdul *et al*., "Enhancing data security in multi-cloud environments: A product cipher-based distributed steganography approach," *International Journal of Safety & Security Engineering*, vol. 14, no. 1, 2024.