Multiple Edge Data Integrity Verification With Multi-Vendors and Multi-Servers in Mobile Edge Computing

Md Rashedul Islam, *Member, IEEE*, Yong Xiang, *Senior Member, IEEE*, Md Palash Uddin, *Member, IEEE*, Yao Zhao, *Member, IEEE*, Jonathan Kua, *Member, IEEE*, and Longxiang Gao, *Senior Member, IEEE*

***Abstract*—Ensuring Edge Data Integrity (EDI) is imperative in providing reliable and low-latency services in mobile edge computing. Existing EDI schemes typically address single-vendor (App Vendor, AV) single-server (Edge Server, ES), single-vendor multi-server, and multi-vendor multi-server scenarios, which consider a single data replica cached by an ES from the AVs. However, the most practical scenario of Multi-Vendors and Multi-Servers with Multiple Data (MVMS-MD) cached by an ES from different AVs remains unexplored. Current solutions struggle when applied to this scenario due to increased computation and communication costs in the verification process across all ESs using the classical *challenge-response per-data multi-round* strategy. To tackle this issue, we propose a Multiple EDI-Verification (MEDI-V) approach in this paper. In particular, our MEDI-V utilizes an adaptive Merkle Hash Tree (ad-MHT) to efficiently generate a tree of multiple data replicas within each AV. Next, the dynamic mechanism computes minimal verification information using ad-MHT to create a *challenge* for individual ESs to produce EDI proofs. The ES then leverages its ad-MHT and the ES's proof to send the reconstructed ad-MHT root to the AV for verification. Theoretical insights into MEDI-V's correctness, efficiency, security, and comprehensive evaluations demonstrate its superiority in addressing MEDI issues in the MVMS-MD scenario.**

***Index Terms*—Adaptive merkle tree, data integrity verification, edge computing, edge data integrity, minimal verification information, single-round proof generation.**

I. Introduction

THE surge in edge devices, such as smartphones/mobile devices, Internet of Things (IoT) devices, and autonomous vehicles, has surpassed the growth witnessed in traditional computing systems [1]. According to recent statistics, the number of edge devices globally is projected to reach 17.08 billion in 2024 [2]. However, new challenges arise with the rapid surge in edge devices, such as the efficiency and dependability of data processing. Over the past decade, cloud computing has emerged as a solution to address the growing need for computing and storage capacity. Cloud services promise scalable and reliable options by offering centralized data storage and processing capabilities [3], [4]. Despite its advantages, cloud computing has its limitations when the number of edge devices increases, particularly in achieving prompt and efficient communication [5].

Therefore, Mobile Edge Computing (MEC) has emerged as a paradigm shift in computing architecture in recent years to deal with these issues [6]. In particular, MEC has been introduced as a transformative solution to overcome the latency issues of traditional cloud-centric architectures and minimize communication delays by bringing computation closer to the data source [7]. This shift from centralized cloud servers to distributed edge devices is particularly advantageous for applications that require quick responses, such as augmented reality, autonomous vehicles, and industrial automation [8], [9]. Despite the benefits of MEC, a significant complexity arises in managing edge data. Unlike traditional cloud environments where a cloud App Vendor (AV) controls the stored data, the decentralized nature of edge infrastructure raises concerns about data control and verification [10]. This underscores the need to thoroughly examine edge data verification mechanisms to ensure the integrity of the processed cache data at the Edge Server (ES) [11]. Ensuring Edge Data Integrity (EDI) is critical not only for technical accuracy but also for its significant impact on society, human development, and economic stability. Reliable data verification supports accurate decision-making in essential sectors like healthcare, finance, and energy, thereby fostering trust in digital systems and protecting against potential risks to public safety and economic stability. This motivates an in-depth examination of the EDI problem, where the difficulty lies in ensuring the reliability of data replicas stored on ESs, which is a critical consideration for AVs navigating the intricacies of MEC [11].

Existing EDI schemes have been designed to accommodate three distinct scenarios: (a) Single-vendor single-server,

Received 23 August 2024; revised 9 November 2024; accepted 2 January 2025. Date of publication 10 January 2025; date of current version 7 May 2025. This work was supported in part by Australian Research Council under Grant LP190100594. Recommended for acceptance by X. Peng. *(Corresponding author: Yong Xiang.)*

Md Rashedul Islam, Yong Xiang, Md Palash Uddin, Yao Zhao, and Jonathan Kua are with the School of Information Technology, Deakin University, Geelong, VIC 3220, Australia (e-mail: md.islam@deakin.edu.au; yong.xiang@deakin.edu.au; m.uddin@deakin.edu.au; cnr@deakin.edu.au; jonathan.kua@deakin.edu.au).

Longxiang Gao is with the Key Laboratory of Computing Power Network and Information Security, Ministry of Education, Shandong Computer Science Center, Qilu University of Technology (Shandong Academy of Sciences), Jinan 250316, China, and also with the Shandong Provincial Key Laboratory of Computer Networks, Shandong Fundamental Research Center for Computer Science, Jinan 250014, China (e-mail: gaolx@sdas.org).

This article has supplementary downloadable material available at https://doi.org/10.1109/TMC.2025.3528016, provided by the authors.

Digital Object Identifier 10.1109/TMC.2025.3528016

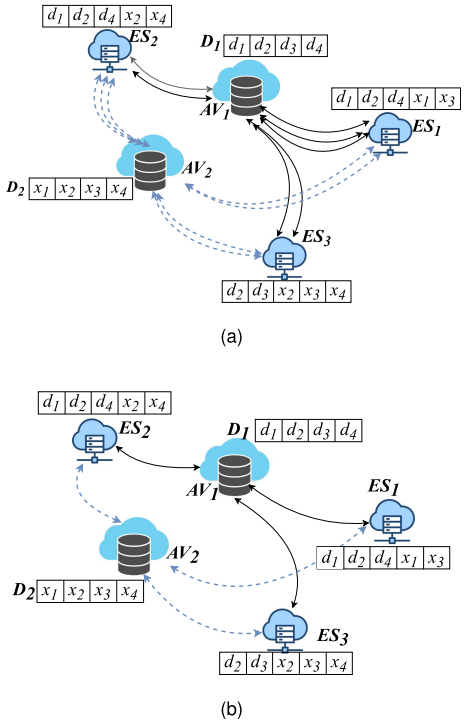


Fig. 1. The challenge-response mechanism: (a) existing per-data multiple rounds approach; (b) proposed per-edge single round approach.

where an ES caches a single data replica d from a single AV [12]; (b) Single-vendor multi-server, where multiple ESs cache a single data replica d from a single AV [13], [14]; and (c) multi-vendors multi-servers, where multiple ESs cache the same single data replica d from multiple AVs [15]. However, the most practical scenario in the real world, i.e., Multi-Vendors and Multi-Servers with Multiple Data (MVMS-MD), where an individual ES caches multiple data from multiple AVs, remains largely unexplored.

The detailed literature review provided in Section VIII highlights the difficulties of existing solutions in extending their applicability to address our identified MVMS-MD scenario. These challenges underscore the need for novel approaches tailored specifically to handle the complexities of multi-vendors and multi-server scenarios with multiple data replicas cached by an ES. Most importantly, the difficulty arises primarily due to the uniform verification process for each data replica of individual AVs across all ESs, which employs a *challenge-response per-data multi-round* approach [11]. Consequently, extending the existing methods to verify the Multiple EDI (MEDI) under the MVMS-MD scenario reveals a notable issue. Specifically, each AV must undergo the same verification process with all corresponding ESs using the *challenge-response* mechanism. For instance, Fig. 1(a) represents the existing classical *challenge-response* mechanism and Fig. 1(b) represents the proposed *challenge-response* mechanism. It can be seen that there is a difference in the number of *challenge-responses* between Fig. 1(a) and (b). Each data item cached by the ES requires a separate challenge and corresponding response from the AV. This means that for n data replicas, n *challenge-response* interactions are needed. If the ES caches m data replicas from k AVs, the total number of interactions increases significantly, contributing to higher communication costs. As the number of cached data items grows, the volume of communication between the AVs and the ES increases linearly. For large-scale deployments where ESs cache data from numerous AVs, this results in high communication costs. Moreover, each *challenge-response* interaction typically involves executing a cryptographic hash function to verify data integrity. Repeatedly performing this hash function for each data item exacerbates computation costs. For instance, if a hash function is executed n times for n data replicas and each execution takes a certain amount of processing time, the total computation load can be considerable, especially under high data loads. Furthermore, while caching at the ES provides reduced latency and improved bandwidth utilization in MEC, these traditional *per-data multiple-round* verification approaches are unsuitable for environments with a wide range of AVs, necessitating a *per-edge single-round* verification approach.

To address the issues of MEDI under the identified MVMS-MD scenario, we propose a novel approach termed MEDI-Verification (MEDI-V), which effectively mitigates computation and communication costs. Specifically, the proposed MEDI-V approach uses a per-edge single-round *challenge-response* mechanism that requires only a single round to verify all data replicas at an ES caching from multiple AVs. This significantly reduces the number of interactions and associated computation costs. In particular, our proposed MEDI-V approach first introduces an adaptive Merkle Hash Tree (ad-MHT) to efficiently generate a tree of multiple data replicas within an individual AV. Compared to conventional Merkle Hash Trees (MHTs) like binary or ternary trees, ad-MHT offers substantial advantages. Its suitability stems from the ability to handle the dynamic nature of data cached by ESs in real-world deployments. These servers may cache any number of replicas, making ad-MHT the optimal choice. It then invokes a dynamic mechanism to compute Minimal Verification Information (MVI) using the ad-MHT, generating minimal auxiliary information to create a *challenge* for the individual ES to produce the EDI proof within the ES. MVI is necessary to reconstruct the ad-MHT root as proof of data replicas because the ES may not cache all the data replicas from the AVs. The ES then *responds* with the reconstructed ad-MHT root to the AV for verification. Finally, the individual AV performs the verification using its ad-MHT and the ES's generated proof. Note that, in the proposed method, each AV operates independently and conducts its own verification process. For example, in Fig. 3, ES_1 caches data replica d_1 from AV_1 and d_2 and d_3 from AV_2. During the verification process, AV_1 can only verify d_1, while AV_2 can verify d_2 and d_3. Thus, each AV verifies its data replicas without relying on others. In contrast, in Fig. 2(c), an ES can cache the same data replica from multiple AVs. Since the data is identical, any AV can verify its integrity, leading to a lack of independence among the AVs in this scenario. We provide theoretical insights into our MEDI-V approach with security dimensions and thoroughly evaluate the proposed scheme in terms of MEDI verification accuracy and efficiency (i.e., computation and communication costs).

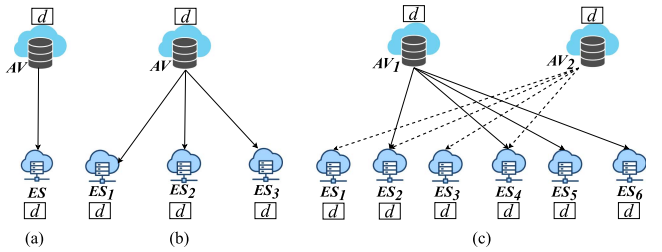


Fig. 2. Existing MEC scenarios: (a) Single-vendor (AV) single-server (ES) caches a single data replica d; (b) Single-vendor (AV) multi-server (ES_1, ES_2, and ES_3) caches a single data replica d; and (c) Multi-vendors (AV_1 and AV_2) multi-servers ($ES_1, ES_2, ES_3, ES_4, ES_5$, and ES_6) cache the same single data replica d (e.g., ES_2 and ES_4 cache the same data replica d from both AV_1 and AV_2.

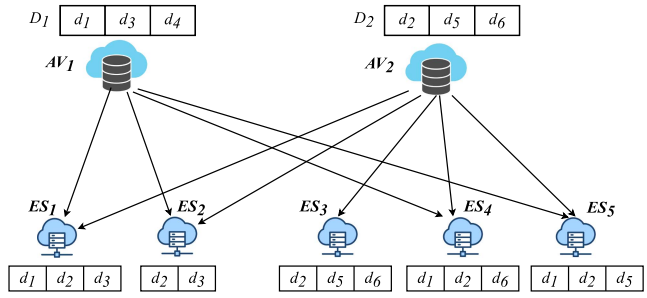


Fig. 3. An illustration of the identified MVMS-MD MEC scenario. MVMS-MD MEC is a real-world practical demonstration of a cloud-edge data caching mechanism where multiple ESs ($ES_1, ES_2, ES_3, ES_4$and ES_5) cache multiple data ($d_1, d_2, d_3, d_4$and d_5) from multiple AVs ($AV_1$and AV_2).

The main contributions of this paper are summarized below.

- We propose a MEDI-V approach for MEDI verification using a single-round approach in a real-world practical MVMS-M scenario that features multi-vendors multi-servers, where each ES caches multiple data replicas from different AVs.
- We introduce a novel mechanism for building an ad-MHT to handle multiple data replicas within individual AVs which can be substantially faster compared to conventional MHT (12.42% to 36.34% faster in our work).
- We present a dynamic mechanism to compute MVI for minimal auxiliary information generation, required to reconstruct the ad-MHT root at the ES for proof generation.
- We provide a theoretical analysis for MEDI-V regarding its correctness and conduct a thorough experimental analysis to demonstrate its efficiency. From our experiments, it is evident that the communication cost of MEDI-V is 4.33×10^4 to 4.12×10^5 times faster than existing methods.

The remainder of the paper is organized as follows. Section II discusses the overview of the MVMS-MD integrity verification problem. Section III presents the core methodology of the proposed MEDI-V scheme, while Section IV describes and outlines the details of the proposed ad-MHT creation and dynamic computation of MVI. Section V discusses the details of the MEDI-V approach. Section VI provides the theoretical analysis of MEDI-V's correctness. Section VII experimentally evaluates MEDI-V against the potential existing approaches. Section VIII reviews the related work. Finally, Section IX summarizes this paper and points out some future works.

TABLE I
KEY SYMBOLS AND NOTATIONS

Notation	Meanings
D	The original data
ES_i	i^{th} ES
AV_i	i^{th} AV
MAV	AV Group
MES	ES Group
d_i	i^th data replica in D
$Hash()$	Specified hash function
H_p^l	The representation of leaf or intermediate nodes
l	The level of ad-MHT tree
Eid_i	The id of the respected ES
Aid_i	The id of the respected ES
rs	The random seed
m	The number of AVs
n	The number of Edge Node
k	The number of data replica
ds	The size of each data replica
ss	Sampling scale in each audit round
cr	Corrupted data Ratio contain in D

TABLE II
ACRONYMS

Acronym	Definition
AV	Application (App) Vendor
ES	Edge Server
EDI	Edge Data Integrity
MVMS-MD	Multivendor Multiserver with Multiple Data MEC Scenario
MEC	Mobile Edge Computing
MHT	Merkle Hash Tree
ad-MHT	Adaptive MHT
MAV	Multiple AV
MES	Multiple ES
MEDI	Multiple EDI
MEDI-V	Proposed Framework (Multiple EDI Verification)
MVI	Minimum Verification Information

II. MVMS-MD INTEGRITY VERIFICATION PROBLEM OVERVIEW

We assume that communication within the MEC environment takes place through secure channels without additional delays, which is an assumption commonly made in existing literature [13], [15], [16]. In this section, we present an overview of the MVMS-MD scenario, a motivational example, and an overview of our problem model. Table I presents key notations and Table III represents the acronyms used throughout this article.

A. MVMS-MD Scenario Overview

In the MVMS-MD scenario, the complexity of verifying the integrity of multiple AVs' different data compared to single data from single or multiple AVs (as depicted in Fig. 2) at the ESs is notably different. In particular, the authors of the existing MEC scenario in Fig. 2(c) considered that AVs outsource a single data replica to multiple ESs. It is seen in Fig. 2(c) that six different ESs ($ES_1, ES_2, ES_3, ES_4, ES_5$, and ES_6) cache the same data d from two different AVs ($AV_1$and AV_2). On the other hand, in our proposed MVMS-MD scenario, each ES can cache multiple data replicas from different AVs. As shown in Fig. 3, five different ESs (ES_1, ES_2, ES_3, ES_4, and ES_5) cache multiple data (d_1, d_2, d_3, d_4, and d_5) from two different AVs (AV_1 and AV_2). As an instance, ES_1 caches d_1 from AV_1 and d_2 and d_3 from AV_2. In this way, we can establish that each existing scenario in Fig. 2 is a special case of our more generic MVMS-MD scenario (Fig. 3). In particular, when the number of AVs, ESs, and data replicas in the MVMS-MD scenario is all one, it aligns with the single AV single ES scenario, as illustrated

in Fig. 2(a). Next, when the number of AVs and data replicas in the MVMS-MD scenario is all one but the number of ESs is more than one, it is compatible with the single AV multiple ESs scenario, as depicted in Fig. 2(b). Finally, when the number of AVs and ESs in the MVMS-MD scenario is more than one but the number of data replicas is single and the same, it aligns with the MVMS-SD scenario, as in Fig. 2(c).

B. Motivational Example

In the MVMS-MD MEC environment, the collaboration between AVs and ESs creates a dynamic scenario where multiple data replicas from different AVs are cached by an ES to cater to individual user needs, as depicted in Fig. 3. Unlike traditional setups with single-vendor single-server, single-vendor multi-servers, and multi-vendor multi-server with single data, this real-world setting (MVMS-MD) poses additional difficulties as it requires maintaining MEDI. To better illustrate this scenario, consider a user who relies on MEC services that seamlessly cache data from various AVs, such as AWS, Google, Amazon, etc., to optimize their application experiences. The user's preferences and requirements dictate the caching decisions made by the ESs, which results in a different array of cached data replicas.

The convergence of diverse applications creates opportunities to optimize network performance. For instance, imagine a mobile content aggregation app utilizing data from news articles, social media updates, and personalized recommendations. MEC infrastructure could cache frequently accessed data from different AVs and consequently reduce the latency. Similarly, in a location-based service integrating data from mapping, navigation, and local business recommendations, MEC could cache relevant data to enhance the user experience. Additionally, in IoT applications such as smart city deployments, MEC could cache data from various AVs responsible for city management, and enable real-time analysis and decision-making at the edge.

Ensuring EDI becomes a daunting task as the volume and variety of data from multiple AVs grows rapidly. The difficulty lies not only in verifying each cached data replica but also in developing efficient mechanisms that can adapt to the ever-changing landscape of AVs and user preferences. Addressing this scenario requires innovative solutions that go beyond conventional approaches presented for single-vendor single-server, single-vendor multi-servers, or multi-vendors multi-servers with only a single data cached by an ES from an AV, hence acknowledging the complexity introduced by the coexistence of multiple AVs and ESs with caching multiple data from the AVs. Our work endeavors to address the intricacies of these difficulties and aims to provide insights and methodologies to enhance MEDI verification in MVMS-MD MEC scenarios.

C. Model Formulation

In our proposed MVMS-MD scenario, the MEC system consists of two discrete entities that are (i) m multiple AVs, denoted as MAV, and (ii) n multiple ESs, represented as MES. Each AV has multiple data replicas, denoted as MD.

Definition 1. (Multiple AVs, MAV): An MAV group $MAV = \{AV_1, AV_2, \ldots, AV_m\}$ consists of $m(m \geq 2)$ different AVs, where

- Each AV $AV_i \in MAV$ has cached its original data replicas D_i on ESs, where $D_i = \{d_1, d_2, \ldots, d_k\} \in MD$.
- Each AV can cache data on a set of q ESs out of n ESs: $MES = \{ES_1, ES_2, \ldots, ES_n\}$, where $q \geq 1$.
- Each AV is responsible for creating an ad-MHT (discussed in Section IV-B) initially and initiating verification challenges by creating MVI (discussed in Section IV-C).

Definition 2. (Multiple ESs, MES): An MES group $MES = \{ES_1, ES_2, \ldots, ES_n\}$ consists of n $(n > 2)$ different ESs, where

- Each ES can cache the data from a set of MAVs. Each ES caches all or a subset of data from each AV.
- We presuppose the independent operation of each ES, with no collaboration or collusion among them, similar assumption found in many works, e.g. [13], [17]. Upon receiving a verification request from the AV, each ES is tasked with showing evidence that attests to the integrity of its cached data.

Definition 3. (The MEDI-V Problem in the MVMS-MD Scenario): In the context of the MVMS-MD scenario, the MEDI-V problem entails a complex MEC mechanism, where an ES within the MES caches diverse data from the AVs of the MAV. As illustrated in Fig. 3, two AVs, AV_1 and AV_2, cache multiple different data replicas in their datasets, i.e., $D_1 = \{d_1, d_3, d_4\}$, and $D_2 = \{d_2, d_5, d_6\}$, respectively. Different ESs then cache different data from different AVs, e.g., ES_1 caches d_1 and d_3 from AV_1 and d_2 from AV_2, etc. As such, MEDI-V needs to be achieved through diverse mechanisms and algorithms implemented at both the AVs and the ESs under this MVMS-MD MEC scenario.

D. Threat Model

We considered the AVs to be honest and trustworthy,[1] and all ESs operate independently without collusion while acknowledging the possibility of fraudulent behavior from individual ESs. In our work, we focus on replay attacks, replace attacks, and forgery attacks, which are relevant to edge data integrity verification within the proposed framework. Other attacks such as spoofing [18], data leakage [19], outsourcing, Byzantine, and collusion [20] are not considered due to their misalignment with our model's assumptions. Specifically, we exclude spoofing and

[1]This assumption is well-grounded and aligns with the literature (e.g., [13], [14], [15], [17], [22], and [33]). The collated reasons are explained as follows. AVs are responsible for maintaining data integrity and ensuring their contents' authenticity. Given that AVs have a vested interest in safeguarding their reputation and credibility, they are generally seen as having little incentive to engage in malicious activity. If AVs are malicious, our MEDI-V approach—and the aforementioned related works—would be ineffective, as a dishonest AV could manipulate verification results, overlook or even promote data corruption, and collaborate with external attackers. A potential extension to address this risk could involve incorporating blockchain-based verification mechanisms, enabling decentralized trust without relying on any single AV. This approach would allow all parties to verify data integrity, even in cases of dishonest AVs. However, integrating blockchain methods would notably increase computational costs and diverge from the primary focus of our work.

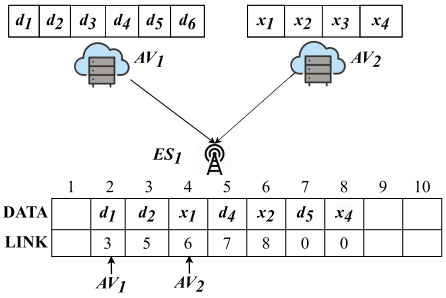


Fig. 4. Data tracking on the ES side.

data leakage attacks because we assume honest AVs and TPAs are not involved. Outsourcing and collusion attacks are omitted as our model assumes no interconnection among ESs. Additionally, we assume no Byzantine attacks [20] exist for simplicity; similar assumptions hold in [21]. Our analysis encompasses both intentional and unintentional threats as follows:

Replay Attacks: The replay attack aims to deceive the AVs into accepting outdated or forged integrity proofs, resulting in the false verification of data integrity. This attack can compromise the reliability of the data integrity verification process, allowing the ES to present misleading evidence that the data replicas are intact when, in reality, they might have been altered or compromised.

Replace Attacks: A replacement attack initiated by an ES in response to an AV's request for data integrity verification involves the malicious substitution of authentic data with unauthorized or compromised information. In this case, the adversarial ES intercepts the AV's request for data integrity verification and strategically replaces genuine integrity proofs with manipulated or fraudulent counterparts.

Forgery Attacks: Dishonest ESs may forge integrity proofs of cached data before sending it to the AV, compromising data integrity and deceiving the vendor's integrity auditing process.

E. Design Goals

Given the outlined MVMS-MD scenario and thread model, our proposed MEDI-V approach should meet the following design objectives.

Correctness: MEDI-V ensures that AVs are capable of accurately verifying the integrity of data replicas at ESs through verification algorithms.

Efficiency: MEDI-V prioritizes efficient time complexity and minimal communication cost, catering to resource-constrained ESs and AV demands, ensuring a balanced approach.

Security: MEDI-V should thwart any attempts by dishonest ESs to engage in replay attacks, replace attacks, or forgery attacks.

III. Key Techniques

A. Cross-Edge Batching

In the context of the MVMS-MD MEC scenario, MEDI-V facilitates the consolidation of locally cached data from various AVs for each ES, as illustrated in Fig. 4. This enables AVs to perform a comprehensive integrity verification of the batch in a single round. As shown in Fig. 4, ES_1 hosts a batch that includes data replicas d_1, d_2, d_4, d_5 and d_6 from AV_1, and x_1, x_2, x_4 from AV_2. This process results in the formation of cross-edge-specific batches, ensuring that each batch contains only the data pertinent to the respective ES and AV.

B. MHT-Based Authentication

In a conventional setting, a single AV may be associated with either a single ES or multiple ESs. In contrast, our MVMS-MD scenario involves multiple AVs and multiple ESs, each caching different data from the various AVs. Consequently, the verification process for such MEDI generally necessitates multiple rounds of data verification. However, in our proposed MEDI-V approach, an ad-MHT is constructed by each AV that provides an efficient mechanism in a single round of verification.

C. Data Tracking in AVs and ESs

Each AV maintains an index list of the original data for each ES to track which data is locally cached by each ES. To better illustrate our innovative linked list-based coordination mechanism, consider Fig. 4. It is observed that ES_1 caches data replicas d_1, d_2, d_4, and d_5 from AV_1 and x_1, x_2, and x_3 from AV_2. During integrity verification, AV_1 is responsible for independently verifying d_1, d_2, d_4, and d_5, while AV_2 verifies x_1, x_2, and x_4. AV_1 cannot verify the data replicas from AV_2 (x_1, x_2, and x_4), and vice versa. To facilitate such independent data integrity verification, our linked list-based coordination mechanism enables each AV to maintain index lists of the original data for each ES, allowing them to track which data is locally cached by each ES. As shown in Fig. 4, AV_1 has the original data list $D = d_1, d_2, d_3, d_4, d_5, d_6$, and AV_2 has the original data list $X = x_1, x_2, x_3, x_4$. ES_1 caches the data d_1, d_2, d_4, and d_5 from AV_1 and x_1, x_2, and x_4 from AV_2. Consequently, AV_1 creates the index list $ES_1 = [1, 2, 4, 5]$, and AV_2 creates the linked list $ES_1 = [1, 2, 4]$ to conduct the integrity verification independently. On the ES side, to track data from multiple AVs, each ES maintains a single linked list and stores the starting pointers of individual AV list pointers, as depicted in Fig. 4. For example, ES_1 contains data d_1, d_2, d_4, and d_5 from AV_1, and x_1, x_2, and x_4 from AV_2. Therefore, ES_1 creates a single linked list and maintains two pointers to track the data from different AVs.

D. MVI Computation

To minimize computation and communication costs, MVI is computed to reconstruct the ad-MHT. MVI refers to a set of essential details required by the AV when issuing a verification challenge to the ES. Each ES, in turn, requires the MVI to reconstruct the adaptive tree root. The concept of MVI ensures that the verification process is carried out efficiently and with minimal resource consumption. It involves providing only the necessary information needed for the ES to reconstruct the ad-MHT's root to verify the data's integrity. This approach minimizes the load on both the AV and the ES, making the verification process more agile and resource-effective.

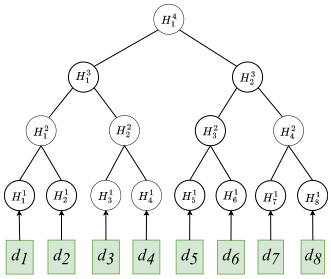


Fig. 5. An instance of the traditional binary MHT.

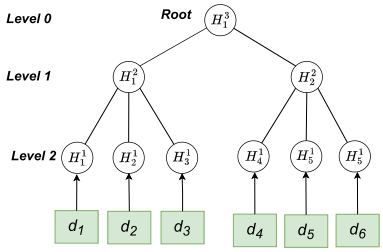


Fig. 6. An instance of the proposed ad-MHT.

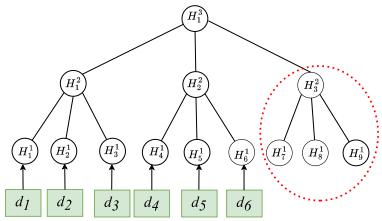


Fig. 7. An example of ternary MHT.

E. Single Round Verification Process

Multiple AVs request integrity checks for each ES in a single-round verification process. Conversely, each ES receives requests from multiple AVs due to its caching of data replicas from various sources. Each AV generates a singular request to each ES, including the generated MVI. In response, each ES provides the ad-MHT root, reconstructed by MVI, and locally cached data. AVs then verify data integrity by comparing ad-MHT roots from both the initiating AV and responding ESs in a single round. This streamlined process ensures efficient verification of cached data integrity across multiple ESs.

IV. Adaptation of MHT and Dynamic Computation of MVI

A. Limitation of Traditional MHT

The conventional MHT is a binary tree created based on a set of data replicas. As illustrated in Fig. 5, given a dataset comprising eight data replicas, the corresponding MHT comprises eight leaf nodes. Each leaf node represents an individual data replica and stores its corresponding hash value. Pairwise, every two nodes form a parent node, such as nodes H_1^1 and H_2^1 combining to create node H_1^2 as their parent. This construction process is iteratively applied until the root node of the MHT is formed, as depicted by H_1^4 in Fig. 5.

In practical cases, the quantity of data replicas frequently deviates from being an exact power of 2. Moreover, the height of the MHT grows proportionally with the augmentation of data replicas. Consequently, constructing MHT roots becomes more time-intensive with an increasing number of data replicas, given the necessity to generate parent nodes through child hash values. An alternative approach to this issue involves adopting a ternary MHT rather than the traditional binary structure. In ternary MHT, each parent node accommodates three children instead of two. This modification allows for a more flexible and efficient representation of data replicas, potentially mitigating the time-consuming nature of constructing MHT roots when the number of data replicas is not an exact power of 2. On the other hand, a quaternary tree, where each parent node accommodates four children, or a tree structure involving an even greater number of branches, like five or six children, is expected to be more intricate than binary and ternary MHTs. However, this heightened complexity poses issues during the creation and management of the MHT, potentially offsetting the advantages gained from the adaptability of the ternary structure. While the ternary MHT presents a promising solution to the limitations associated with binary MHTs, it introduces difficulties when creating an additional branch in each parent node, differentiating it from the binary counterpart. For instance, consider a ternary tree as shown in Fig. 7. If the number of data replicas is 6, using a ternary MHT initially creates an extra branch, which adds complexity when calculating the root on the server side. While this extra branch contributes to the adaptability of the ternary MHT, and can accommodate a larger range of data quantities, it also brings about an initial computation cost. In scenarios where the dataset is sparse or comprises only a few data replicas, this additional branch may lead to unnecessary complexity and resource consumption during the initial construction of the MHT. To address these aforementioned problems, we introduced an ad-MHT, which is constructed based on the number of leaf nodes (data replicas). The details of the construction of ad-MHT are discussed below.

B. ad-MHT

Adaptive Merkle Hash Tree (ad-MHT) can be described as a data structure in which each non-leaf node can have either three or two children, depending on the availability of leaf nodes or intermediate nodes on each level. The tree's construction follows a bottom-up approach, dynamically adjusting the number of children for each non-leaf node based on the specific configuration of the available nodes in the structure. The adaptability of the MHT allows for efficient representation and verification of data integrity, making it particularly useful in scenarios where the number of elements or nodes may vary dynamically. ad-MHT is constructed by considering the number of available leaf nodes or intermediate nodes (r) at each level. The construction follows the following rules while the detailed pseudo-code of the ad-MHT construction is illustrated in Algorithm 1.

- We consider three cases: (i) If the number of nodes (r) is greater than twice the maximum number of children ($2n$),

where $n = 3$, or (ii) if r is equal to 3, or (iii) if r is less than twice the maximum number of children but greater than 3, and the modulo of r is greater than or equal to 2, when creating an ad-MHT using a ternary structure where Parent = hash(x, y, z), and x, y, and z are the siblings of ad-MHT.

- Otherwise, for each set of three leaf nodes or intermediate nodes x and y, a parent node is created using two children (binary) as Parent=hash(x, y).

ad-MHT is a structured tree with a specific order. Each node within this tree includes two sequential numbers denoting its level (l) and position (p) within that level. In the context of an ad-MHT, the level (l) starts at 1 and increments in a bottom-up manner. Consequently, the leaf node is assigned a level of 1, and its parents are assigned levels 2in increasing order, such as 2 for the immediate parent, and so on. Nodes situated at the same level are arranged in order from left to right, with the leftmost node on any level being assigned a position of 1 (p=1). Subsequent nodes on the same level are assigned sequential positions, with the second node having a position of 2, and so forth. A node at level l with a position p is represented as H_p^l.

Fig. 6 displays a 3-level ad-MHT featuring 9 nodes created from edge data containing 6 data replicas. Each node at level 3 (l=2) holds the hash value of its corresponding data replica (e.g., node H_3^1 holds the hash value of data block d_3). Non-leaf nodes store the hash values of all their child nodes (e.g., node H_1^2 holds the hash values of nodes H_1^1, H_1^2, and H_1^3). ad-MHT begins by establishing 6 leaf nodes at level 3 (l=2). These leaf nodes serve as the foundation for creating two intermediary parent nodes (H_2^1 and H_2^2) utilizing the ternary MHT structure. Moving to level 2 (l=1), where only 2 nodes are available, ad-MHT constructs its root (H_3^1) using a binary MHT structure.

C. MVI for ad-MHT Reconstruction

MVI plays a crucial role in providing additional contextual information to each ES individually during the MEDI verification process. This supplementary information is provided by the AV to the ES for reconstructing the ad-MHT root. Since multiple data replicas originate from different AVs, there is a possibility of some data replicas being absent in the ES. To construct the ad-MHT root in the ES and generate proof for data integrity verification, it is necessary to transfer the missing data replicas to the ES. In particular, for a given ES denoted as ES_i, the AV formulates MVI by leveraging the record index list derived from the data cached by ES_i and its initial ad-MHT. This proactive generation of MVI precedes the transmission of an integrity investigation request to ES_i, ensuring that the ES can effectively utilize this supplemental information for constructing the ad-MHT root based on its local dataset. The term "minimal" in MVI is aptly chosen because it signifies the creation of the minimum information required for reconstructing the ad-MHT. For instance, as illustrated in Fig. 8(a), local caching by any ES_i includes data elements d_1, d_2, d_4, and d_5. To properly reconstruct the ad-MHT, ES_i requires information about the non-cached data elements (H_3^1 and H_6^1), which is supplied by the AV. In contrast, as depicted in Fig. 8(b), if an ES_j caches x_1, x_2, and x_3, the non-cached data elements x_4, x_5, and x_6 need to be sent by the AV to the ES. However, the proposed MVI simplifies this process by requiring only H_2^2 to be sent by the AV for the accurate reconstruction of the ad-MHT. Consequently, the term "minimal verification information" aptly characterizes this streamlined approach, emphasizing efficiency and precision in the MEDI verification process. The pseudocode for constructing MVI is given in Algorithm 2. The procedure details and data structure at each step are illustrated in Fig. 9. Next, we outline the main steps of Algorithm 2. Initially, the cache data nodes list is $Data = \{H_1^1, H_2^1, H_3^1\}$, the $MVI(A_i)$ and $Stack$ are empty. The following steps are recursively run until there is no element (node) in the stack $Stack$.

- The root node is pushed into the $Stack$.

Algorithm 1: Construction of the ad-MHT.

Require: The original data list, D.
Ensure: The ad-MHT root node.
procedure *Construct_ad-MHTD*
  function HASHDATA($data$)
    return $hash(data)$ ▷ Hash function for the data
  end Function
  Initialize an empty queue, Q.
  $Q.enqueue$(HASHDATA(D)) ▷ Insert hashed data into Q
  while $Q.Size() > 1$ **do**
    $r \leftarrow Q.Size()$
    while $r > 1$ **do**
      if ($r > 3$ **and** $r\%3 = 2$) **or** $r > 6$ **or** $r = 3$ **then**
        $x \leftarrow Q.dequeue()$ ▷ 1st node in current Q
        $y \leftarrow Q.dequeue()$ ▷ 2nd node in current Q
        $z \leftarrow Q.dequeue()$ ▷ 3rd node in current Q
        $Q.enqueue$(HASHDATA($\{x, y, z\}$))
        $r \leftarrow r - 3$
      else
        $x \leftarrow Q.dequeue()$ ▷ 1st node in current Q
        $y \leftarrow Q.dequeue()$ ▷ 2nd node in current Q
        $Q.enqueue$(HASHDATA($\{x, y\}$))
        $r \leftarrow r - 2$
      end if
    end while
  end while
  $root \leftarrow Q.dequeue()$ ▷ The root node of ad-MHT
end procedure

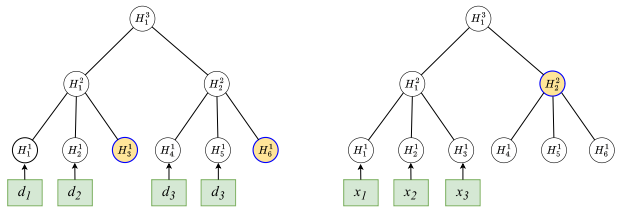


Fig. 8. The ad-MHT for creating MVI (a) ES caches d_1, d_2, d_4, and d_5 data (b) ES caches x_1, x_2, and x_3 data replicas.

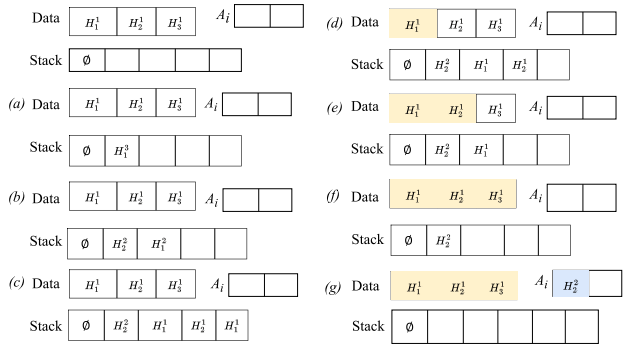


Fig. 9. Construction of MVI: Initially, the leaf nodes list $Data = \{H_1^1, H_2^1, H_3^1\}$ of D_i. We then create an empty stack, S, and the corresponding empty MVI, A_i. After applying Algorithm 2, all initial nodes in *Data* are traversed at the end of step (g), and the final MVI becomes $A_i = \{H_2^2\}$.

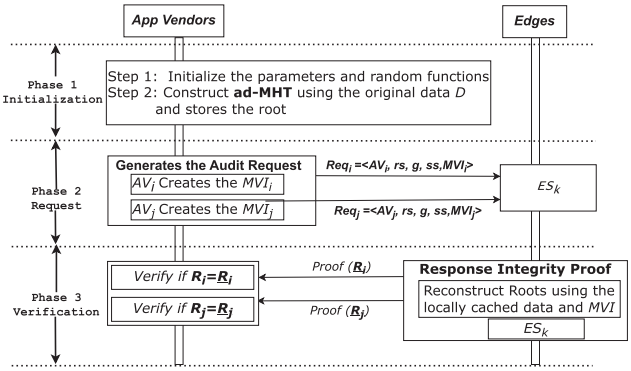


Fig. 10. The proposed MEDI-V framework.

- A node is popped from the $Stack$. For the popped node, the ad-MHT's left node, middle node, and right node are traversed (post-order tree traversal). These nodes are pushed onto the $Stack$ until a leaf node appears in the $Stack$.
- Nodes are continuously popped from the $Stack$ and searched for in the $Data$ list. If found, the node is marked as traversed; otherwise, it is pushed into A_i as an MVI corresponding to lines 11 through 25 of Algorithm 2 and Fig. 9(d) and (e).

V. MEDI-V Meta-Approach

In our entire proposed MEDI-V approach, the AV utilizes a *challenge-response* mechanism to verify the integrity of its edge data replicas. This evaluation can be conducted periodically or initiated as needed. The Proposed framework is illustrated in Algorithm 3 and depicted in Fig. 10 that consists of three main phases. In the initial phase, the AVs generate the necessary parameters, calculate data replica hashes, and construct the ad-MHT for each ES. In the next phase, the AV initiates the audit request by creating the MVI for each ES and utilizing the generated parameters. Following this phase, the verification takes place between the AVs and ESs. Below we describe the three phases in detail.

Phase 1: Initialization

Algorithm 2: Construction of the MVI.

Require: Cache data node list $Data$, $ad\text{-}MHT$.
Ensure: MVI $\leftarrow A_i$.
1: **procedure** $Create\_MVIData, ad\text{-}MHT$ ▷ IsLeafNode is a function to check if any node is a leaf node or not.
2: **function** IsLeafNode$node$
3: **return** node.level== 1 ? True: False
4: **end Function**
5: ▷ CheckInData is a function to check a node belonging to already cache $data$.
6: **function** CheckInData($Data$, $node$)
7: **return** $node$ **in** $Data$
8: **end Function**
9: $S \leftarrow ad - MHT[Root]$
10: $A_i \leftarrow \emptyset$
11: **while** $S \neq \emptyset$ **do**
12: $t \leftarrow S[TOP]$
13: **if** IsLeafNode(t) **then**
14: **if** CheckInData($Data$, t) **then**
15: $Data.remove(t)$
16: $S[-1].remove(t)$
17: **else**
18: $A_i \leftarrow t$
19: **end if**
20: **else**
21: $S.append(t.right - child)$
22: $S.append(t.middle - child(if available))$
23: $S.append(t.left - child)$
24: **end if**
25: **end while**
26: **end procedure**

Step 1. Initializing Parameters: AVs mandate utilizing two distinct random functions to thwart the "replay and forge attacks" described in Section II-D: one for creating the security code g and another for generating the random seed rs. The parameters rs and sample scale ss jointly decide the sample data replicas. These functions create random sample data replicas that prevent replay attacks. Throughout each inquiry, the AVs and all ESs share the parameters g and rs. However, upon initiating a new investigation round, it is imperative to regenerate the parameters g and rs, ensuring that the newly generated random parameters differ from their predecessors.

Step 2. Creating Data Hash and Constructing ad-MHT: The AV is involved in computing data hashes for each data replica using a hash function, such as SHA-256, similar to the function used in [11], [13], [14], [15], [16], [21], [22]. While there are many alternatives to SHA-256, such as MD5 [23], SHA-3, and Whirlpool, these hash algorithms are unsuitable for use in diverse MEC scenarios for calculating hash values due to vulnerabilities to collision attacks, relatively small hash sizes, and the lack of a distributed verification mechanism [23]. Notice that in the existing EDI verification literature, SHA-256 is predominantly used solely for hash calculation, not for the overall

Algorithm 3: MEDI-V Meta Algorithm.

Require: Original data D, random functions, parameters.
Ensure: Verification status for each AV.

procedure *MEDI-VD*
  Root, $R_i \leftarrow Construct\_ad - MHT(D)$
  for each App Vendor AV_i **do**
    $MVI_i \leftarrow Create\_MVI(D, AV_i)$
    Send(MVI_i, ES_k)
    ▷ Send MVI to Edge Server ES_k
  end for
  for each Edge Server ES_k **do**
    Root, $R_j \leftarrow Reconstruct\_ad - MHT(D, MVI_i)$
    Send(R_j, AV_i)
    ▷ Send Reconstructed Root to AV, AV_i
  end for
  for each App Vendor AV_i **do**
    if $R_i = R_j$ **then**
      VerificationStatus$[AV_i] \leftarrow$ ″Success″
    else
      VerificationStatus$[AV_i] \leftarrow$ ″Corrupted″
    end if
  end for
  Return VerificationStatus
end procedure

EDIV process. To construct ad-MHT, the AV first determines the sample data replicas based on the parameters g, rs, and ss. These data hashes are then used to create ad-MHT using Algorithm 1, with the root of ad-MHT serving as the key reference for proof verification.

Phase 2: Request

In this phase, each AV creates an MVI based on the cache data list and ad-MHT of the AV using Algorithm 2, for every ES that caches data from this AV. At this stage, each ES may receive several requests from multiple AVs. The AVs construct the request information Req_i for each ES_i together with randomly generated parameters and the created MVI. The tuple Req_i is a combination of five parameters: $Req_i =< Aid_i, rs, g, ss, MVI_i >$, where Aid_i is the identity of each AV that is used to establish communication between the AV and ES. Finally, the AV initiates an integrity request for the ES_i that caches data from that AV_i.

Phase 3: Verification

Upon receiving a verification request (Req_i) from AV_i, the ES_i meticulously verifies the AV's ID Aid_i to see if it matches or not. If the identity is correct, the ES_i proceeds with the verification process. In contrast, if any inconsistencies or signs of contradiction are detected, the ES_i instantaneously rejects the integrity proof, triggering the commencement of a new integrity investigation specifically tailored for AV_i. Initially, the ES_i is tasked with creating a data hash for each of its cached data replicas, a procedure akin to *Step 2* in the initialization phase. Subsequently, ES_i generates the data hash for each replica by employing the security code g, random seed rs, and sample scale ss, compiling the cache data list. Following this, ES_i proceeds to construct the ad-MHT using MVI_i and the sampled data list, as illustrated in Algorithm A.S1 and Fig. S1 of the supplemental document. The root of the ad-MHT acts as the integrity proof for ES_i. Upon completion, ES_i is provided with proof $\hat{R}_i$ and returns to the corresponding AV.

When the AV receives a verification response Res_i from ES_i, it begins by verifying the integrity proof R_i included in the response. The process starts with verifying the identity of Eid_i. If the identity is legitimate, the AV continues with the verification. If not, the AV rejects the integrity proof and initiates a new integrity investigation for ES_i. The AV then determines the correct integrity proof using the root of the ad-MHT generated in Phase 1, which contains the accurate values of the integrity proofs. Finally, the AV compares the received $\hat{R}_i$ with the expected R_i. If they match, it indicates that the data stored in ES_i is intact. If they do not match, it suggests that the data in ES_i is corrupted, and further investigation is needed to identify which data replicas are affected.

VI. Theoretical Result Analysis

This section provides the theoretical analysis that includes the correctness, effectiveness, and security of the proposed MEDI-V approach, along with further examination of the attributes of ad-MHT.

A. Correctness Analysis

To validate the correctness of our MEDI-V approach, we first prove its ability to accurately verify the integrity of sampled data replicas. Subsequently, we demonstrate its capability to identify data corruption with a guaranteed level of probability.

Lemma 1: If a cached data replica of an ES includes a corrupted block among the sampled blocks, the integrity verification process during the corrupted block localization phase will detect the corrupted block, provided that the *Hash()* function demonstrates collision resistance.

The proof of Lemma 1 is provided in Appendix B.1 of the supplemental document.

Theorem 1: Given a dataset D that consists of k data replicas, $D = \{d_1, d_2, \ldots, d_k\}$, let $cr = \frac{kc}{k}$ represent the corruption rate in data D, where kc is the number of corrupted data in D. Let ns represent the sample data replicas within D. The proposed MEDI-V approach can successfully detect corruption with a probability of P that satisfies $P > 1 - (1 - cr)^{ns}$.

The detailed proof for Theorem 1 is provided in Appendix B.1 of the supplemental document. It illustrates that MEDI-V accomplishes the initial goal outlined in Section II-D, namely, the objective of correctness.

B. Efficiency Analysis

The results of MEDI-V theoretical time complexity and communication cost are presented in Table S1 of Appendix B.2 of the supplementary document, where m represents the number of AVs, n represents the number of ESs, and k represents the

number of data replicas. A comprehensive examination is provided, which theoretically validates that MEDI-V achieves the second objective outlined in Section II-D, namely, the efficiency goal.

C. Security Analysis

Now, we demonstrate MEDI-V's effectiveness in addressing the replay attack and the forge attack attacks outlined in Section II-D.

Theorem 2: It can be concluded that the MEDI-V scheme is secure against replay, replace, and forge attacks by an ES, as long as the $Hash()$ function is collision-resistant and the random functions used are secure.

The proof for Theorem 1 is provided in Appendix B.3 of the supplemental document. It illustrates that MEDI-V accomplishes the third goal outlined in Section II-D, namely, the security objective.

VII. Experimental Result Analysis

A. Experimental Settings

The section aims to assess the effectiveness of our proposed MEDI-V approach in comparison to existing methods, by considering computation time, corruption detection rate, and communication cost as key metrics. The experiments are iterated 50 times, and the mean value is calculated for each iteration. All experiments are implemented on a computing system equipped with an Intel Core i5 CPU operating at 1.3 GHz and 16 GB of RAM, utilizing Python 3.11.

1) Baseline Approaches: The following potential EDI verification methods are considered as baselines to validate the efficiency of our proposed MEDI-V approach.

PDP-B [24]: The PDP-B scheme originates from a prominent cloud data integrity verification model that involves AVs initiating an inquiry into the integrity of all ESs. Within this framework, ESs sample data blocks and produce a collection of integrity proofs using randomly provided parameters. The subsequent step involves AVs scrutinizing these proofs to assess the integrity of the data stored by the ESs.

EDI-V [13]: EDI-V employs a centralized probabilistic method utilizing a modified MHT. The fundamental procedure involves AVs choosing essential parameters and transmitting them to individual ESs. Subsequently, the ESs calculate and return integrity proofs to the AVs, which then scrutinize each proof to identify any instances of corruption.

ICL-EDI [22]: ICL-EDI employs a centralized approach to incorporate Third-Party Auditors (TPAs) for verifying EDI while ensuring privacy preservation. Additionally, it proposes caching verification tags on ESs using homomorphic verifiable tags to reduce communication costs.

2) Performance Measure Metrics. Computation Time (CT): The efficiency of the data integrity investigation process is computed as CT by combining the computation time and communication time essential for both the AVs and all ESs to complete the verification task. A reduced CT value signifies enhanced performance.

TABLE III
Parameters Settings

Parameters	Value Varied	Value Fixed
AV Scale (m)	2, 4, 8, 16, 32	16
Edge Scale (n)	10, 20, 30, 40, 50, 60	50
Data Scale (k)	16, 32, 64, 128, 256, 512	256
Data Size (ds)	16, 32, 64, 128, 256, 512, 1024	256
Sample Scale (ss)	32, 64, 128, 256, 512	128
Corrupted data Ratio (Cr)	0.015, 0.02, 0.025, 0.03	0.025

Corruption Detection Rate (CDR): The assessment of a scheme's CDR relies on the ratio of identified corrupted data replicas to the overall count of corrupted data replicas. Enhanced performance is indicated by a higher CDR ratio.

Communication Cost (CC): The quantification involves assessing the overall CC incurred during the completion of the data integrity investigation process between the AVs and all ESs.

3) Parameter Settings: To comprehensively assess the efficacy of MEDI-V, we define the following parameters, as outlined in Table III, aligned with the baselines works [12], [13], [22].

AV Scale (m): It denotes the number of AVs within the MVMS-MD problem scenario, ranging from 2 to 32. *ES Scale (n):* This parameter signifies the overall count of ESs to undergo auditing within the MVMS-MD problem scenario, ranging from 5 to 100. *Data Scale (k):* The overall volume of data in the AV is distributed among multiple ES instances. To enhance the representation of real-world scenarios, we assign to each ES the responsibility of caching any number of data replicas from multiple AVs. *Data Size (ds):* It represents the size of each data replica (KB as a unit). *Sample Scale (ss):* measured by the number of sampled data blocks in each data replica. *Corrupted Block Ratio (Cr):* The proportion of impaired blocks within corrupted data replicas, is expressed as the ratio of the corrupted block count to the overall number of blocks. For each evaluation, a single parameter is adjusted while the remaining parameters are maintained at their default values. The experiments are systematically repeated/iterated, and the results are recorded to derive the mean values, ensuring a robust and thorough analysis [12], [13].

B. Experimental Results Related to Existing EDIV Methods

1) Computation Time: We conduct a comprehensive analysis to evaluate the time consumption across different phases of the MEDI-V process, examining factors influencing each phase's duration.

1) *Time Consumption for Phase 1 (Initialization):* Fig. 11 presents a visual representation of the time required for the initialization phase across four distinct EDI schemes, varying with m, ds, k, and ss while the detailed experimental results are provided in Table S2 of the supplemented document. In Fig. 11(a), (b), and (c), our proposed MEDI-V approach demonstrates optimality in this phase. Additionally, Fig. 11(d) indicates that MEDI-V tends to showcase a minimal time consumption, but EDI-V is constant. To clarify this observation, we conducted the following thorough analysis of the time needed for hash operations carried out by the AVs. In the PDP-B approach,

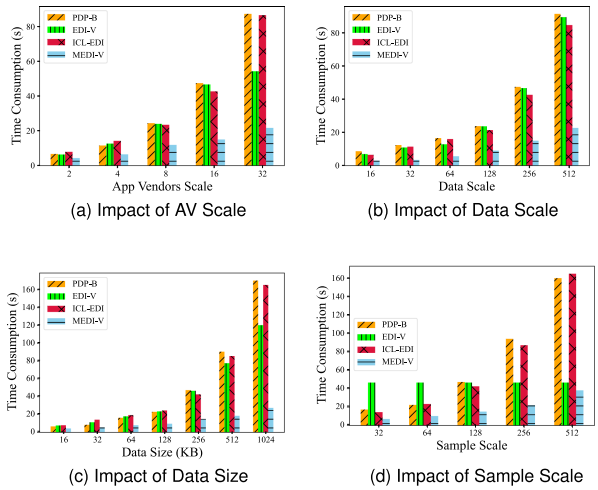


Fig. 11. Phase 1 Time Consumption.

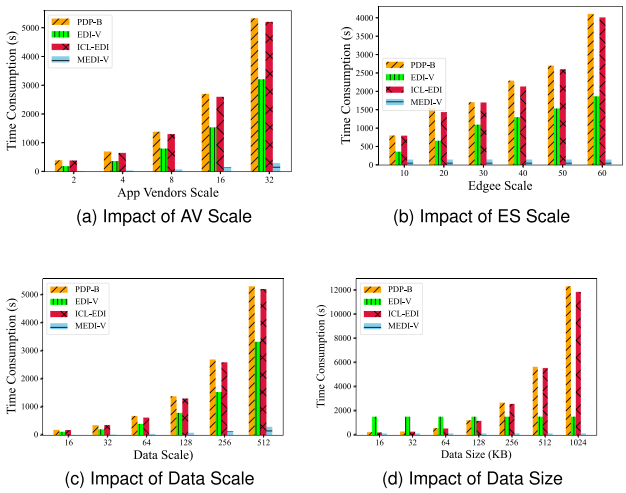


Fig. 12. Phase 2 time consumption.

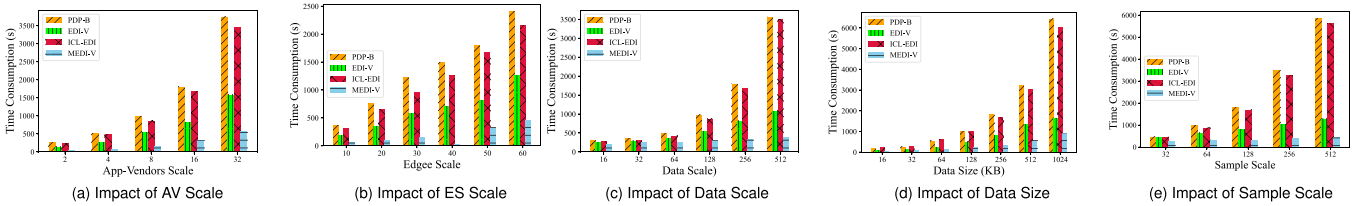
the calculation of hashes for individual sampling data entails conducting $k \times ss$ hash operations. Within the EDI-V framework, the creation of the variable MHT necessitates $k \times (2ds - 1)$ hash operations. In contrast, the ICL-EDI technique involves $2k \times ss$ hash operations for sampling data. Conversely, our MEDI-V scheme constructs an ad-MHT utilizing all data hashes, demanding $(2ss \times k - 1)$ hash operations. Fig. 11(a) indicates that time consumption is directly proportional to m. Furthermore, Fig. 11(b) illustrates that an increase in ds results in heightened computation cost due to prolonged computation times for the $Hash()$ function. Fig. 11(c) reveals a direct correlation between time consumption and k for the PDP-B, EDI-V, ICL-EDI, and MEDI-V schemes. These schemes involve hash computations for data or ad-MHT construction based on all data hashes. Lastly, Fig. 11(d) demonstrates that PDP-B, ICL-EDI, and MEDI-V exhibit escalated time consumption with larger ss values, whereas EDI-V remains unaffected. This discrepancy arises because PDP-B, ICL-EDI, and MEDI-V exclusively compute hashes for ss sampled data, whereas EDI-V hashes all data.

2) *Time Consumption for Phase 2 (Request):* Fig. 12 illustrates the time consumption during the request phase by the four competing schemes across varying configurations of m, n, k, and ds while the detailed experimental data are provided in Table S3 of the supplementary document. Based on these results, it can be seen that MEDI-V consistently surpasses PDP-B, EDI-V, and ICL-EDI in terms of temporal efficiency across most of the parameter values. In particular, Fig. 12(a) illustrates the impact of varying m on computation efficiency across four approaches, revealing a consistent increase in processing time as m increases. This escalation stems from the heightened workload due to the high number of AVs, which requires additional rounds of integrity proofs and requests to ESs for data replicas. The direct correlation between m and computation time is evident, with MEDI-V exhibiting lower time consumption due to its single-round per-edge approach, despite increasing m. Fig. 12(b) depicts the effects of changing n on efficiency, with all methods except MEDI-V experiencing increased time as n increase. MEDI-V maintains almost constant times by constructing the MVI data structure, which is attributed to prolonged request generation for corruption detection. This approach entails constructing MVI in a per-edge, single-round fashion to authenticate data integrity, thus minimizing computation time. A notable feature of the MEDI-V approach is its minimal time requirement, which is achieved by constructing MVI exclusively for generating requests to each ES during corruption detection.

In Fig. 12(c), MEDI-V demonstrates superior time efficiency compared to PDP-B, EDI-V, and ICL-EDI as the data scale k increases from 16 to 512. Despite all four schemes experiencing elevated time requirements with higher k values, MEDI-V exhibits a less steep growth rate in time consumption. For instance, at k=16, the average time taken by PDP-B, EDI-V, ICL-EDI, and MEDI-V is denoted by 191.22, 124.1, 183.17, and 11.36, respectively. With an increase in k to 512, the corresponding time consumption rises to 5307.92 (2775.82% increase) for PDP-B, 3328.27 (2681.93% increase) for EDI-V, 5208.3 (2843.42% increase) for ICL-EDI, and 298.66(2629.05% increase) for MEDI-V. The per-edge single-round methodology of MEDI-V incurs a lower computation cost compared to PDP-B, EDI-V, and ICL-EDI.

Finally, in Fig. 12(d), MEDI-V demonstrates effective time efficiency compared to PDP-B, EDI-V, and ICL-EDI as the data size ds increases from 16KB to 1024KB. Here, the time requirements for PDP-B, EDI-V, and ICL-EDI approaches increase with higher ds, but MEDI-V remains the same for increased ds. The reason is that, in this phase, MEDI-V constructs an MVI data structure that is unaffected by varying data sizes.

3) *Time Consumption for Phase 3 (Verification):* Fig 13 depicts the time allocated during the verification phase by the four competing schemes across different configurations of m, n, k, ss, and ds. In the scenarios presented (Fig. 13(a)–(e)), MEDI-V consistently



(a) Impact of AV Scale (b) Impact of ES Scale (c) Impact of Data Scale (d) Impact of Data Size (e) Impact of Sample Scale

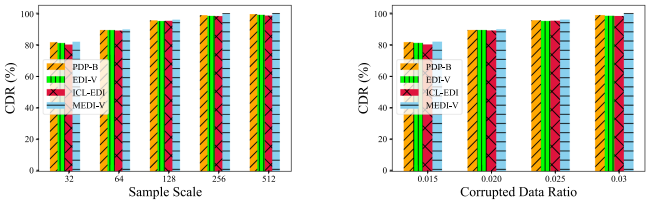
Fig. 13. Phase 3 time consumption.

outperforms PDP-B, EDI-V, and ICL-EDI in terms of temporal efficiency across most parameter values.

Specifically, the graphical representations in Fig. 13(a) and (b) illustrate the impact of varying m and n on computation efficiency across four distinct approaches, respectively. Increasing m in Fig. 13(a) results in consistent growth in time expended by all methods, which is attributed to the heightened workload on ESs for integrity-proof computation, hence affirming a direct positive correlation between m and time consumption. Conversely, Fig. 13(b) shows that escalating n leads to proportional time increments across approaches, as more ESs prolong proof generation and corruption detection, hence affirming a direct relationship between n and time consumption. Notably, MEDI-V exhibits lower time consumption when compared with alternative approaches in both cases. MEDI-V stands out for its lower time consumption due to its streamlined verification process in a single round by reconstructing the root of ad-MHT, emphasizing its comparative advantage.

In Fig. 13(c), MEDI-V exhibits superior performance in time consumption compared to PDP-B, EDI-V, and ICL-EDI as k increases from 16 to 512. While all four schemes experience increased time requirements with higher k values, MEDI-V demonstrates a slower rate of time consumption growth. For instance, at k =16, the average time taken by PDP-B, EDI-V, ICL-EDI, and MEDI-V is 298.25s, 260.89s, 275.59s, and 204.65, respectively. Upon increasing k to 512, the corresponding time consumption rises to 3564.31s (1195.08% increase) for PDP-B, 1065.36s (408.36% increase) for EDI-V, 3497.25s (1269.01% increase) for ICL-EDI, and 380.91s (186.13% increase) for MEDI-V. MEDI-V's per-edge single-round approach results in a smaller computation cost compared to PDP-B, EDI-V, and ICL-EDI. Specifically, at k values of 128, 256, and 512, MEDI-V's time consumption is, on average, 329.82%, 561.10%, and 935.74% lower than PDP-B. Additionally, it is 181.88%, 255.69%, and 279.69% lower than EDI-V, and 292.86%, 526.47%, and 918.13% lower than ICL-EDI, respectively.

In Fig. 13(d) and (e), a consistent pattern emerges concerning the time consumption of PDP-B, EDI-V, and ICL-EDI, whereas we provide the detailed experimental results in Table S4 of the supplemented document. Notably, with an increase in both the ss and ds, all three schemes exhibit noteworthy escalations in time consumption. The augmentation in ss demands a greater number of hash operations, while the increase in ds results in prolonged hash computation times. In particular, within PDP-B and ICL-EDI schemes, additional time is required for ESs to generate integrity proofs for each data replica. Similarly, in the case of EDI-V, ESs are mandated to generate a larger MHT for each data replica. In contrast, MEDI-V demonstrates a reduced time requirement when compared with EDI-V for the reconstruction of the ad-MHT. Nevertheless, with an increase in ds, corresponding to an increase in data size, the time necessary for hash calculation experiences a proportional rise. On the other hand, the reconstruction of ad-MHT is less affected as ss increases from 32 to 512.



(a) Impact of Sample Scale (b) Impact of Corruption Ratio

Fig. 14. Corruption detection rate.

2) Corruption Detection Rate: In the methodology of PDP-B, EDI-V, ICL-EDI, and MEDI-V schemes, a constant quantity of data replicas is sampled each time from available data replicas for verification. Consequently, the effectiveness of their CDRs is influenced by both ss and cr. The correlation between CDR and various ss and corrupted data ratios cr is depicted in Fig. 14. In specific, Fig. 14(a) illustrates the comparative analysis of CDR among the evaluated methodologies including PDP-B, EDI-V, ICL-EDI, and the proposed MEDI-V method across varying ss from 32 to 256. Several key observations emerge from the presented results. Initially, all methodologies achieve nearly perfect CDRs approaching 100% as the ss increases significantly. Conversely, when the ss remains relatively small, the CDRs do not surpass 93%. This limitation stems from the inherent probabilistic nature of sampling subsets from the complete dataset, which may occasionally result in missing corrupted data. Ultimately, the analysis underscores a positive correlation between ss and CDRs. As ss increased from 32 to 256, the average CDR for PDP-B, EDI-V, ICL-EDI, and the proposed MEDI-V also exhibited growth from 81.56% to 98.65%, 81.07% to 98.22%, 80.08% to 98.10%, and 81.88% to 99.87%, respectively. However, it is crucial to highlight that this enhancement was accompanied by notable computation costs for PDP-B, EDI-V, and ICL-EDI, as illustrated in Fig. 13(e). In contrast, MEDI-V demonstrated a relatively lower overhead.

In Fig. 14(b), the impact of augmenting cr on the mean CDR is explored while maintaining a fixed ss of 128. The outcomes revealed a uniform trend across the four methodologies, where

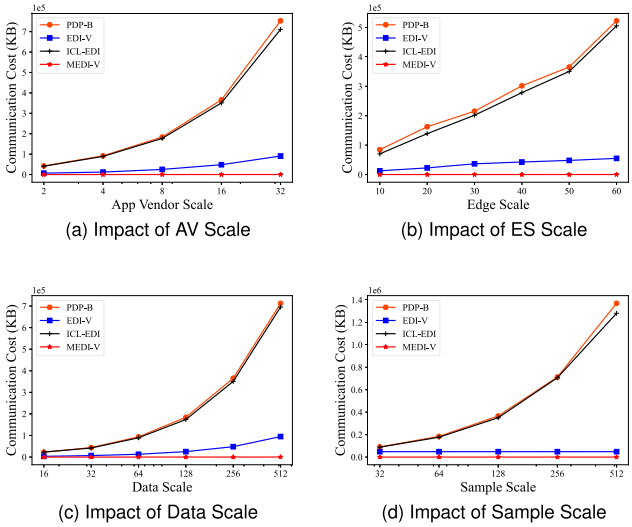


Fig. 15. Communication cost in KB.

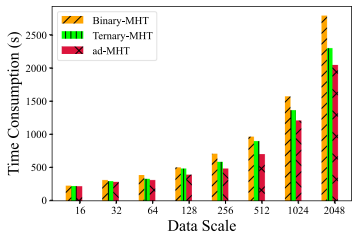


Fig. 16. Overall time consumption for constructing the MHTs.

PDP-B, EDI-V, ICL-EDI, and MEDI-V register an incremental rise in average detection precision from 86.75%, 87.21%, 86.38%, and 87.09% to 98.83%, 98.71%, 98.07%, and 98.87%, respectively, as cr progresses from 0.015 to 0.03.

3) Communication Cost: CC becomes apparent in Phase 2 and Phase 3, such as during the request and verification phase. In Fig. 15, the comparative analysis illustrates the CCs associated with integrity verification across various schemes, considering the impact of m, n, k, and ss. Notice that the detailed experimental findings are presented in Table S5 of the supplementary document. In Fig. 15(a), the comparison of communication costs is presented as m increases from 2 to 32. The impact of m on the communication efficiency of all approaches is notable. The reason lies in the increased m, leading to a substantial transmission (*challenge-response*) of integrity proofs between ESs and AVs for PDP-B, EDI-V, and ICL-EDI. In contrast, MVMS-MD introduces a more complicated scenario between the AV and ESs, which results in higher network traffic. Nevertheless, it is important to note that MEDI-V, being a single-round verification approach, contributes significantly less to the overall network traffic. Across all cases, MEDI-V consistently yields 5072.33 times, 634.82 times, and 4975.9 times less network traffic when compared with PDP-B, EDI-V, and ICL-EDI, respectively. Fig. 15(b) illustrates the impact of n on CC. It is evident that as n increases, there is a corresponding escalation in the communication cost across all methodologies, including PDF-B, EDI-V, and ICL-EDI. This phenomenon stems from the necessity for additional time to inspect more ESs, thereby requiring increased network traffic for the transmission of integrity proofs. Notably, MEDI-V consistently outperforms PDP-B, EDI-V, and ICL-EDI across all scenarios, albeit with varying margins. Furthermore, the majority of MEDI-V's communication expenses occur at the network edge, in contrast to backhaul networks such as PDP-B, EDI-V, and ICL-EDI.

In Fig. 15(c), it is observed that the CC in MEDI-V remains unaffected despite an increase in data scale k. Conversely, for PDP-B, EDI-V, and ICL-EDI, the rise in k significantly influences the communication patterns. This discrepancy arises from the fact that larger k values result in heightened communication occurrences between ESs and AVs in PDP-B, EDI-V, and ICL-EDI, as well as increased interactions among individual ESs. Notably, MEDI-V exhibits no discernible impact from changes in ds. Finally, Fig. 15(d) illustrates that the CC of EDI-V and MEDI-V remain consistent across various ss. This consistency arises from EDI-V providing proof that it is the root of an MHT, and MEDI-V providing an ad-MHT, irrespective of the ss. Conversely, only PDP-B and ICL-EDI notably impact CC concerning ss.

C. Adaptability Analysis of ad-MHT

We now analyze to evaluate the adaptability of the proposed ad-MHT to other MHTs in terms of time consumption. When examining the adaptability of ad-MHT, it is worth noting that it is affected by the data scale k. Fig. 16 provides a visual representation of the time required across three distinct MHTs which varies with k. Although all three variants experience increased time requirements with higher k values, ad-MHT exhibits a less steep growth rate in time consumption. For instance, for $k = 16$, the average times taken by binary MHT, ternary MHT, and ad-MHT are 218.98s, 221.4s, and 227.3s, respectively. However, with an increase in k to 2048, the corresponding time consumption rises to 2787.4 (1126.3% increase) for binary MHT, 2298.4 (938.12% increase) for ternary MHT, and 2044.39s (833.69% increase) for ad-MHT, in the case of the proposed MEDI-V. This illustrates that ad-MHT reduces time consumption as the data scale increases compared to binary and ternary MHTs.

D. Evaluation of MEDI-V Security

We have now experimentally verified the resilience of our proposed MEDI-V framework against the specified attack model. In particular, we have conducted an evaluation that simulated the potential threats outlined in our attack model: replay attack, replace attack and forgery attack. Each attack type was individually applied to assess the MEDI-V model's resistance in the MVMS-MD MEC scenario in this experimental setup. For replay attacks, previously generated integrity proofs were stored and later resent by an ES in place of newly generated proofs, simulating unauthorized reuse. For model replace attacks, we have altered specific data within the ES's cache and then generated and transmitted a modified integrity proof to the AV. Lastly, forgery attacks were tested by attempting to generate integrity proofs without access to the original cached data, challenging the system's ability to prevent unauthorized proof creation. Each attack was conducted across multiple trials,

TABLE IV
EVALUATION OF MEDI-V SECURITY AGAINST THE THREAT MODEL

Attack Type	No. of Trials	Detection Success Rate	False Positive Rate	Hash Collisions Observed
Replay	50	100%	0%	0
Replace	50	100%	0%	0
Forgery	50	100%	0%	0
Overall	150	100%	0%	0

with outcomes recorded in terms of detection success rate, false positive rate, and hash collisions. The experimental results are included in Table IV. These results manifest that our MEDI-V approach has defended each attack completely by achieving a 100% detection success rate, 0% false positive rate, and no hash collisions.

VIII. RELATED WORK

Before the emergence of EDI, Cloud Data Integrity (CDI) underwent extensive exploration [25], [26]. Prominent studies, such as those on Proof of Data Recoverability (PoR) [27], [28] and Provable Data Possession (PDP) [29], contributed significantly to the understanding of CDI [30]. PDP, designed to verify the integrity of data stored on remote servers through probabilistic guarantees, has been subject to various adaptations [29]. However, many PDP variations rely on a trusted TPA [31] and demand more computation resources, making them less suitable for the specific requirements of EDI [16]. Likewise, PoR entails additional storage and computation demands, rendering it impractical for addressing the unique challenges posed by EDI.

Recognizing the limitations of traditional CDI approaches, recent studies have delved into specialized techniques to address EDI issues. Tong et al. [11] utilized the traditional PDP scheme to verify the integrity of users' data on ES. While their primary focus was on safeguarding AV privacy, they did not address the challenge of efficiently verifying the integrity of large-scale data replicas from the perspective of AVs. Li et al. [13] proposed EDI-V, utilizing a variable MHT and shuffled data block samples for integrity verification. However, EDI-V's probabilistic nature and suboptimal efficiency in high-integrity scenarios led to the development of MEDI-V, a probabilistic guarantee scheme offering improved accuracy and efficiency. In response to the limitations of EDI-V, Li et al. proposed EDI-S, a full integrity guarantee-based scheme utilizing elliptic curve cryptography [14], CooperEDI, a distributed consensus mechanism-based scheme for efficiency improvement [21], and EdgeWatch, a blockchain-based distributed verification scheme resilient to Byzantine ES and targeted attacks [32]. Additionally, Cui et al. [22] introduced ICL-EDI, leveraging homomorphic signature technology for computation simplicity, and Ding et al. [33] proposed EDI-DA, a scheme supporting dynamic updates of edge data replicas using Index-Single Linked Table (I-SLT) and dynamic arrays. Batch verification techniques, as investigated by Tong et al. [34], employ a TPA to aid in the verification procedure. Nevertheless, the adoption of these methods escalates the intricacy associated with data integrity verification and introduces potential privacy vulnerabilities.

TABLE V
RELATED WORK COMPARISON

Representative schemes	Auditing Scenario [1]	Security Assumption (AV-TPA-ES) [2]	Challenge Response Round [3]	Architectural Scenario [4]
PDP-B [11]	A-T-E	T-T-S	M	SV-SS-SD
EDI-V [13]	A-E	T-U	M	SV-MS-SD
EDI-S [14]	A-E	T-U	M	SV-MS-SD
CooperEDI [21]	E-E	T-T	M	SV-MS-SD
ICL-EDI [22]	A-E	T-U	M	SV-MS-SD
EdgeWatch [32]	E-E	U-U	M	SV-MS-SD
EDI-DA [33]	A-E	T-U	M	SV-MS-SD
MVMS-SA [15]	A-E	T-U	M	MV-MS-SD
DVA-P [35]	A-E	T-U	M	SV-MS-SD
MEDI-V (Proposed)	A-E	T-U	S	MV-MS-MD

[1] A-E (AV-ES); A-T-E (AV-TPA-ES); E-E (ES-ES)
[2] T-T-S (Trusted AV, Trusted TPA, semi-trusted ES); T-U (Trusted AV-Untrusted ES); T-T (Trusted ES-trusted ES); U-U (Untrusted ES-untrusted ES)
[3] M-Multi-round; S-Single round
[4] SV-single AV; MD-multi AV; SS-single ES; MS-multiple ES; SD-single data; MD-multi-data

Typically, most of the methods discussed above use a TPA. In our proposed method, TPA is not employed as a verifier. The AV itself verifies the data integrity, ensuring that the complexities and concerns associated with TPA are not present in the proposed methodology. Consequently, other types of attacks like spoofing attacks [18] and data leakage attacks [19] are not considered in the proposed model. Existing EDI solutions generally focus on scenarios with a single AV and multiple ESs. To address this limitation, Zhao et al. [15] developed a Smart Inspection Algorithm (SIA) that pre-selects unreliable data replicas for different AVs in each verification round by considering both the Quality-of-Service (QoS) of cache services and the unverified time of data replicas. Additionally, the work in [35] explored the dynamic verification assignment problem in EDI to optimize the distribution of computation and communication resources on edge servers, aiming to maximize the long-term verification of data replicas within resource constraints.

On the other hand, these discussed methods commonly employ a per-edge multi-round approach. Specifically, they verify data independently, which requires multiple rounds of communication between the AV and each ES. This results in elevated communication costs and verification time. Notably, none of these methodologies aligns with the specific use-case scenario we have studied for EDI. To address this challenge, this paper proposes a per-edge single-round scheme. In this scheme, the entire verification process entails only one round of communication between the AV and each ES. This streamlined verification process verifies all data cached by each ES, thereby reducing both communication costs and verification time in the integrity verification process. As demonstrated in Section VII, MEDI-V consistently outperforms representative EDI schemes in terms of performance. Table V provides a comprehensive comparison of key elements between our work and related studies. Our work specifically focuses on the following aspects: the auditing environment, security model assumptions, *challenge-response* mechanisms employed, and architectural design choices.

IX. CONCLUSIONS AND FUTURE WORK

In this paper, we present a practical MEC scenario named MVMS-MD, which poses significant challenges for existing methods to verify the integrity of multiple-edge data. To tackle these challenges, we propose a solution called MEDI-V for per-edge, single-round MEDI verification. The objective of MEDI-V

is to verify all data replicas cached on the ESs while minimizing computation and communication costs. Our proposed approach introduces the concept of ad-MHT, which enables the construction of an adaptive tree and auxiliary information known as MVI essential for reconstructing the ad-MHT. This simplifies the integrity verification process for both application AVs and ESs with negligible additional load. We thoroughly evaluate the effectiveness of MEDI-V through theoretical analysis and experimental validation and compare it against three potential existing schemes. Our results convincingly demonstrate that MEDI-V provides an efficient and effective solution to MEDI challenges and demonstrates a clear advantage in terms of both efficiency and correctness.

In future work, we aim to enhance the MEDI-V framework by introducing a method for efficiently detecting corrupted ESs. We could swiftly identify potentially corrupted servers and then prioritize the inspection of these identified servers by leveraging user activity, thereby significantly reducing inspection time and ensuring more robust data integrity in dynamic network conditions.

REFERENCES

[1] Z. Chang, S. Liu, X. Xiong, Z. Cai, and G. Tu, "A survey of recent advances in edge-computing-powered Artificial Intelligence of Things," *IEEE Internet Things J.*, vol. 8, no. 18, pp. 13849–13875, Sep. 2021.

[2] A. N. (if available), "Top edge computing trends to watch in 2020," 2020. [Online]. Available: https://www.techtarget.com/searchcio/tip/Top-edge-computing-trends-to-watch-in-2020

[3] Q. Zhang, L. Cheng, and R. Boutaba, "Cloud computing: State-of-the-art and research challenges," *J. Internet Serv. Appl.*, vol. 1, pp. 7–18, 2010.

[4] J. Zhang, Z. Zhang, and H. Guo, "Towards secure data distribution systems in mobile cloud computing," *IEEE Trans. Mobile Comput.*, vol. 16, no. 11, pp. 3222–3235, Nov. 2017.

[5] B. Varghese and R. Buyya, "Next generation cloud computing: New trends and research directions," *Future Gener. Comput. Syst.*, vol. 79, pp. 849–861, 2018.

[6] N. Abbas, Y. Zhang, A. Taherkordi, and T. Skeie, "Mobile edge computing: A survey," *IEEE Internet Things J.*, vol. 5, no. 1, pp. 450–465, Feb. 2018.

[7] J. Ren, D. Zhang, S. He, Y. Zhang, and T. Li, "A survey on end-edge-cloud orchestrated network computing paradigms: Transparent computing, mobile edge computing, fog computing, and cloudlet," *ACM Comput. Surv.*, vol. 52, no. 6, pp. 1–36, 2019.

[8] D. Liu, H. Liang, X. Zeng, Q. Zhang, Z. Zhang, and M. Li, "Edge computing application, architecture, and challenges in ubiquitous power Internet of Things," *Front. Energy Res.*, vol. 10, 2022, Art. no. 850252.

[9] R. Singh and S. S. Gill, "Edge AI: A survey," *Internet Things Cyber- Phys. Syst.*, vol. 3, pp. 71–92, 2023.

[10] W. Yu et al., "A survey on the edge computing for the Internet of Things," *IEEE Access*, vol. 6, pp. 6900–6919, 2017.

[11] W. Tong, B. Jiang, F. Xu, Q. Li, and S. Zhong, "Privacy-preserving data integrity verification in mobile edge computing," in *Proc. IEEE 39th Int. Conf. Distrib. Comput. Syst.*, 2019, pp. 1007–1018.

[12] F. Yang, Y. Sun, Q. Gao, and X. Chen, "EDI-C: Reputation-model-based collaborative audit scheme for edge data integrity," *Electronics*, vol. 13, no. 1, 2023, Art. no. 75.

[13] B. Li, Q. He, F. Chen, H. Jin, Y. Xiang, and Y. Yang, "Auditing cache data integrity in the edge computing environment," *IEEE Trans. Parallel Distrib. Syst.*, vol. 32, no. 5, pp. 1210–1223, May 2021.

[14] B. Li, Q. He, F. Chen, H. Jin, Y. Xiang, and Y. Yang, "Inspecting edge data integrity with aggregate signature in distributed edge computing environment," *IEEE Trans. Cloud Comput.*, vol. 10, no. 4, pp. 2691–2703, Oct.-Dec. 2022.

[15] Y. Zhao, Y. Qu, F. Chen, Y. Xiang, and L. Gao, "Data integrity verification in mobile edge computing with multi-vendor and multi-server," *IEEE Trans. Mobile Comput.*, vol. 23, no. 5, pp. 5418–5432, May 2024.

[16] J. Li, H. Yan, and Y. Zhang, "Efficient identity-based provable multi-copy data possession in multi-cloud storage," *IEEE Trans. Cloud Comput.*, vol. 10, no. 1, pp. 356–365, Jan.-Mar. 2022.

[17] L. Qiao, Y. Li, F. Wang, and B. Yang, "Lightweight integrity auditing of edge data for distributed edge computing scenarios," *Ad Hoc Netw.*, vol. 133, 2022, Art. no. 102906.

[18] G. Xu, Y. Yang, C. Yan, and Y. Gan, "A probabilistic verification algorithm against spoofing attacks on remote data storage," *Int. J. High Perform. Comput. Netw.*, vol. 9, no. 3, pp. 218–229, 2016.

[19] Y. Ren, J. Qi, Y. Liu, J. Wang, and G.-J. Kim, "Integrity verification mechanism of sensor data based on bilinear map accumulator," *ACM Trans. Internet Technol.*, vol. 21, no. 1, pp. 1–19, 2021.

[20] S. Meena, E. Daniel, and N. Vasanthi, "Survey on various data integrity attacks in cloud environment and the solutions," in *Proc. 2013 Int. Conf. Circuits, Power Comput. Technol.*, 2013, pp. 1076–1081.

[21] B. Li et al., "Cooperative assurance of cache data integrity for mobile edge computing," *IEEE Trans. Inf. Forensics Secur.*, vol. 16, pp. 4648–4662, 2021.

[22] G. Cui et al., "Efficient verification of edge data integrity in edge computing environment," *IEEE Trans. Serv. Comput.*, vol. 15, no. 6, pp. 3233–3244, Nov./Dec. 2022.

[23] S. R. Prasanna and B. Premananda, "Performance analysis of md5 and SHA-256 algorithms to maintain data integrity," in *Proc. 2021 Int. Conf. Recent Trends Electron., Inf., Commun. Technol.*, 2021, pp. 246–250.

[24] A. F. Barsoum and M. A. Hasan, "Provable multicopy dynamic data possession in cloud computing systems," *IEEE Trans. Inf. Forensics Secur.*, vol. 10, no. 3, pp. 485–497, Mar. 2015.

[25] F. Zafar et al., "A survey of cloud computing data integrity schemes: Design challenges, taxonomy and future trends," *Comput. Secur.*, vol. 65, pp. 29–49, 2017.

[26] L. Zhou, A. Fu, S. Yu, M. Su, and B. Kuang, "Data integrity verification of the outsourced Big Data in the cloud environment: A survey," *J. Netw. Comput. Appl.*, vol. 122, pp. 1–15, 2018.

[27] A. Juels and B. S. Kaliski Jr, "PORS: Proofs of retrievability for large files," in *Proc. 14th ACM Conf. Comput. Commun. Secur.*, 2007, pp. 584–597.

[28] C. B. Tan, M. H. A. Hijazi, Y. Lim, and A. Gani, "A survey on proof of retrievability for cloud data integrity and availability: Cloud storage state-of-the-art, issues, solutions and future trends," *J. Netw. Comput. Appl.*, vol. 110, pp. 75–86, 2018.

[29] G. Ateniese et al., "Provable data possession at untrusted stores," in *Proc. 14th ACM Conf. Comput. Commun. Secur.*, 2007, pp. 598–609.

[30] K. D. Bowers, A. Juels, and A. Oprea, "Proofs of retrievability: Theory and implementation," in *Proc. ACM Workshop Cloud Comput. Secur.*, 2009, pp. 43–54.

[31] W. Guo et al., "Outsourced dynamic provable data possession with batch update for secure cloud storage," *Future Gener. Comput. Syst.*, vol. 95, pp. 309–322, 2019.

[32] B. Li, Q. He, L. Yuan, F. Chen, L. Lyu, and Y. Yang, "EdgeWatch: Collaborative investigation of data integrity at the edge based on blockchain," in *Proc. 28th ACM SIGKDD Conf. Knowl. Discov. Data Mining*, 2022, pp. 3208–3218.

[33] Y. Ding, Y. Li, W. Yang, and K. Zhang, "Edge data integrity verification scheme supporting data dynamics and batch auditing," *J. Syst. Archit.*, vol. 128, 2022, Art. no. 102560.

[34] W. Tong, W. Chen, B. Jiang, F. Xu, Q. Li, and S. Zhong, "Privacy-preserving data integrity verification for secure mobile edge storage," *IEEE Trans. Mobile Comput.*, vol. 22, no. 9, pp. 5463–5478, Sep. 2023.

[35] Y. Zhao, Y. Qu, Y. Xiang, C. Shi, F. Chen, and L. Gao, "Long-term over one-off: Heterogeneity-oriented dynamic verification assignment for edge data integrity," *IEEE Trans. Mobile Comput.*, vol. 23, no. 5, pp. 4601–4616, May 2024.

Md Rashedul Islam (Member, IEEE) received the BSc degree in computer science and engineering from Hajee Mohammad Danesh Science and Technology University (HSTU), Bangladesh and the MSc degree in computer science and engineering from the Rajshahi University of Engineering and Technology (RUET), Bangladesh, in 2019. He is currently working toward the PhD degree with the School of Information Technology, Deakin University. He is an academic faculty member at HSTU. His research direction includes edge computing, machine learning, federated learning, blockchain applications, and remote sensing image analysis.

Yong Xiang (Senior Member, IEEE) received the PhD degree in electrical and electronic engineering from the University of Melbourne, Australia. He is a professor with the School of Information Technology, Deakin University, Australia. His research interests include distributed computing, cybersecurity and privacy, machine learning and AI, and communications engineering. He has published 7 authored books, more than 270 refereed journal articles, and more than 110 conference papers in these areas. Professor Xiang is the senior area editor of *IEEE Signal Processing Letters*, the associate editor of *IEEE Communications Surveys and Tutorials*, and the associate editor of *Computer Standards and Interfaces*. He was the associate editor of *IEEE Signal Processing Letters* and *IEEE Access*, and the guest editor of *IEEE Transactions on Industrial Informatics*, *IEEE Multimedia*, etc. He has served as honorary chair, general chair, program chair, TPC chair, symposium chair, and track chair for many conferences and has been invited to give keynotes at numerous international conferences.

Md Palash Uddin (Member, IEEE) received the BSc degree in computer science and engineering (CSE) from Hajee Mohammad Danesh Science and Technology University (HSTU), Bangladesh, the MSc degree in CSE from the Rajshahi University of Engineering & Technology, Bangladesh, and the PhD degree in IT from Deakin University, Australia. Dr. Uddin is currently serving as a postdoctoral research fellow with the School of IT, Deakin University, Australia, and is also a faculty member with HSTU, Bangladesh. He has authored more than 75 journal and conference papers in leading venues, including *IEEE Transactions on Parallel and Distributed Systems*, *IEEE Transactions on Services Computing*, *IEEE Transactions on Mobile Computing*, *IEEE Signal Processing Letters*, and *ACM Computing Surveys*. Additionally, he actively reviews for top-tier journals and conferences. His primary research interests lie in federated learning and machine learning, with a focus on their applications in data privacy, security, and integrity verification.

Yao Zhao (Member, IEEE) received the MS degree in computer science and technology from the Nanjing University of Aeronautics and Astronautics, in 2019. She is currently working toward the PhD degree with the School of Information Technology, Deakin University. She worked for Suning Finance as the blockchain product Manager. Besides, she used to be a blockchain developer in Dazhong News Group, in 2020. Her research direction includes blockchain, IoT, and edge computing.

Jonathan Kua (Member, IEEE) received the BEng (First Class Hons.) degree in telecommunications and network engineering and the PhD degree in telecommunications engineering from the Swinburne University of Technology, Melbourne, Australia, in 2014 and 2019, respectively. He is currently a senior lecturer in Internet of Things with the School of Information Technology, Deakin University, Melbourne. His research interests include data transport protocols and active queue management, adaptive streaming and content delivery, Internet of Things and industrial cyber-physical systems, with an overarching focus on improving their network performance and quality of service. Dr. Kua was the recipient of the Netflix PhD scholarship award (2015–2019) and the second runner-up of the DASH-IF Best PhD Dissertation Award on "Algorithms and Protocols for Adaptive Content Delivery over the Internet" (2019). He currently serves as a guest editor for the IEEE Journal on Selected Areas in Communications (JSAC) Special Issue on "Co-Design of Communication, Computing, and Control in Industrial Cyber-Physical Systems".

Longxiang Gao (Senior Member, IEEE) received the PhD degree in computer science from Deakin University, Australia. He is currently a professor in Shandong Computer Science Center with the Qilu University of Technology (Shandong Academy of Sciences). He was a senior lecturer with the School of Information Technology, Deakin University, and a post-doctoral research fellow with IBM Research & Development, Australia. His research interests include Fog/Edge computing, Blockchain, data analysis, and privacy protection. Dr. Gao has more than 180 publications, including patents, monographs, book chapters, journal and conference papers. Some of his publications have been published in top venues, such as *IEEE Transactions on Mobile Computing*, *IEEE Transactions on Parallel and Distributed Systems*, *IEEE Transactions on Services Computing*, *IEEE Transactions on Knowledge and Data Engineering*, *IEEE Transactions on Dependable and Secure Computing*, *IEEE Transactions on Vehicular Technology*, *IEEE Transactions on Computational Social Systems*, *IEEE Transactions on Industrial Informatics*, *IEEE Transactions on Network Science and Engineering*, *IEEE Transactions on Wireless Communications* and *ACM Computing Surveys*. He has been the chief investigator (CI) for more than 30 research projects (the total awarded amount is over $6million), from pure research projects to contracted industry research. He is a ACM.