symmetry

MDPI

Article

eCLASS: Edge-Cloud-Log Assuring-Secrecy Scheme for Digital Forensics

Junyoung Park and Eui-Nam Huh *

Department of Computer Science and Engineering, Kyung Hee University, 1732, Deogyeong-daero, Giheung-gu, Yongin-si, Gyeonggi-do 17104, Korea; parkhans@khu.ac.kr

* Correspondence: johnhuh@khu.ac.kr; Tel.: +82-31-201-3778

Received: 30 August 2019; Accepted: 19 September 2019; Published: 22 September 2019

Abstract: User activity logs are important pieces of evidence in digital forensic investigations. In cloud forensics, it is difficult to collect user activity logs due to the fact of virtualization technologies and the multitenancy environment, which can infringe upon user privacy when collecting logs. Furthermore, the computing paradigm is shifting from conventional cloud computing toward edge computing, employing the advances of 5G network technology. This change in the computing paradigm has also brought about new challenges for digital forensics. Edge nodes that are close to users are exposed to security threats, and the collection of logs with limited computing resources is difficult. Therefore, this study proposes a logging scheme that considers log segmentation and distributed storage to collect logs from distributed edge nodes and to protect log confidentiality by taking into account edge-cloud characteristics. This scheme protects the integrity of log data collected by a multi-index chain network. To demonstrate the performance of the proposed scheme, edge nodes with three different capacity types were used, and the proposed log-segmentation method performed 29.4% to 64.2% faster than the Cloud-Log Assuring-Secrecy Scheme (CLASS) using 2048 bit Rivest-Shamir-Adleman (RSA) in three types of edge nodes for log-confidentiality protection. The log segmentation of edge CLASS (eCLASS) reduced the log size to approximately 58% less than CLASS log encryption, and edge-node CPU usage was also reduced from 14% to 28%.

Keywords: security; edge cloud; digital forensics; integrity; confidentiality

1. Introduction

According to International Telecommunications Union Telecommunication (ITU-T) Study Group 13 (SG13) [1] that is a group of international standardization organization that establishes cloud computing related standard technologies, an edge cloud is defined as "cloud computing deployed to the edge of the network accessed by cloud service customers (CSCs) with small-capacity resources enabling cloud service". The edge cloud, which provides various computing services based on the advantages of edge computing, has recently received considerable attention as a new computing paradigm. Gartner, the world's leading research and advisory company, mentioned "cloud to the edge" as one of its top 10 strategic technology trends for 2018 and included "empowered edge" in its 2019 list. The emergence of the edge-cloud paradigm has generated active efforts to redesign the network, increase coverage, boost network capacity, and cost-effectively bring content closer to the user.

The edge cloud, which brings services close to customers, is less manageable and secure than conventional cloud-computing environments because edge nodes are closer to users than edge-cloud managers. For example, malicious users or attackers might attack edge nodes, man-in-the-middle (MITM) attacks enable information modulation and deletion, and Rogue Gateway and Rogue Data Center attacks are disguised as normal edges between data center and users [2–4]. In fact, in 2017, attackers hacked into a thermometer installed in the aquarium of a casino hotel and then infiltrated the

casino network. In May 2018, a company website was suspended for four days after an Internet of Things (IoT) device containing routers, security cameras, and digital video recorders was attacked. The number of hacking attempts through these service end-points is increasing, and the collection of forensic data for the enhanced security of end-points and the investigation of security incidents has become important.

1.1. Problem Statement

Digital forensics in the edge cloud has gained importance, as edge nodes have become the target of security attacks [5]. As a result, digital-forensic experts need a forensic-data-collection system for edge-cloud environments because edge nodes are intercepted, manipulated, and deleted by attackers, which makes it difficult to collect forensic data from edge nodes. However, since the standardized architecture and definitions for the edge cloud and the form of services for the edge cloud have not yet been clearly defined, the structure of edge-cloud services should be defined before proposing a logging system for digital forensics for those services.

Edge-cloud services are actually provided by using edge nodes unlike a conventional cloud service. The edge nodes that provide services are weak on security management because they are geographically separated from the cloud. This geographic separation of management is an attack target from malicious users or attackers, and it is difficult to safely keep and manage the log data of edge nodes, as shown in Figure 1. In addition, edge nodes with limited computing resources have potential problems on log-data collection, such as log-generation failures and incorrect logging by computing overhead.

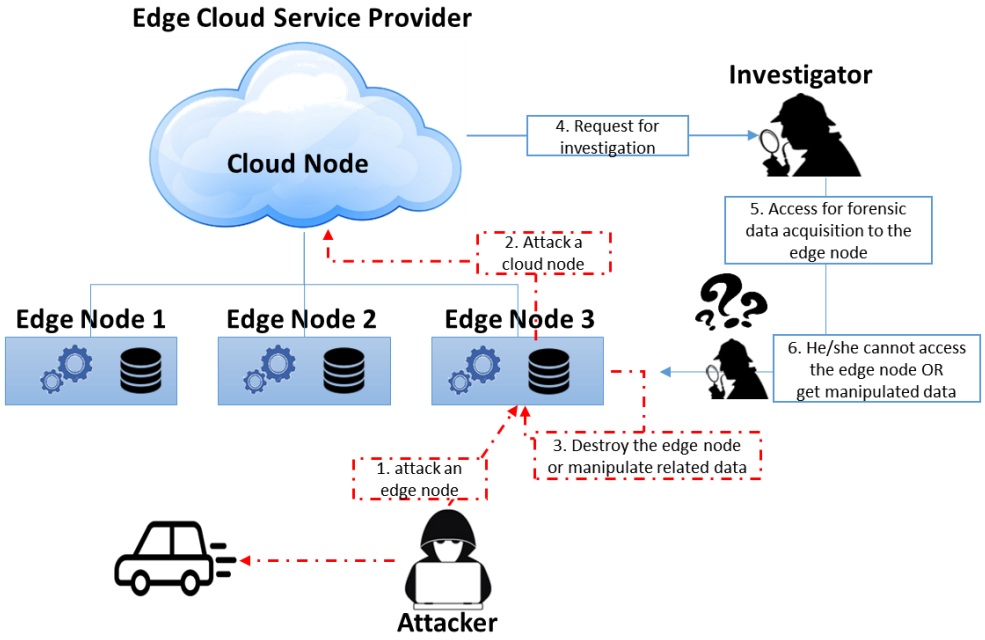


Figure 1. Attack and investigation scenario in edge cloud.

Existing cloud logging systems encrypt and store log data generated in the cloud, and a cloud service provider (CSP) manages the log data. Because these logging systems are stored and managed by the CSP, malicious CSPs can modify log data at any time. In addition, non-cooperation with forensic-data collection for security-incident investigations increases the difficulty in investigating incidents. Therefore, a logging system is needed to collect log data without cooperation from the CSP.

1.2. Contributions

In edge-cloud environments, logging systems need to identify and solve these problems. Therefore, we proposed a new logging system that takes into account the characteristics of the edge cloud to solve these problems and our contributions are as follows.

- We propose a new secure logging scheme in consideration of the edge-cloud environment. This logging scheme provides log-data confidentiality and integrity using log-data segmentation, distributed storage, and multi-index-chain (MIC) techniques for solving edge-node problems such as low computing resources and the geographically separated management from the owner eCSP.
- We introduce the MIC technique and distributed-storage cluster to acquire forensic data without the cooperation of the corresponding service provider. The index files include information of the distributed log block being shared with MIC peers through the MIC network. Therefore, investigators can collect the related log blocks based on the index files and distributed-storage cluster (DSC).
- We outline a security analysis and performance evaluation that prove that the security of our scheme improved upon existing logging schemes, and that our scheme could reduce the log processing time and required storage size.

The remainder of this paper is organized as follows: Section 2 discusses related work; Sections 3 and 4 define threat models, security properties, and propose an edge-cloud logging scheme for digital forensics; Section 5 verifies the proposed logging scheme through performance and security evaluation; finally, Section 6 concludes the paper and outlines plans for future work.

2. Related Work

This chapter provides an overview of recent definitions of the edge-cloud environment and structure. Moreover, the logging system and logging services used for digital forensics in traditional cloud computing are examined.

2.1. Edge Cloud

The roots of the edge cloud go back to the late 1990s when Akamai [6] introduced content delivery networks (CDNs) to accelerate web performance. A CDN uses nodes at an edge close to users to prefetch and cache web content. These edge nodes can also perform content customization, such as adding location-relevant advertising and video content. The edge cloud generalizes and extends the CDN concept by leveraging the cloud-computing infrastructure.

The edge cloud aims to quickly provide cloud services to users through near edge nodes. According to the standard draft of ITU-T SC13 [1], the edge cloud is defined as follows:

> "The edge cloud is deployed at the edge of the network accessed by CSCs, and has small resource capacity. The edge cloud requires specialized hardware resource on purpose, and resources in the edge cloud are constrained due to limitations of space or power."

The edge cloud defined by the standard draft is one of the distributed cloud architectures that consist of a core cloud and an edge cloud. The edge cloud provides services to edge nodes that are close to the user by launching user-demanded services. Moreover, if there is no service request, the launched services of the edge node are deleted or suspended so the limited computing resources of the edge node are effectively managed.

The edge cloud provides services through edge nodes, and most service logs are also recorded in edge nodes. Therefore, in an edge-cloud environment, collecting logs for digital forensics requires working around edge nodes, while limited computing resources mean less computing load for log collection.

2.2. Conventional Cloud Logging Systems

Various studies on logging systems in cloud computing are underway in the area of digital forensics. Among them, Secure Logging-as-a-Service (SecLaaS) [7–10] and the Cloud-Log Assuring-Soundess and Secrecy Scheme (CLASS) [11] are representative cloud logging systems, taking into account cloud forensic challenges such as data volatility, multitenancy in the cloud, and ensuring user privacy.

Secure Logging-as-a-Service collects data on the basis of network logs to identify potential intrusions into virtual machines (VMs) within the cloud; it utilizes log accumulators to ensure the integrity of log data with log-chain technology. In addition, by using the proposed Bloomtree technology in SecLaaS, the accumulated hashed log values are quickly retrieved, showing faster performance than existing logging technologies. However, privacy issues occur because SecLaaS allows investigators to access and read user logs. Log-chain technology ensures integrity by sequentially accumulating log entries, which can cause performance degradation during large log-entry creation and, thus, does not provide multiple log sources, such as a mobile edge cloud [12–16] and fog computing [13,17]. Furthermore, Bloomtree [8] has a low probability of false positives, which are inherent in search failure.

By applying content-concealment technology, CLASS has complemented user privacy and CSP trust issues, which are the limitations of SecLaaS. The user can check logs generated by the CSP, and sensitive information is encrypted with the user's public key. The CSP then publishes proof of past log (PPL) through the log accumulator based on encrypted logs to ensure the integrity of log data and user privacy. However, in the process of storing the log entry, the user should encrypt the log entry with their public key, but CLASS has multiple log-source and false-positive issues, like SecLaaS.

Therefore, since SecLaaS and CLASS cannot support multiprocessing for multiple edge nodes and do not take into account edge-cloud vulnerabilities, we need a logging system that takes into account the edge-cloud environment and can be used as forensic data. This study proposes an edge-cloud logging system for digital forensics, taking into account the characteristics of the edge cloud, such as weak security and edge nodes with limited computing resources.

2.3. Data Protection Techniques

In digital forensics, protection forensics data and data integrity are one of most important methods. To protect log data in a logging system, almost all logging systems encrypt log data using an encryption key, such as Advanced Encryption Standard (AES), Data Encryption Standard (DES), and RSA. Raja Sree [10] suggested a secure logging scheme for forensic analysis in a cloud. The scheme also encrypts log data using 2048 bit RSA. Numerous studies [18–20] have attempted to protect user privacy based on asymmetric encryption in a cloud logging system. However, data encryption needs enough computing resources.

Lei Yang [21] and Selvakumar [22] suggested a new cloud logging system using a data partitioning method for digital forensics. However, they used data partitioning to improve the processing of data streams and to protect the security of data stored in the cloud storage.

Yasir Karam [23] and Muhammad Asim [24] designed access control for an objective-driven programming model through provisioning assurance auditing (PAA) which provides a secured separated abstraction layer for cloud users. Urmi Priyadarshani Das [25] proposed an intrusion detection system for cloud and fog computing. To protect user data, the system provides decoy data to intruders and decrypted data to normal users after a user identification procedure. Ximeng Liu [26] addressed identity/attribute-based cryptography, security, and privacy challenges, user-privacy preservation, and data-protection methods for the IoT and fog computing. However, addressed methods such as asymmetric encryptions and full homomorphic encryption, cause high computing overhead.

2.4. Ensuring Data Integrity Technique

To ensure data integrity, most logging systems have a log accumulator or blockchain network. Conventional logging systems, such as SecLaaS and CLASS, use a log accumulator for log integrity. EI

Ghaouani [27] suggested a blockchain and multiagent system using a blockchain network for data integrity. Konstantinos Rantos [28] addressed blockchain-based consent management for personal data processing in the IoT ecosystem. Noshina Tariq [29] researched security challenges in fog-enabled IoT applications, including blockchain. These studies ensure the integrity of sensitive and big data in the cloud. We needed to redesign the blockchain network to consider features of edge computing.

3. Edge-Cloud Threat Model and Security Properties

This section presents terms and their definitions, and edge-cloud service models to help understand how log data are collected for digital forensics in the edge cloud. Security threats to log data and security features used for protection are described below.

3.1. Terms and Definitions

Table 1 shows the description of Edge-Cloud-Log Assuring-Secrecy Scheme (eCLASS) terms.

Table 1. Description of Edge-Cloud-Log Assuring-Secrecy Scheme (eCLASS) terms.

Terms	Description
Log (Log entry)	The log is a network log, process log, registry log, application log, or any customized text that meets the requirements of being stored for digital forensics.
Index	An index is created by storing segmented logs in a distributed-storage cluster and aggregating the stored information. Users and investigators can recover distributed and stored segmented logs based on information in the index.
Multi-index (MI)	A MI is a set of uploaded indices collected by the investigator during a certain period of time.
MI chain (MIC)	An MIC is information that prevents deletion and exchange of, and tampering with, specific indexes. MICs share information with participants in a multi-index network to ensure the integrity of data in the multi-index. A new MIC is created including previously published MIC information.
Edge cloud service provider (eCSP)	An edge-cloud service provider is a service provider that uses an edge node to deliver cloud services.
Edge node	An edge node deploys cloud services from an eCSP and provides services to users and Internet of Things (IoT) devices. An edge node is close to users and IoT devices, and has limited computing resources.
Edge-cloud service user	An edge-cloud service user is an end-user who receives services through edge nodes.
Investigator	An investigator can conduct forensic investigations into security accidents in the event of a security accident. Moreover, investigators manage and control multi-index chain networks.
Distributed-storage cluster(DSC)	A DSC is a group of storage services that can store segmented logs and provide storage services through secure application programming interfaces (APIs).

3.2. Edge-Cloud Service Models

An eCSP can provide edge services to users with various service models. For example, eCSP supports the mobility of users and devices by using edge nodes with limited computing capacity. As shown in Figures 2–5, edge service models are defined and described as four types of service models that can be provided to users in an edge-cloud environment.

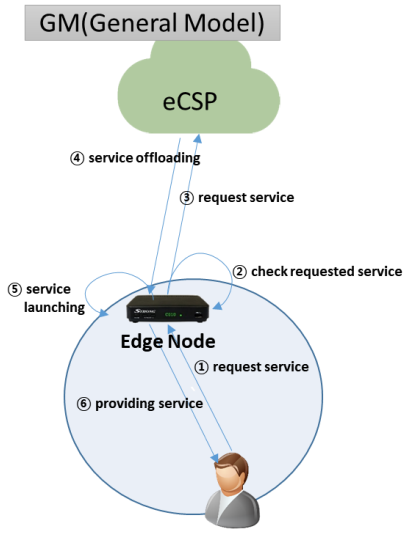


Figure 2. General model.

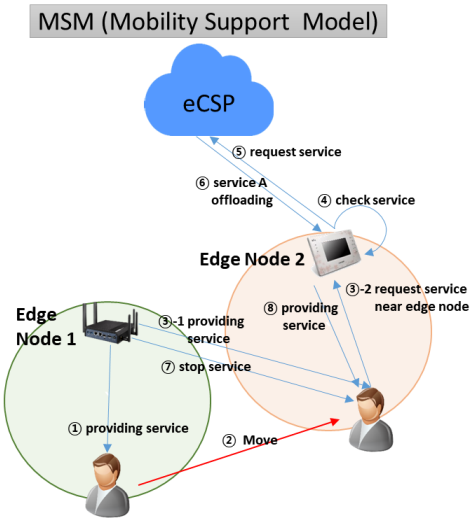


Figure 3. Mobility-support model.

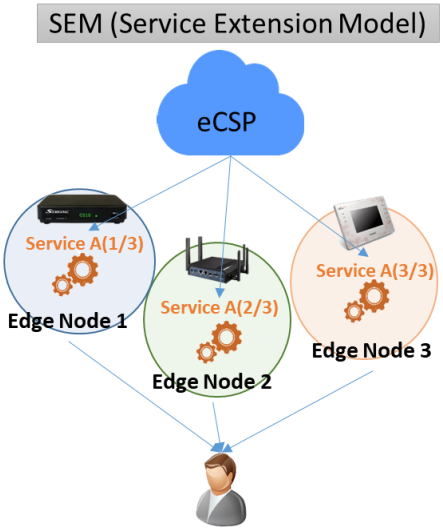


Figure 4. Service-extension model.

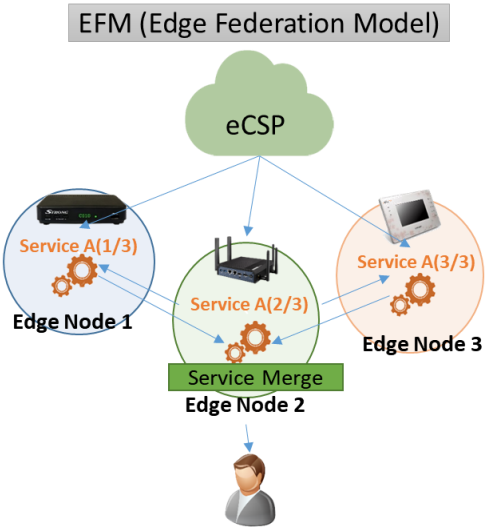


Figure 5. Edge-federation model.

3.2.1. General Model

As shown in Figure 2, the general model (GM) is a conventional edge-cloud service model. If a user requests a cloud service on the nearest edge node, and the service is not launched in the edge node, the edge node requests the user's service from the eCSP, and then downloads and launches the service to provide the service to the user. If the requested service is already launched, it quickly provides the service to the user without requesting it from the eCSP. In such cases, log information is recorded only on edge nodes.

3.2.2. Mobility-Support Model (MSM)

As shown in Figure 3, the MSM is a service model that provides seamless cloud services to mobile users or devices. When a user or device is moved from Edge Node 1 to Edge Node 2 to efficiently realize a service, the edge node that is providing the service is replaced by a close edge node to ensure quality of service (QoS). In this case, logs are written to multiple edge nodes depending on the path of the user and device.

3.2.3. Service-Extension Model

As shown in Figure 4, the service-extension model (SEM) is a service model that simultaneously processes across multiple edge nodes when the network environment is proper and computing resources of the requested service are higher than one edge node. For example, when a user uses a virtual-reality (VR) service on a 5G network, the eCSP can divide it into three VR screens and provide images from each of the three edge nodes because network speed is sufficient but exceeds edge-node performance. In this model, a service log is generated in every edge node.

3.2.4. Edge-Federation Model

As shown in Figure 5, the edge-federation model (EFM) is a service model that provides services by dividing tasks into different edge nodes and integrating the result into the main edge node (Edge Node 2) because the computing resources of a single edge node are lower than the computing resources of the requested service. This model handles task distributions, such as SEM, but does not directly connect all edge nodes to the user (or device). Only Edge 2 directly provides the requested service to the user, but Edge Nodes 1 and 3 are only connected with Edge 2. In this case, because log information for Edge Nodes 1, 2, and 3 is different, additional information is required to identify logs of the same service for Edge Nodes 1, 2, and 3.

3.3. Threat Models

The proposed scheme was designed on the basis of the "zero-trust network" policy. This scheme does not trust any party from eCSPs, investigators, DSCs, and users. In addition, the proposed scheme requires functions for privacy protection and forensic data.

In an edge-cloud environment for forensic data, logging systems need to consider potential security threats, such as log corruption and forensic-data contamination, based on the characteristics of the edge-cloud environment. In an edge-cloud environment used by various users, malicious users can undermine or tamper with log data, thereby disrupting forensic investigations. Models that threaten the integrity of these logs should be defined as follows, as well as countermeasures on edge-cloud logging systems. Moreover, Table 2 shows the comparison of threat models among SecLaaS, CLASS, and eCLASS.

Table 2. Comparison of threat models among SecLaaS, CLASS, and eCLASS.

Threat Models	SecLaaS	CLASS	eCLASS
Log modification	O	O	O
Privacy violation	O	O	O
Ownership repudiation	O	O	O
CSP service confidentiality violation	X	X	O
Edge-node tempering	X	X	O
Computing overhead in edge node	X	X	O

Log modification: A dishonest CSP can modify logs generated by edge nodes before the log-data verification process takes place. In addition, malicious investigators may conduct log collusion with users and CSPs to modify relevant logs in order to deny or mitigate cybercrimes. These log modifications include adding invalid entries, deleting important entries, modifying existing entities, and arranging log entries. These modified logs may mislead investigators or hide some misbehavior. In edge clouds, because edge nodes are geographically separated from CSPs when logs generated by edge nodes are sent to CSPs, an attacker can change normal logs through a log-modulation attack, such as an MITM attack.

Privacy violation: Leakage of a log file can reveal privacy-related information that includes user identity. Even with cryptography, cloud employees and dishonest investigators who have key decryption can transfer the log to an entity, which is how privacy is violated.

Ownership repudiation: Because cloud servers have information about a large number of users, malicious cloud users can submit to other users' activity logs. It can also be denied that the CSP did not record the investigated logs. In some logging systems, the CSP creates and stores logs for all users, but users cannot figure the logs out. As a result, this increases users' suspicion about log integrity and reliability and the CSP can deny the log.

The CSP service confidentiality: In some cloud logging systems, logs created on cloud servers should be read to users and the service provider only, but other CSPs, users, and investigators can easily access logs that include service information, user status, and user-access rate and area. Because the service log is also confidential CSP information and an asset, access by other parties (other users, CSPs, investigators) violates the confidentiality of the service.

Edge-node tempering: Geographically outside the scope of CSP management, malicious users or attackers can physically access and manipulate stored log information.

Computing overhead in edge nodes: As edge nodes have limited computing resources, they can be overloaded by providing many functions on edge nodes. This overload can cause service logging delay, incorrect logging, and non-logging.

3.4. Security Properties

Correctness: Logging systems for digital forensics should have accurate logging of services, and log information should prove that there has not been any tampering. Distributed logs should show

that they are related to each other and should record which actions the requested service handled on which edge node. Finally, it is necessary for users, investigators, and auditors to verify that logs are correctly recorded.

Tamper resistance: In CLASS and SecLasS, encrypted log DB, PPL, and an accumulator are kept by the CSP so logs can be modified at any time in collusion with malicious users. In addition, log modulation and regeneration are possible even after PPL if CSPs, malicious users, and investigators collude. Therefore, it is necessary to store separately encrypted log DB and PPL, and check modulation even if CSPs, users, and investigators are involved.

Verifiability: Since the proposed scheme generates logs from distributed edge nodes in a zero-trust network, verification steps are necessary to ensure log accuracy, consistency, and integrity. An edge-cloud logging system should thus be able to verify data modulation and corruption through verification of generated logs.

Confidentiality and privacy: Logs are information about the services that eCSP provides to users. The information about eCSP services and users' personal information may be exposed if log information generated by the eCSP is easily accessed by investigators, auditors, and other users. Therefore, only authorized users should be able to access authorized log information. In addition, low-overhead data-protection methods are needed to protect confidentiality and privacy, considering the limited computing resources of the edge cloud.

Admissibility: Logs generated in edge clouds should be utilized as forensic data and should be acceptable in court. For legal admissibility, it should be possible to verify that there has not been any log tampering, corruption, or omission in the process of log-data generation, management, and verification. Moreover, log collection for edge clouds should be done on a legitimate security-policy basis because illegal log-data collection violates admissibility.

4. Proposed Scheme: eCLASS

As shown in Figure 6, eCLASS consists of an eCSP, users, investigators who conduct forensic investigations after a security incident and manage the MIC network, and DSC for storing the segmented log data.

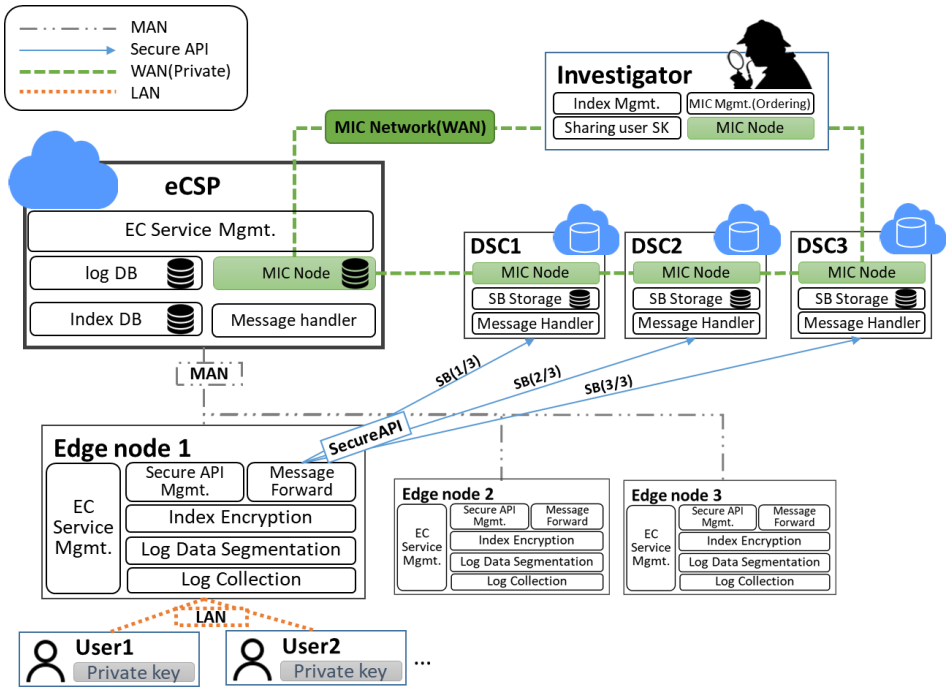


Figure 6. Overview of eCLASS.

4.1. Overview

The eCLASS ensures log-data integrity and protects user privacy/service confidentiality for logs generated in edge nodes outside the geographic management scope. The eCLASS uses log segmentation and a distributed-storage technique instead of log encryption, taking into account the computing resources of limited edge nodes. It also ensures log-data integrity by sharing the index (including log segmentation and distributed-storage path information) with private network participants for distributed-storage logs.

The following part of the paper describes each role and performance capability of edge nodes, eCSPs, investigators, and distributed-storage clusters.

Edge nodes: As edge nodes provide services using a deployment service image close to users based on virtualization technologies such as containers and virtual machines, logs for cloud services are generated in the edge nodes. Logs generated in edge nodes do not keep logs in edge nodes because of the manageable vulnerability of edge nodes, and perform log segmentation and distributed storage for the sake of confidentiality and privacy.

Existing logging systems protect data confidentiality by storing encrypted logs and local repositories. However, edge nodes have limited computing resources, which cause overload issues due to the fact of log encryption. Therefore, log segmentation is performed with units that are unrecognizable to protect the confidentiality of log data and upload the group of log-segmentation blocks to DSCs over secure APIs. In addition, since recovering log data from edge nodes can result in tampering and loss by attackers, the edge node creates an index file that includes the information of segmented logs stored in a DSC.

The generated index file is encrypted with users' public key to protect service confidentiality and user privacy from malicious investigators and other DSCs. The encrypted index file is sent to the investigator who can preserve data integrity. The delivered encrypted index file is managed as an MI group through the MIHeader, which is issued for a certain period of time. This process leaves no log files and index files on edge nodes, and allows log segmentation and distributed storage to protect the confidentiality and privacy of data from attackers.

The eCSPs: An eCSP manages its own edge nodes, cloud services, and service images to launch services on edge nodes that are close to the user. An eCSP communicates with edge nodes over a metropolitan area network (MAN).

Investigators: Investigators have the right to investigate security accidents and collect log data regarding services and users from eCSPs if there is a search-and-seizure warrant related to a security incident. An investigator manages the MIC network, collects an encrypted index file from distributed edge nodes, publishes an MI that groups several encrypted index files on the MIC network, and then shares the MI with all network participants to ensure the integrity of encrypted index files. In addition, if log collection is required for a security-incident investigation, the investigator can collect distributed stored segmented logs, receive the user's private key, and recover those segmented log files. Investigators share the MIC that includes information of many indices with MIC network participants over a wide area network (WAN).

DSCs: DSCs are repositories where logs generated from each edge node are stored distributed, and the segmented logs are stored through secure APIs. These logs should be viewed by anyone as read only.

4.2. eCLASS Specification

This section describes the log-data collection process and the verification process of collected logs based on edge-cloud characteristics. Table 3 describes the notation for the proposed scheme.

Table 3. Notation summary.

Notations	Description
eLE_i^n	the i-th log entry in the n-th edge node $H(eLE_i^n)$: hash file of the i-th log entry in the n-th edge node
f_s	function for segmentation of $eLE_i^n+H(eLE_i^n)$
f_p	function for patitioning segmentation blocks
SB_j	set of n-th segmentation block of a log entry $SB=\{SB_j \mid j=1, \ldots, w\}$
GSB_k	set of lth group of segmentation blocks $GSB=\{GSB_k \mid k=1, \ldots, l\}$
SBI_m	information on m-th partition block for log recovery
IDX_m	file that includes SBI_m and PathGSBs
$EIDX_m$	encrypted file of IDX_m
$PathGSB_k$	path information that GSB_k has stored in a DSC
MI_g	g-th multi-index block including g-th MIHeader, EIDXs, g-th MIC.
$MIHeader_g$	identification value of MI_k

4.2.1. Log Collection Procedure

The process of collecting logs from the proposed eCLASS and ensuring log integrity through an MIC network is described in the process flow of Figure 7 below.

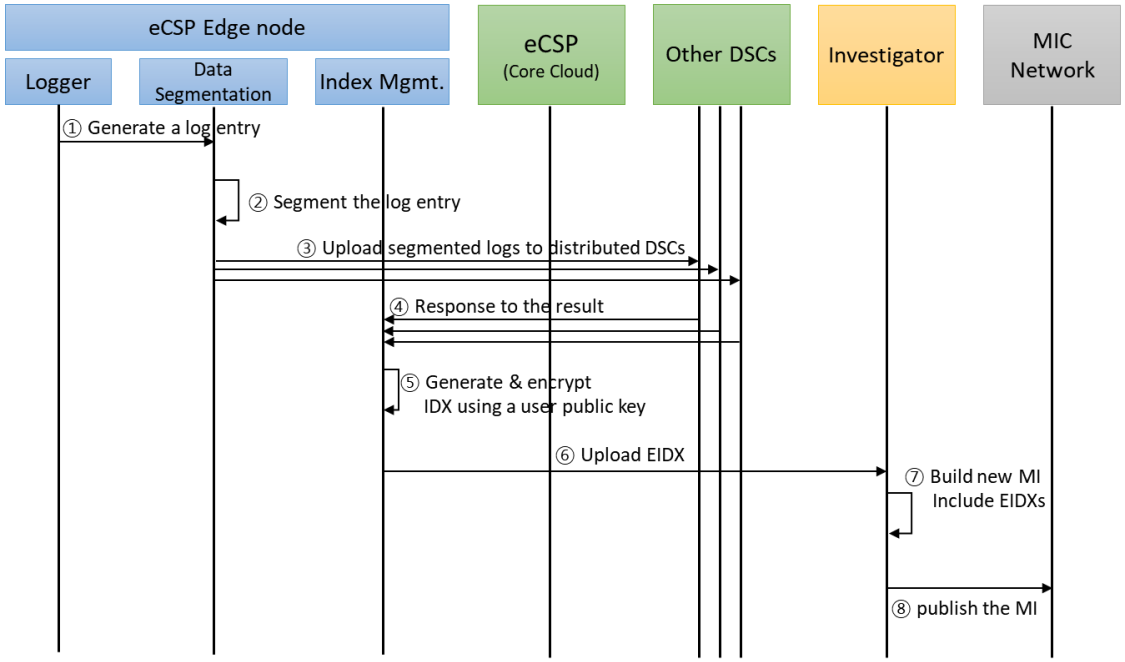


Figure 7. Process flow of eCLASS in edge cloud.

- Log Entry Generation

The logger collects logs stored in a specific directory on an edge node. The logs are stored in a specific format, and contain service information for digital forensics. For eCLASS, eCSP ID (eCSP_ID), edge node IP (Edge_IP), user information (User_Device_IP, User_Device_MAC, User_ID), log time (LT), and service information (Service_URL) are used.

eLE_i^k = ith log entry generated in the kth edge node.
eLE_i^k = <eCSP_ID, Edge_IP, User_Device_IP, User_Device_MAC, User_ID, LT_i, $Service\_URI_i$>

- Log Entry Segmentation

To protect the confidentiality and privacy of eLE_i^n delivered by the logger, the existing logging system performs log encryption, but considering the low capacity of the edge node, log-segmentation technology is used. Log segmentation divides data into blocks that cannot be identified as information by a person. In addition, it is impossible to identify two or more blocks, as they are listed non-continuously as shown in Figure 8. Data-segmentation methods include the round-robin, range-segmentation, hash-segmentation, and list-segmentation methods.

$$f_s\left(eLE_i^n + \mathrm{H}\left(eLE_i^n\right)\right) = \left\{SB_1 \| SB_2 \|, \ldots, \left\| SB_j \right\}\right. \tag{1}$$

$$\begin{gathered} f_p\left(\left\{SB_1 \| SB_2 \|, \ldots, \left\| SB_j \right\}\right\}\right) = \left\{GSB_1 \| GSB_2 \|, \ldots, \left\| GSB_k \right\}\right. \\ \text{where, } GSB_k = \left\{\text{A group of } f_p\left(SB_1 \| \mathrm{SB}_2 \|, \ldots, \left\| \mathrm{SB}_j \right)\right\}\right., \ 1 \le k \le j \end{gathered} \tag{2}$$

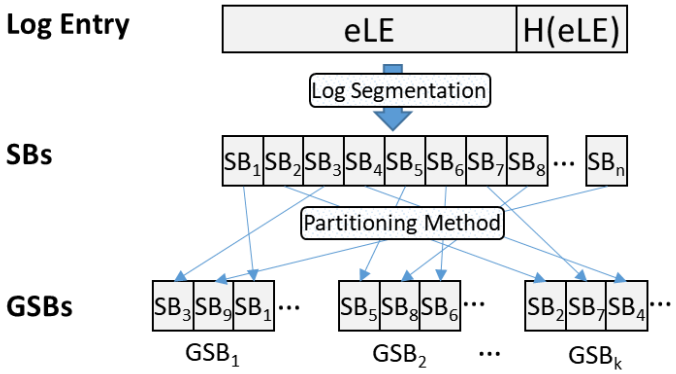


Figure 8. Process of log segmentation and partitioning.

- Sending to DSCs

Log segmentation transfers the segmented log data (SB_j) to the contracted DSC and stores the segmented log data (SB_j) in each storage cluster.

- Response to Storage Complete

The storage that stores the segmented logs sends the storage path ($PathGSB_k$) to the corresponding edge node. The edge node that receives the storage path generates an index file (IDX) with segmented log information (SBI_m). An index file, including SBI and a distributed-storage path, is shown below.

$$IDX_m = \{SBI_m \,\|\, PathGSB_1 \|\, PathGSB_2 \| \ldots \| PathGSB_k\} \tag{3}$$

- Index Generation

To ensure the integrity of index files, index files are shared with other users. Moreover, log recovery is possible with shared index files and segmented data blocks. Therefore, an index file is encrypted by using the user's public key to protect user privacy and service-provider confidentiality, and to access logs for that user. In addition, user ID and generation time of the index file (denoted to TL_m) are recorded together to facilitate retrieval:

$$EIDX_m = \left\{\text{UserID} \,\|\, TL_m \,\|\, Enc_{pk_{user}}(IDX_m)\right\} \tag{4}$$

- Uploading $EIDX_m$ to an Investigator

In traditional cloud logging systems SecLaaS and CLASS, cloud service providers own an accelerator, encrypted log data, and PPL. However, eCLASS shares index information with MIC network participants to prevent the service provider from tampering, and malicious loss. In addition, even if it is modulated or deleted, *EIDX* can be used to recover modulation verification through *EIDX* stored by other participants. Thus, edge nodes are uploaded to investigators for the integrity of *EIDX*.

- Multi-Index Generation

Investigators combine several *EIDX*s received over a period of time to publish a multi-index and include the hash values (MIC) of the previous multi-index to verify its modulation:

$$MI_g = \left\{ MIHeader_g \left\| (EIDX_m + EIDX_{m+1} \ldots) \right\| MIC_g \right\} \quad (5)$$

- Multi-Index Publication

Through ordering, the investigator publishes MI_k and shares it with the MIC network to ensure value integrity, and enable users to search and view their *EIDX* through a nearby MIC network participant.

As shown in Figure 9, the log-collection procedure of eCLASS can be divided into two main phases, as in CLASS. However, subprocedures are totally different. The first is to segment the log data and the other is to share the index file with the MIC network. The first algorithm (Algorithm 1) shows how to divide log entries into unidentifiable sizes and how to partition them to be non-continuous. This algorithm is intended to ensure the confidentiality of log entries, and requires fewer computing resources than encryption. The second algorithm (Algorithm 2) is the MIC network sharing method for index files. This method allows encrypted index files to be shared with network participants that can check if the index is tampered with; it also serves to prevent malicious index addition and deletion.

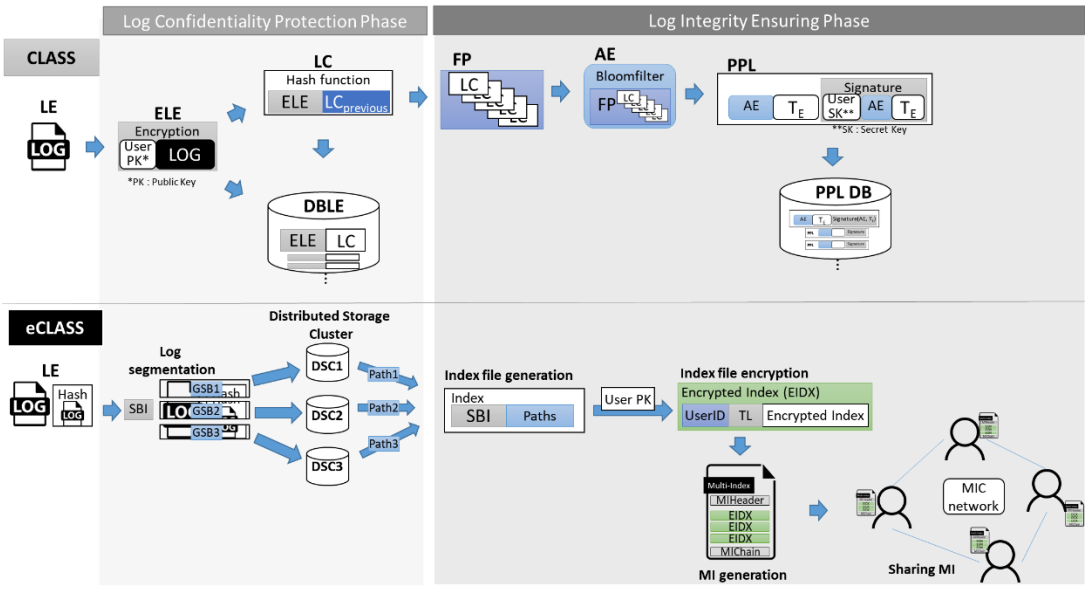


Figure 9. Comparison of CLASS and eCLASS log processing.

Algorithm 1. LogCollection pseudocode for log confidentiality

Input : eLE, H(eLE)
Output : SBI, $PathGSB_{1\ to\ K}$

LogCollection(edge log entry eLEs, H(eLEs))
Int s = LogSegmentation size;
Int d = The number of DSCs;

foreach edge node **do**
ELE = eLE + H(eLE);

/*log segmentation part*/
for (i = 1, length(ELE), i + s)
SB[i] = logSegmenation(ELE, s); /*segment the ELE by LogSegmentation size*/
end for;

/*log partitioning and distribution storage part*/
for (k = 1, d, i + 1)
GSB[k] = Partitioning(SB[], d);
/* partition the SB[] by the number of DSCs according to the partitioning method*/

Generate SBI ← add.info.Partitioning(SB[], d);
send GSB[k] to DSCs over Secure API;
get $PathGSB_k$ from corresponding the DSC;
end for;
end foreach;
end;

LogCollection of Time Complexity: T(n)
T(n) = each edge node(log segmentation part + log partitioning and distribution storage part)
$= 2 + n(2n + 1 + 2n + 3) + 1 = 4n^2 + 4n + 4$
$\therefore T(n) = O(N^2)$

LogCollection of Space Complexity: S(n)
S(n) = each edge node(log segmentation part + log partitioning and distribution storage part)
$= 2 + n(1 + n + 1 + n + 1 + n) = 2 + n(3n + 3) = 3n^2 + 3n + 2$
$\therefore S(n) = O(N^2)$

Algorithm 2. IndexSharing pseudocode to generate and share multi-index

Input : SBI, $PathGSB_{1\ to\ K}$
Output : MI

IndexSharing(Segemtation info SBI, $PathGSB_{1\ to\ K}$)

// Encrypted Index generation part
foreach edge node **do**
IDX = (SBI || $PathGSB_{1\ to\ k}$);
EIDX = UserID + TL + encrypt(IDX).using_user's_publickey;
send EIDX to Investigator;
end foreach;

// MI generation and sharing part
for investigator **do**
MI = MIHeader+Ordering($EIDX_1$, $EIDX_2$, ... , $EIDX_n$) + MIC(Hash($MIC_{previous}$+MIHeader));
Publish MI;
end for;
end;

IndexSharing of Time Complexity: T(n)
T(n) = each edge node(Encrypted Index generation part) + MI generation and sharing part
= n(3 + 2) = 5n
$\therefore$ T(n) = O(N)

IndexSharing of Space Complexity: S(n)
S(n) = each edge node(Encrypted Index generation part) + MI generation and sharing part
= n(1 + 1) + n = 3n
$\therefore$ S(n) = O(N)

The LogCollection and IndexSharing algorithms are the most important algorithms in eCLASS. The log collection algorithm describes the process of log data segmentation and distribution storage to protect the confidentiality of log data. Also, the IndexSharing algorithm describes the index generation and multi-index sharing methods to ensure the integrity of log data. Since eCLASS operates on distributed edge nodes, the time/space complexity of the LogCollection and IndexSharing algorithms are $O(N^2)$ and O(N). As a result, as the number of edge nodes increase, time/spatial complexity increases.

4.2.2. Log Verification Procedure

In the event of a traditional cloud security incident, an investigator with investigative authority first requests an integrity-based log from the CSP in order to collect forensic data and analyze the logs. If the integrity of the logs is not verified, they cannot be recognized as forensic data and are not legally valid.

However, this system ensures log integrity using the hash function and verifies the integrity of indices and segmented logs through a distributed-storage cluster and an MIC network. The following paragraphs describe how to verify the integrity of log data and the MI sequence.

- Log Monitoring by Users

Users can check their logs after using edge-cloud services. The eCLASS provides edge-cloud services through edge nodes and encrypts segmented and stored log information through user public keys so it can be stored and shared on the MIC network. Therefore, users can search for and download their own encrypted index information through participants in the MIC network. In addition, log recovery can be performed through the decrypt index using the user's private key to check user logs.

- Log- and Index-Integrity Verification

Edge nodes generate hash values *H(eLE)* for *eLE* with *eLE* generation to ensure the integrity of generated logs. Therefore, *eLE* and *H(eLE)* are created to perform log segmentation and distributed storage.

To verify the integrity of log data, investigators collect distributed segmented data, as shown in Figure 10, and recover log data through an index. Since the recovered data consist of *eLE* and *H(eLE)*, recovered data *eLE* are hashed to check if they match *H(eLE)*. If these data match, the integrity of the log data and index file can be verified.

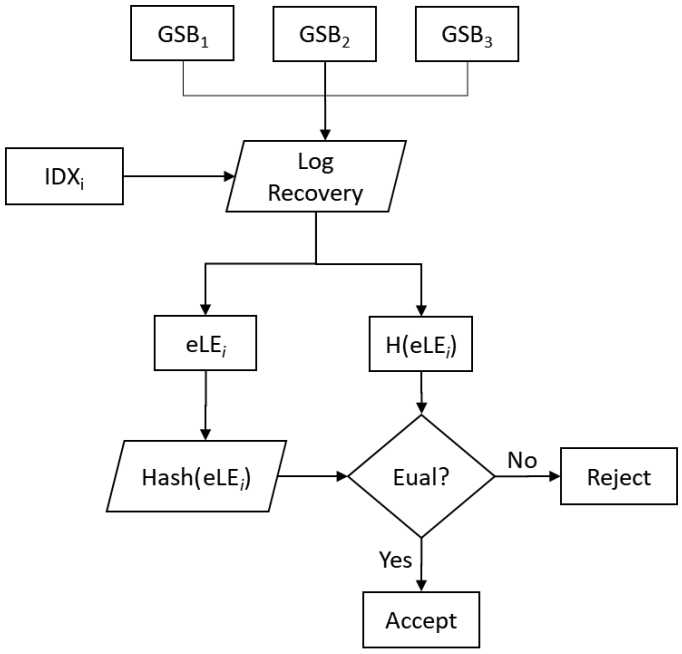


Figure 10. Process of log-integrity verification.

- Multi-Index Sequence Verification

Figure 11 illustrates the MI order verification process, where we verified whether the current MI actually came after the previous MI in the original sequence of MI publication. In Figure 11, MI Header 1 denotes the MI header of the first multi-index and MI Header 2 represents the same for the second MI header.

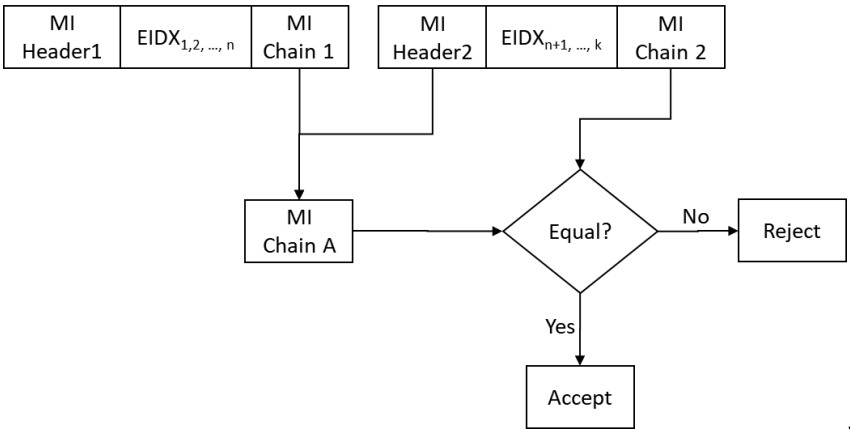


Figure 11. Process of multi-index sequence verification.

To verify the correct order, the auditor calculates the MI chain ($MIChain_a$) from the first MI chain ($MIChain_1$) and the second MI header ($MIHeader_2$) according to the following equation:

$$MIChain_a = H(MIHeader_2, H(MIChain_1))$$

If $MIChain_a$ matches the second MI chain ($MIChain_2$), then the auditor accepts the *MI*; otherwise, the auditor rejects it.

5. Performance and Security Evaluation

This section describes the security evaluation of the proposed scheme based on the security properties in an edge-cloud environment. In addition, the proposed scheme performs evaluation of log collection because it collects logs by employing log-segmentation and partitioning methods, taking into account edge nodes of limited computing capacity.

5.1. Implementation

As shown in Figure 12, edge-cloud environments are implemented using the Virtual Box virtualization program. Edge nodes with limited computing resources evaluate the usage of computer resources to protect the confidentiality of logs. Edge-node types are classified into three categories: low-, medium-, and high-capacity. Each edge node runs an OS with Ubuntu 18.04 LTS 64 version, and the computing-resource specifications are as follows.

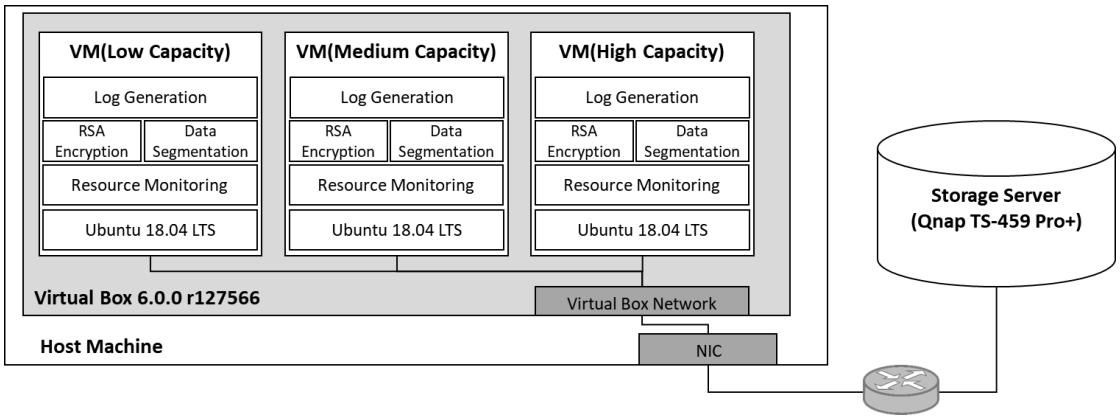


Figure 12. Performance evaluation environment for eCLASS/CLASS log processing.

- Host machine hardware configuration: Intel Core i5-8400 hexa-core CPU, 16 GB RAM, 500 GB SSD, Windows 10 Education used as host operating system.
- Network configuration: The host machine and storage server were configured with an internal network using a 100 M network switch.
- Virtual Box 6.0.0 r0127566 for Windows 10.
- Low-capacity edge node: one core of Intel Core i5-8400 CPU 2.80 GHz, 2 GB RAM, 40 GB
- Local storage.
- Medium-capacity edge node: two cores of Intel Core i5-8400 CPU 2.80 GHz, 4 GB RAM, 40 GB
- local storage.
- High-capacity edge node: four cores of Intel Core i5-8400 CPU 2.80 GHz, 4 GB RAM, 40 GB
- Local storage.

Log generation, data segmentation, and RSA encryption were implemented with Python 3.7.3. The log-confidentiality tools used were 1024 and 2048 bit RSA public-key encryption, and 5 and 10 byte data-segmentation methods.

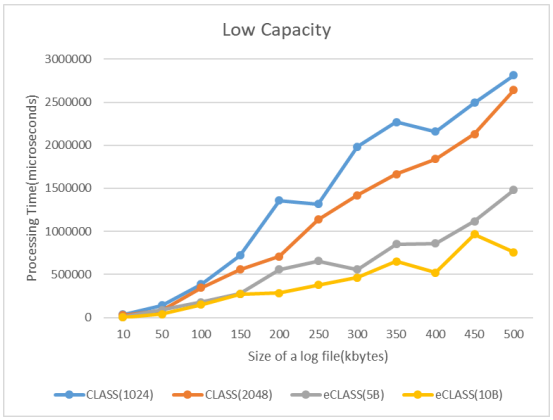
5.2. Performance Analysis

This section presents the performance evaluation of how log confidentiality is protected on edge nodes for eCLASS. Because eCLASS aims to collect log data in an edge-cloud environment with forensically available data, fast data processing and proof of data integrity are important. In addition, an edge node needs to measure the computing overhead of the log-processing phase on edge nodes because of limited computing resources.

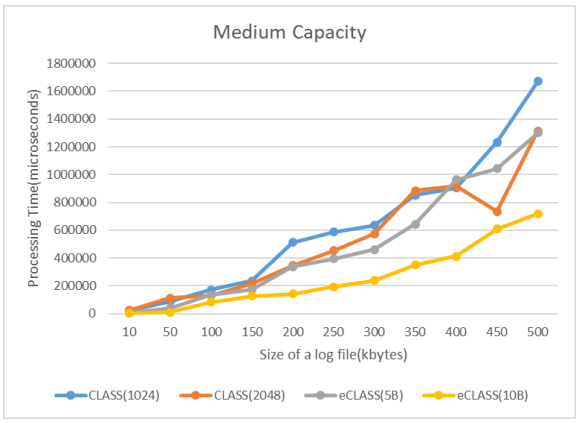
5.2.1. Logging Processing Time

There are two main differences between eCLASS and the CLASS scheme. First, to prevent log contamination, CLASS incorporates asymmetric encryption with an individual user's public key to avoid data leakage. The eCLASS, however, applies a data-segmentation method to divide and partition log data into 5 or 10 bytes for protecting the log data. This study took 10 KB of the log file with almost 200 log entries. The used log files were 10, 50, 100, 150, 200, 250, and 300 KB.

Figure 13 shows the performance analysis of log processing time with RSA encryption and the log-segmentation method. It was found that, for low- and medium-capacity edge nodes, the log-segmentation method was faster than RSA encryption. However, RSA encryption was similar to or better than high capacity and a large log file.

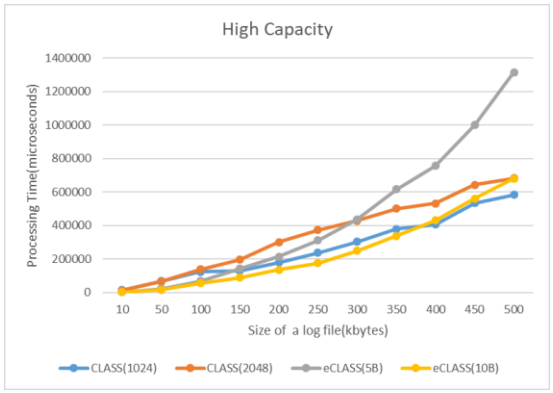


(**a**)



(**b**)

Figure 13. *Cont.*



(**c**)

Figure 13. Log processing-time comparison: (**a**) low computing capacity; (**b**) medium computing capacity; (**c**) high computing capacity.

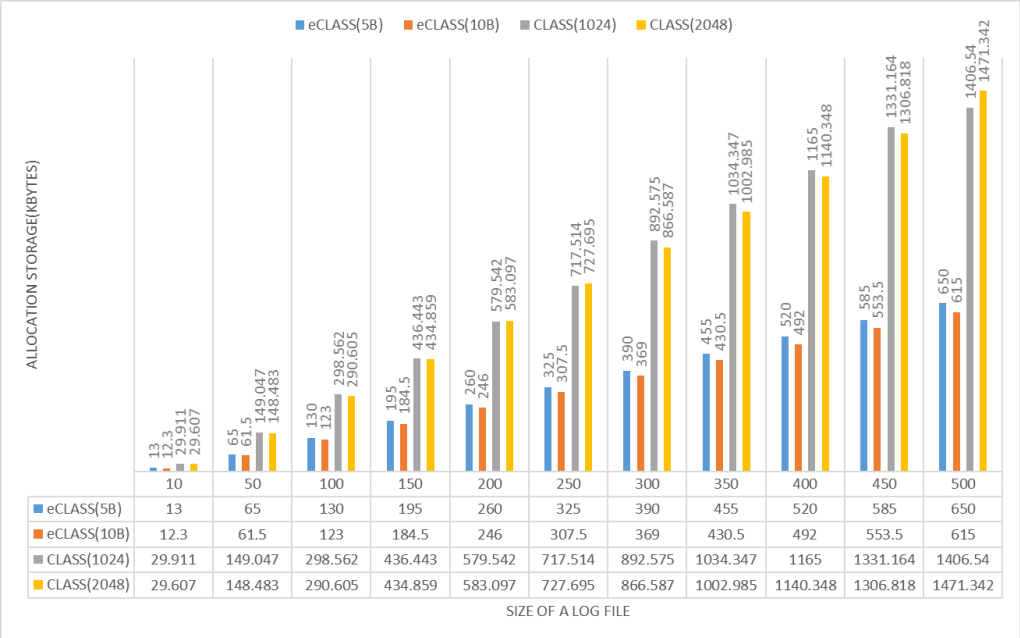
5.2.2. Computing Resource Allocation

This study performed a comparative evaluation of CLASS RSA encryption and the data segmentation of the log-collection phase in edge nodes. This performance evaluation compared CPU storage usage for the log collection of CLASS and eCLASS.

The CPU allocation rate: The study compared the total CPU used for the data-collection phase from 10–500 KB log sizes. As shown in Figure 14, CLASS encryption and eCLASS data segmentation had similar performance in high capacity. However, at low capacity, eCLASS was more efficient: 5 byte segmentation used 19.77% less CPU than CLASS (RSA 2048). In addition, at medium capacity, 10 byte segmentation used 28.78% less CPU than CLASS (RSA 1024), providing a more reliable and cost-effective logging service.

Figure 14. CPU usage comparison for each capacity.

Storage allocation rate: Data size actually increases with data encryption. As shown in Figure 15, CLASS log encryption results in approximately three times larger files and requires more storage. As a result, CLASS needs storage capacity for as long as data increase. In eCLASS, the size of the log data does not increase because eCLASS uses the log-segmentation and partition methods. When segmentation information is included, log data are increased by about 1.3 times.



	10	50	100	150	200	250	300	350	400	450	500
eCLASS(5B)	13	65	130	195	260	325	390	455	520	585	650
eCLASS(10B)	12.3	61.5	123	184.5	246	307.5	369	430.5	492	553.5	615
CLASS(1024)	29.911	149.047	298.562	436.443	579.542	717.514	892.575	1034.347	1165	1331.164	1406.54
CLASS(2048)	29.607	148.483	290.605	434.859	583.097	727.695	866.587	1002.985	1140.348	1306.818	1471.342

Figure 15. Comparison of data size according to confidentiality-protection methods.

5.2.3. Operation Cost

Operation cost is cost evaluation of computing cost and storage cost in edge service modes with CLASS (1024 and 2048 bits) and eCLASS (5 and 10 bytes). The edge service models are GM, MSM, SEM, and EFM. The GM consists of one edge node, MSM consists of two edge nodes, and SEM and EFM consist of three edge nodes. These test factors are as follows:

- Log size generated of each edge node: 200 kb.
- Computing cost (Amazon EC2 [30]): Low capacity($0.0116/hour, t2.micro), medium capacity ($0.0464/hour, t2.medium), high capacity ($0.1856/hour, t2.xlarge).
- Storage cost (Amazon S3 [31]): $0.023/GB.

As shown in Figures 16–18, GM has the lowest operation cost of the three capacities, while SEM and EFM have the highest; eCLASS is more efficient than CLASS in an edge cloud. In the GM, using 5 byte segmentation in low capacity costs $0.00000657. It is approximately 169% more efficient than CLASS using RSA (1024; $0.0000177). Consequentially, we verified that eCLASS is a more economic logging system in edge computing than CLASS.

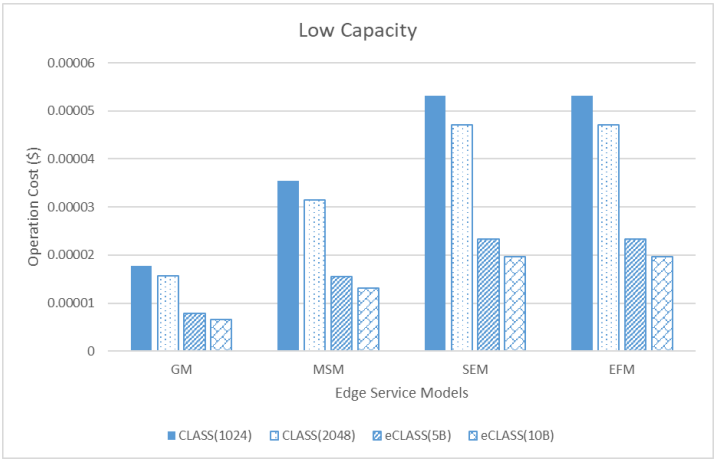


Figure 16. Comparison of operation cost between edge service models in low capacity.

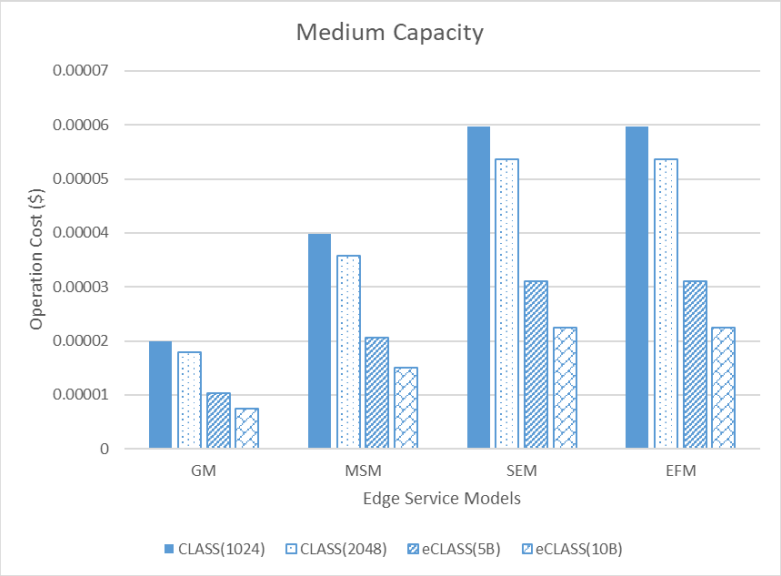


Figure 17. Comparison of operation cost between edge service models in medium capacity.

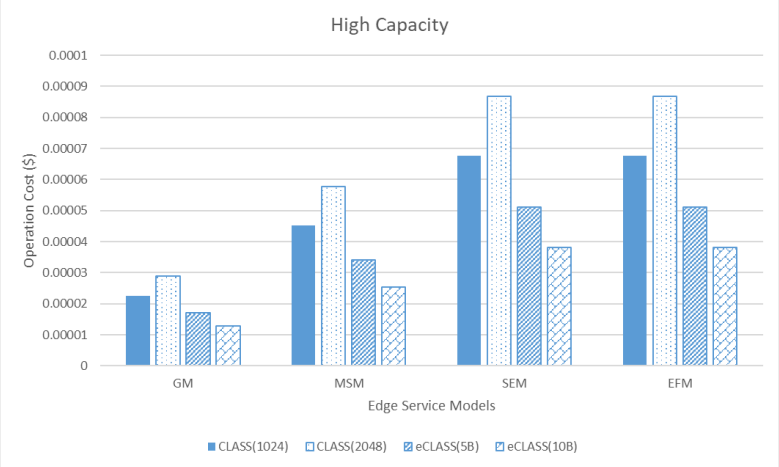


Figure 18. Comparison of operation cost between edge service models in high capacity.

5.3. Summary

We compared the proposed eCLASS and CLASS, cloud logging systems, from three perspectives (log processing time, computing-resource usage, and operation cost from three perspectives), and configured three types of computing capacity of an edge node; low-, medium-, and high-capacity. Moreover, from an operation-cost perspective, we compared eCLASS and CLASS by total operation cost (including log processing cost and storage cost) and four types of edge service models.

The eCLASS's log-data-segmentation and distributed-storage method was faster in log processing time than CLASS's RSA encryption method (eCLASS 10 bytes and CLASS 1024 bit RSA, low capacity (249%), medium capacity (139%), high capacity (8%)) However, as edge-node computing capacity increased, the gap in processing time between data encryption and log partitioning decreased. For this reason, it was important to verify the size of the identifiable segmentation log and find the appropriate segmentation size for edge-node performance. These results can also be seen from a computing-usage (CPU) perspective. In computing usage (storage), CLASS, which performs RSA encryption, needed more storage space than eCLASS, which segments only log data and generates index files. Therefore, we verified that eCLASS is more efficient than CLASS in an edge cloud. Finally, with regard to operation-cost evaluation of CLASS and eCLASS in the edge service models, cost evaluation was

related to CPU usage time and stored log data size. Storage cost was the same, but costs depend on VM computing capacity. In GM, when conducting log processing of 200 kb of log entries, the cost is $0.0000177 in low capacity, but $0.0000225 in high capacity. This confirmed that the same process was carried out at a 52% increase in high capacity.

Thus, in an edge cloud, the log-segmentation and distributed-storage methods of eCLASS are faster at log processing, but log processing time is similar as computing capacity increases. In addition, high capacity in terms of cost has high CPU cost, so high capacity is more expensive than low capacity for the same logging processing.

5.4. Security Analysis

Correctness: For log accuracy, SecLaaS and CLASS create a PPL after a user checks the logs. The eCLASS allows users to check their logs at any time via the MIC network and, in the event of an incorrect log, requests log correction from the CSP, investigator, or auditor. Therefore, users can check logs at any time in order to prevent incorrect logging.

Tamper resistance: In CLASS and SecLaaS, CSPs keep encrypted log DBs and PPLs. For this reason, users, CSPs, and investigators could manipulate encrypted log databases and PPLs if they conspired. However, since eCLASS shares relevant indices with MIC network participants by segmentation and the distributed storage of logs generated from edge nodes, modifying the logs requires the participation of CSPs, investigators, users, and a large number of MIC participants. The collusion of many participants is very difficult and becomes impossible as the number of MIC participants increases.

Verifiability: eCLASS consists of several processing steps, such as log collection, log segmentation, distributed storage, and recovery. Methods for verifying each of these steps were developed. Log- and index-integrity verification was performed, and the integrity of the MIC network was verified through MIC sequence verification. Users could also check the integrity of their logs at any time through MIC network participants.

Confidentiality and privacy: To protect log confidentiality and user privacy, eCLASS prevents information disclosure by dividing logs into sizes that make it difficult to extract information, and by non-continuously arranging logs. It also performs distributed and overlaid storage for different storage providers, preserving log confidentiality and safety. Finally, the index is encrypted using the user's public key to protect privacy.

Admissibility: eCLASS legally collects log data and performs the verification of log modulation for log collection. It also ensures the integrity of log data. Log files in eCLASS are thus acceptable in court and can be checked for data modulation through MIC network participants. Even if all participants (CSP, users, and investigators) of eCLASS are in collusion, data recovery is possible by using MIC network participants, even if it is intentionally deleted. The eCLASS was designed to meet all of the previously addressed security properties, and it is able to overcome some of the limitations inherent in CLASS. The limitations and how eCLASS can overcome each of them are described below.

- Logging scheme that takes into account the security environment of edge nodes

Edge nodes in edge-cloud environments are installed in no-trust zones with traditional cloud environments, enabling physical and logical intrusion. Therefore, because log modulation is possible at any time prior to the segmentation and distributed-storage phases, eCLASS was designed to hash copies with log generation to check the modulation of logs during the log-verification phase.

- Preventing log contamination due to the collusion between owner CSP, users, and investigators

Because both the encrypted log DB and PPL storage were kept by CSPs, there was no way to prevent or verify if users, investigators, and CSPs colluded to tamper with the log DB and PPL. In eCLASS, it is not possible to change all MI values for all MIC network participants because eCLASS shares encrypted index values with participants over the MIC network.

- Independent digital forensic system from CSP

As mentioned in the previous paragraph, CLASS allows CSPs to delete encrypted log DB and PPL storage and destroy relevant servers for malicious purposes so that they cannot submit data to the court. The CLASS protects the log data using methods such as encryption and content concealment, but is too dependent on the corresponding CSP. In eCLASS, all relevant logs can be collected without cooperation with the corresponding CSP because eCLASS splits and stores segmented log data, and preserves index values over the MIC network, as shown in Figure 19. In addition, Table 4 suumarizes the comparison between CLASS and eCLASS for the security analysis.

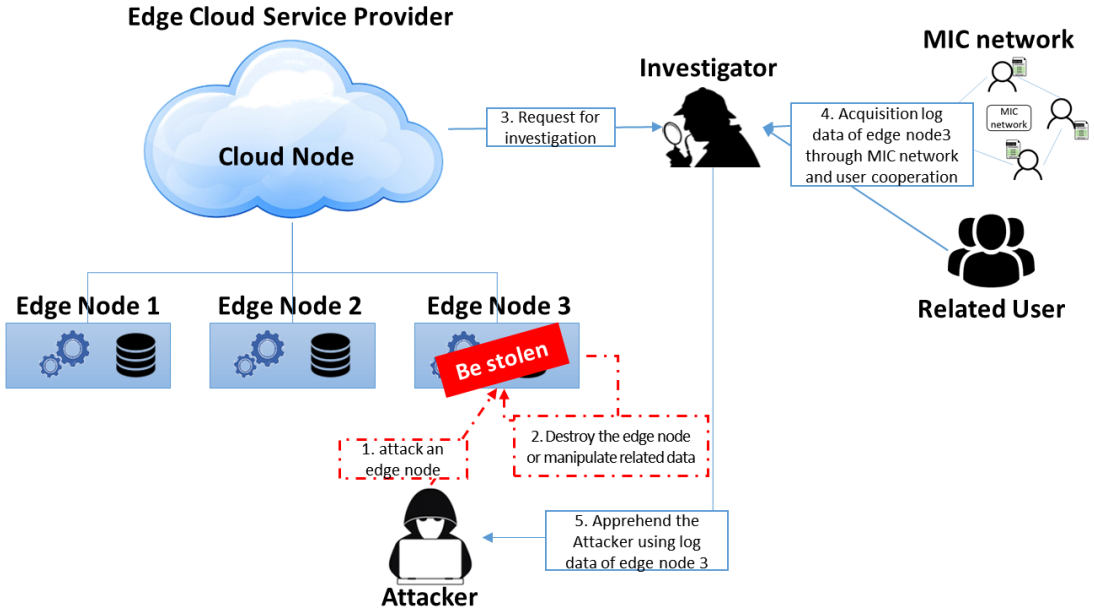


Figure 19. Procedure of log-data recovery for digital investigation.

Table 4. Comparison summary between CLASS and eCLASS.

Challenge/ Threat	CLASS	eCLASS
1. Modification of logs after publishing PPL/MI	Detected	Detected
2. Log-tampering resistance of collusion with CSP, investigator, and users	Undetected	Detected through MIC network participants
3. Log repudiation by CSP, investigator, and users	Detected	Detected
4. User-privacy violation by collusion between CSP and investigator	Privacy preserved	Privacy preserved
5. Log processing interruption by users	Possible; user should encrypt user log by user public key	Impossible; user can check user log after publishing MI
6. Edge-cloud-environment compatibility	Supports accumulator with single processing	Supports distributed edge nodes and multiprocessing of several indices
7. Users or investigators can recover logs without cooperation with CSP	Unrecoverable	Recoverable

5.5. eCLASS Quality of Service

Since eCLASS is a scheme that guarantees the integrity and confidentiality of logs generated at edge nodes that provide services, QoS for eCLASS is also based on edge-node data security. Other types

of QoS, fog computing, MEC computing, and cloud computing are measured in various ways, and QoS criteria are defined according to service characteristics [32,33]. The QoS of eCLASS, QoS_eCLASS_{level}, and the QoS of an edge node, QoS_Edge_{level} are defined as below, considering three aspects, log-data integrity (I_{level}), confidentiality (C_{level}) with the log-segmentation method, and resource usage (R_{level}) of edge nodes.

$$QoS\_eCLASS_{level} = \sum all\ of\ QoS\_Edge_{level} \tag{6}$$

$$QoS\_Edge_{level} = I_{level} + C_{level} + R_{level} \tag{7}$$

Integrity level (denoted to I_{level}) was increased linearly by the number of integrity checks (denoted to Check_Count); that means that, when the number of integrity checks is high, the integrity level is also high.

$$I_{level} = A_1 * \text{Check\_Count} + A_2 \tag{8}$$

where Check_Count is the number of integrity checks, and both A_1 and A_2 are determined by the technique complexity of the integrity checks.

Confidentiality level(denoted to C_{level}) is mainly determined by the size of a segmentation block (denoted to Block_Size). In eCLASS, the log-segmentation and partitioning method is used to protect log-data confidentiality for considering the limit of edge-node computing resources. Moreover, the smaller the size of a segmentation block is, the more it improves log confidentiality, because it becomes more difficult to recognize and recover data. The security level is directly proportional to the size of a segmentation block. Therefore, C_{level} is defined as follows with References [34,35]:

$$C_{level} = B_1 * Block\_Size + B_2 \tag{9}$$

where Block_Size is the size of the segmentation block for ensuring log confidentiality, e.g., 5 or 10 bytes. Both B_1 and B_2 are determined by the type of log data, e.g., character, number and size of overlapping segmentation.

Resource level (denoted to R_{level}) is the amount of available resources except for the total amount of resources used (denoted to R_{used}) for services, default security functions, and management functions at the amount of resource of edge node (denoted to R_{total}). As resource usage increases, the collection and confidentiality protection of log data at the edge nodes is delayed. For this reason, data security at the edge nodes is vulnerable because attackers can easily access data for data tampering and theft. Thus, resource level is also closely related to QoS of eCLASS. The resource level is defined as below. In eCLASS, only CPU usage was considered as a resource for QoS:

$$R_{level} = R_{total} - R_{used} \tag{10}$$

Therefore, the QoS of each edge node is measured according to the three above criteria, and the QoS of eCLASS can be checked by QoS sum or average of the related edge nodes.

6. Conclusions

In this paper, we proposed a secure logging scheme in edge clouds for digital forensics with features that facilitate the preservation of user privacy and confidentiality, ensure log data with a MIC network, and take into account the characteristics of edge-node security.

We also defined service models, threat models, and security properties of the edge cloud and help to understand the structure and logging procedure of the proposed eCLASS. Moreover, we proposed the log-data-segmentation and distributed-storage method for edge nodes that have limited computing resources. The problem of dependent CSP for digital forensics was solved by the eCLASS MIC network that can collect log data without the cooperation of a CSP, and a user can also check log data at any time through MIC network participants without CSP cooperation. Moreover, our

implementation on a virtual box demonstrated the feasibility and practicality of the proposed eCLASS. Through the performance and security evaluation of eCLASS, we verified that the log-segmentation and distributed-storage methods are efficient in low capacity, and we calculated the operating costs according to the edge-cloud service models.

This paper presented a new logging scheme in edge clouds for digital forensics. However, there were several limitations to this study. So, potential future extensions include the following:

- In the proposed eCLASS, we focused on computing overhead and operation cost in edge nodes. However, eCLASS consisted of three major entities: edge node (including eCSP), investigators, and DSCs. Thus, we need to design an authentication and access-control scheme for eCLASS that is composed of three entities, such as the one in Reference [36].
- Commonly, service logs are low-level data and hard for the common user to understand. In addition, many service providers use different log-data formats. Thus, we will explore standardization of the log format to cover most service-log data.
- Designing and implementing a prototype of the proposed scheme in collaboration with real-world eCSPs, storage-service providers, and forensic investigators with the aim of evaluating its utility in a real-world environment.

Author Contributions: Conceptualization, E.-N.H. and J.P.; methodology, J.P.; software, J.P.; validation, E.-N.H. and J.P.; formal analysis, E.-N.H. and J.P.; investigation, J.P.; resources, E.-N.H. and J.P.; data curation, E.-N.H. and J.P.; writing—original draft preparation, J.P.; writing—review and editing, E.-N.H.; visualization, J.P.; supervision, E.-N.H.

Funding: This research received no external funding

Acknowledgments: This research was supported by the MSIT (Ministry of Science and ICT), Korea, under the ICT Consilience Creative program (IITP-2019-2015-0-00742) and "Service mobility support distributed cloud technology" (No.2017-0-00294) supervised by the IITP(Institute for Information & communications Technology Planning & Evaluation).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Huh, E.-N.; HE, X. Draft new Recommendation ITU-T Y.3508 (formerly Y.ccdc-reqts): "Cloud computing—Overview and high-level requirements of distributed cloud"—for consent. *ITU-T SG13* **2019**, 1–27. Available online: https://www.itu.int/rec/T-REC-Y.3508/en (accessed on 29 August 2019).
2. Wang, Y.; Uehara, T.; Sasaki, R. Fog Computing: Issues and Challenges in Security and Forensics. In Proceedings of the IEEE 39th Annual International Computers, Software, and Applications Conference, Taichung, Taiwan, 1–5 July 2015; pp. 53–59.
3. Roman, R.; Lopezm, J.; Mambo, M. Mobile edge computing, Fog et al.: A survey and analysis of security threats and challenges. *Future Gener. Comput. Syst.* **2018**, *78*, 608–698. [CrossRef]
4. Mukherjee, M.; Matam, R.; Shu, L.; Maglaras, L.; Ferrage, M.A.; Choudhury, N.; Kumar, V. Security and Privacy in fog computing: Challenges. *IEEE Access* **2017**, *5*, 19293–19304. [CrossRef]
5. Yaqoob, I.; Hashem, I.A.T.; Ahmed, A.; Kazmi, S.M.A.; Hong, C.S. Internet of things forensics: Recent advances, taxonomy, requirements, and open challenges. *Future Gener. Comput. Syst.* **2019**, *92*, 265–275. [CrossRef]
6. Satyanarayanan, M. The emergence of edge computing. *IEEE Comput. Soc.* **2017**, *50*, 30–39. [CrossRef]
7. Zawoad, S.; Dutta, A.K.; Hasan, R. SecLaaS: Secure Logging-as-a-Service for cloud forensics. In *Proceedings of the 8th ACM SIGSAC Symposium on Information, Computer and Communications Security*; ACM: New York, NY, USA, 2013; pp. 219–230.
8. Ray, I.; Belyaev, K.; Strizhov, M.; Mulamba, D.; Rajaram, M. Secure Logging As a Service—Delegating Log Management to the Cloud. *IEEE Syst. J.* **2013**, *7*, 323–334. [CrossRef]
9. Zawoad, S.; Dutta, A.K.; Hasanm, R. Towards Building Forensics Enabled Cloud Through Secure Logging-as-a-Service. *IEEE Trans. Dependable Secur. Comput.* **2016**, *13*, 148–162. [CrossRef]

10. Sree, T.R.; Bhanu, S.M.S. Secure logging scheme for forensic analysis in cloud. *Concurr. Comput. Pract. Exp.* **2019**, *31*, e5143. [CrossRef]
11. Ahsan, M.A.M.; Wahab, A.W.A.; Idris, M.Y.I.; Khan, S.; Bachura, E.; Choo, K.K.R. CLASS: Cloud Log Assuring Soundness and Secrecy Scheme for Cloud Forensics. *IEEE Trans. Sustain. Comput.* **2018**, 1–15. [CrossRef]
12. Wang, S.; Zhang, X.; Zhang, Y.; Wang, L.; Yang, J.; Wang, W. A survey on mobile edge networks: Convergence of computing, caching and communications. *IEEE Access* **2017**, *5*, 6757–6779. [CrossRef]
13. Klas, G.I. Fog computing and mobile edge cloud gain momentum open fog consortium, etsi mec and cloudlets. Y.I Readings. 22 November 2015. Available online: https://yucianga.info/?p=938 (accessed on 21 September 2019).
14. Dolui, K.; Datta, S.K. Comparison of edge computing implementations: Fog computing, cloudlet and mobile edge computing. In Proceedings of the 2017 Global Internet of Things Summit (GIoTS), Geneva, Switzerland, 6–9 June 2017; pp. 1–6.
15. Shahzadi, S.; Iqbal, M.; Dagiuklas, T.; Qayyum, Z.U. Multi-access edge computing: Open issues, challenges and future perspectives. *J. Cloud Comput.* **2017**, *6*, 30. [CrossRef]
16. Borcoci, E.; Obreja, S. Edge Computing Architectures—A Survey on Convergence of Solutions. In Proceedings of the FUTURE COMPUTING 2018: The Tenth International Conference on Future Computational Technologies and Applications, IARIA, Barcelona, Spain, 18–22 February 2018.
17. Tang, B.; Chen, Z.; Hefferman, G.; Wei, T.; Haibo, H.; Yang, Q. A hierarchical distributed fog computing architecture for big data analysis in smart cities. In *Proceedings of the ASE BigData and Social Informatics 2015*; ACM: New York, NY, USA, 2015; pp. 1–6.
18. Patrascu, A.; Patriciu, V.-V. Logging System for Cloud Computing Forensic Environments. *J. Control. Eng. Appl. Inform.* **2014**, *16*, 80–88.
19. Lin, C.-Y.; Chang, M.-C.; Chiu, H.-C.; Shyu, K.-H. Secure logging framework integrating with cloud database. In Proceedings of the 2015 International Carnahan Conference on Security Technology (ICCST), Taipei, Taiwan, 21–24 September 2015; pp. 13–17.
20. Zawoad, S.; Mernik, M.; Hasan, R. FAL: A forensics aware language for secure logging. In Proceedings of the 2013 Federated Conference on Computer Science and Information Systems, Krako¿w, Poland, 8–11 September 2013; pp. 1579–1586.
21. Yang, L.; Cao, J.; Yuan, T.; Li, T.; Han, A.; Chan, A. A Framework for partitioning and execution of data stream application in mobile cloud computing. *ACM SIGMETRICS Perform. Eval. Rev.* **2013**, *40*, 23–32. [CrossRef]
22. Selvakumar, C.; Rathanam, G.J.; Sumalatha, M.R. PDDS-Improving cloud data storage security using data partitioning technique. In Proceedings of the 2013 3rd IEEE International Advance Computing Conference (IACC), Ghaziabad, India, 22–23 February 2013; pp. 7–11.
23. Karam, Y.; Baker, T.; Taleb-Bendiab, A. Security Support for Intention Driven Elastic Cloud Computing. In Proceedings of the 2012 Sixth UKSim-AMSS 6th European Modelling Symposium, Valetta, Malta, 14–16 November 2012; pp. 67–73.
24. Asim, M.; Yautsiukhin, A.; Brucker, A.D.; Baker, T.; Shi, Q.; Lempereur, B. Security policy monitoring of BPMN-based service compositions. *J. Softw. Evol. Process.* **2018**, *30*, e1944. [CrossRef]
25. Das, U.P.; R, V.K.; Ravishankar, B.R. Securing Data in Cloud using Disinformation Data Fog Computing. *Int. J. Comput. Sci. Trends Technol. (IJCST)* **2018**, *6*, 244–248.
26. Liu, X.; Yang, Y.; Choo, K.K.R.; Wang, H. Security and Privacy Challenges for Internet-of-Things and Fog Computing. *Wirel. Commun. Mob. Comput.* **2018**, *2018*, 1–3. [CrossRef]
27. Mohamed, E.G.; Kiram, E.; Ahmed, M.; Latifa, E.R. Blockchain and Multi-Agent System: A New Promising Approach for Cloud Data Integrity Auditing with Deduplication. *Int. J. Commun. Netw. Inf. Secur.* **2019**, *11*, 175–184.
28. Rantos, K.; Drosatos, G.; Demertzis, K.; Ilioudis, C.; Papanikolaou, A. Blockchain-based consents management for personal data processing in the IoT ecosystem. In Proceedings of the 15th International Joint Conference on e-Business and Telecommunications, Porto, Portugal, 26–28 July 2018; Volume 2, pp. 572–577.
29. Tariq, M.; Asim, M.; Al-Obeidat, F.; Farooqi, M.Z.; Baker, T.; Hammoudeh, M.; Ghafir, I. The Security of Big Data in Fog-Enabled IoT Applications Including Blockchain: A Survey. *Sensors* **2019**, *19*, 1788. [CrossRef] [PubMed]
30. Amazon EC2 Pricing. Available online: https://aws.amazon.com/ec2/pricing/on-demand/?nc1=h_ls (accessed on 30 August 2019).

31. Amazon S3 Pricing. Available online: https://aws.amazon.com/s3/pricing/?nc1=h_ls, (accessed on 30 August 2019).
32. Babu, R.; Jayashree, K.; Abirami, R. Fog Computing Qos Review and Open Challenges. *Int. J. Fog Comput.* **2018**, *1*, 109–118. [CrossRef]
33. Wu, Y.; Nordstrom, L.; Wang, Y.; Hauser, C. Adaptive Cyber-Security Scheme Incorporating QoS Requirements for WAMC Applications. In Proceedings of the Power Systems Computation Conference(PSCC), Dublin, Ireland, 11–15 June 2018; pp. 1–8.
34. Hwang, S.-J.; Kim, D.-Y. A Study on the Optimal Assignment of the Network Bandwidth Considering the Security and QoS. In Proceedings of the Symposium of the Korean Institute of communications and Information Sciences, Jeju, Korea, 22–24 June 2009; pp. 194–195.
35. Rachedi, A.; Benslimane, A. Multi-objective optimization for Security and QoS adaptation in Wireless Sensor Networks. In Proceedings of the IEEE ICC Ad-hoc and Sensor Networking Symposium, Kuala Lumpur, Malaysia, 22–27 May 2016; pp. 1–7.
36. Aghili, S.F.; Mala, H.; Shojafar, M.; Peris-Lopez, P. LACO: Lightweight Three-Factor Authentication, Access Control and Ownership Transfer Scheme for E-Health Systems in IoT. *Future Gener. Comput. Syst.* **2019**, *96*, 410–424. [CrossRef]