electronics

MDPI

Article

Proactive Forensics in IoT: Privacy-Aware Log-Preservation Architecture in Fog-Enabled-Cloud Using Holochain and Containerization Technologies

Kanwal Janjua [1], Munam Ali Shah [1], Ahmad Almogren [2,*], Hasan Ali Khattak [1,*], Carsten Maple [3] and Ikram Ud Din [4]

1 Department of Computer Science, COMSATS University Islamabad, Islamabad Campus, Islamabad 45000, Pakistan; kanwaljanjua00@gmail.com (K.J.); mshah@comsats.edu.pk (M.A.S.)
2 Department of Computer Science, College of Computer and Information Sciences, King Saud University, Riyadh 11633, Saudi Arabia
3 Warwick Manufacturing Group, University of Warwick, Coventry CV4 7AL, UK; cm@warwick.ac.uk
4 Department of Information Technology, The University of Haripur, Haripur 22620, Pakistan; ikramuddin205@yahoo.com
* Correspondence: ahalmogren@ksu.edu.sa (A.A.); hasan.alikhattak@gmail.com (H.A.K.)

Received: 23 May 2020; Accepted: 16 July 2020; Published: 19 July 2020

Abstract: Collecting and preserving the smart environment logs connected to cloud storage is challenging due to the black-box nature and the multi-tenant cloud models which can pervade log secrecy and privacy. The existing work for log secrecy and confidentiality depends on cloud-assisted models, but these models are prone to multi-stakeholder collusion problems. This study proposes 'PLAF,' a holistic and automated architecture for proactive forensics in the Internet of Things (IoT) that considers the security and privacy-aware distributed edge node log preservation by tackling the multi-stakeholder issue in a fog enabled cloud. We have developed a test-bed to implement the specification, as mentioned earlier, by incorporating many state-of-the-art technologies in one place. We used Holochain to preserve log integrity, provenance, log verifiability, trust admissibility, and ownership non-repudiation. We introduced the privacy preservation automation of log probing via non-malicious command and control botnets in the container environment. For continuous and robust integration of IoT microservices, we used docker containerization technology. For secure storage and session establishment for logs validation, Paillier Homomorphic Encryption, and SSL with Curve25519 is used respectively. We performed the security and performance analysis of the proposed PLAF architecture and showed that, in stress conditions, the automatic log harvesting running in containers gives a 95% confidence interval. Moreover, we show that log preservation via Holochain can be performed on ARM-Based architectures such as Raspberry Pi in a very less amount of time when compared with RSA and blockchain.

Keywords: cloud computing; fog computing; docker containers; Holochain; privacy preservation; log security; proactive forensics

1. Introduction

Cloud computing has provided numerous features such as on-demand services, resilience to security attacks, and ubiquity to many fields in enterprise networks [1–3]. Usually, Small and Medium Enterprises (SMEs) put their workload on the cloud, and it is estimated that 83% of total business will be on cloud reaching up to $411 billion of market value [4]. Security and privacy of data have always been the ever-growing and prime issue in cloud computing models and services. In 2018, the Massachusetts Institute of Technology (MIT) forecast that the ransomware would be the prevalent

cyber threat in cloud computing [5]. In the cloud environment, the logs are the first and significant piece of evidence for digital forensic investigations. To secure the log data in cloud storage, the existing cloud logging schemes use encryption; thus, the validity of logs becomes uncertain in the presence of an adversary and a malicious Cloud Service Provider (CSP) [6].

The ubiquity, smartness, and communication abilities of the Internet of Things (IoT) devices coupled with cloud, provide tremendous assistance in healthcare, industrial control systems, smart homes, and transportation [7–9]. As signposted by CISCO, the estimation of IoT revenue will be approximately $14.4 trillion between 2013 to 2022. On the other hand, security and privacy threats such as disruption of IoT networks, Denial of Service (DoS) attacks, and traffic analysis attacks are the main problems in the digital investigation in a cyber-crime scene [10,11]. An illustration of security trepidation in IoT-based smart environments as shown in Figure 1.

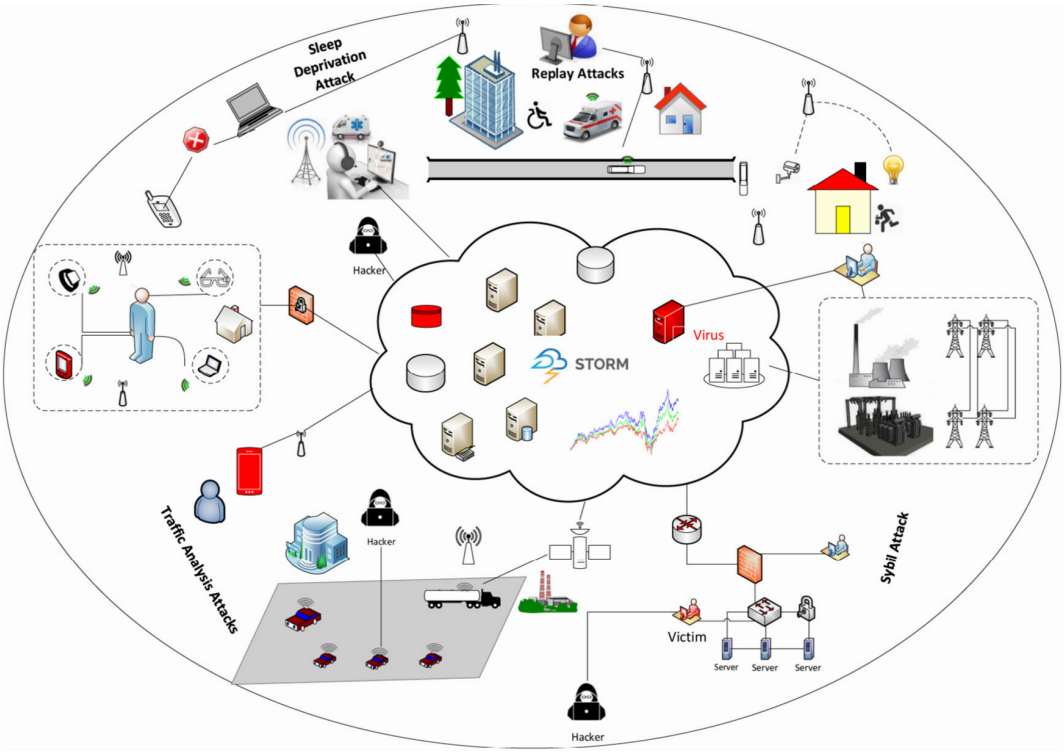

Figure 1. Threats in different IoT scenarios.

Cloud-connected IoT edge nodes are fragile on security management as they are geographically distributed from the cloud, which makes it easy for an adversary to launch an attack and remove the logs [12,13]. Moreover, edge nodes are not energy efficient and have potential problems for continuous log collection and preservation [14]. For instance, malicious users, attackers, edge-nodes, and the attacks of the rogue gateway and the rogue data center may be disguised as regular fogs between the data center and the users [15,16]. In May 2018, an IoT system with routers, surveillance cameras, and digital video recorders was disabled for four days after an attempt was made to hinder the service. There are many intrusion attempts at these service endpoints and the processing of forensic data are critical for ensuring the protection of endpoints and for resolving protection accidents.

Digital forensics has become more relevant in the fog cloud because fog nodes and edge nodes are targeted [17]. As a consequence, digital experts need a fog-cloud forensic data collection method, as attackers capture, exploit and remove the fog nodes, boundary nodes and the computers, rendering the processing of forensic data from fog nodes impossible [18]. Furthermore, the data of IoT devices are updating with time and, if a crime occurs using these devices, the attacker always intends to remove the traces of logs that are sent to virtual machines in clouds [19,20]. Therefore, a logging scheme is required to collect log data without CSP cooperation.

In case of a crime, sometimes it is not possible to separate and shut down the victim device and to carry it to the evidence extraction laboratory as it is done in traditional digital forensic scenes in which the investigator gets the equipment to obtain evidence [21]. Consider an example in which an attack is launched on an IoT device, the device is forcefully shut down and the volatile data of forensic significance are lost and there is no way to reconstruct the incident. Therefore, IoT forensics need to collect and retain useful knowledge for forensic purposes proactively. This will improve the forensic capability of the environment and will reduce the cost of incident responses [11]. Logging applications must define and solve these issues in fog-cloud environments.

Recently, the fog-cloud computing paradigm has been proposed, which offers different tools that focus on the advantages of fog computation [22]. In one of the top 10 strategic technical developments in 2018, Gartner has classified "fog cloud" as "fog driven". The implementation of the fog cloud model demands more efforts to upgrade the network, expand network availability, increase network efficiency, and get services closer to the consumer in a cost-effective manner [23]. Customer facilities are less safe and less efficient in the edge-cloud than the cloud structures. The reason is that a fog node is more user-appropriate than the edge-cloud.

Fog computing expands the computing and storing properties of cloud computing for the network edge which is a reasonable solution that provides low-latency via computational offloading. To avoid delay in edge cloud network communication, fog assisted IoT network is used to assign the tasks. This is done through computation offloading which lessen the load of the core network [24,25]. Moreover, it delivers short delay services, particularly for those computation-exhaustive and delay-sensitive jobs. Conversely, fog nodes also face constrained computing resources paralleled to the cloud. Therefore, fog node auxiliary offloads the task complexity to the cloud to attain supplementary computing resources. This mutually enhances the computing and communication resources in fog edge and cloud nodes [26,27].

Nonetheless, considering that the fog-enabled cloud edge security and privacy design and specifications have not been described explicitly, it is recommended that a fog enabled cloud framework should be used for security services before the deployment of the forensic logging mechanism [28]. Unlike a traditional cloud provider, fog-cloud systems are provided with fog nodes.

The following theoretical situation can demonstrate the particular problem that we anticipate to solve: An IoT edge node connects directly to the cloud and send its statistics for analytics. The attacker launches an attack to edge device for two purposes: (i) *compromise the edge device and use it as a bridge to get into the cloud and launch a bigger attack*; (ii) *delete the log information of edge device or make it dead so none of the digital assets can be retrieved for forensics reincarnation*.

In this research, we propose a Proactive Forensics in IoT, **P**rivacy-Aware **L**og-preservation **A**rchitecture in **F**og-enabled-cloud using Holochain and Containerization Technologies (PLAF). The proposed architecture introduced the holistic log preservation scheme which ensures the security and privacy of logs generated by IoT devices by considering the features of the fog-cloud. The salient features of the proposed PLAF are; *Log preservation* via ensuring log integrity, log verifiability, log provenance, and temper resistance, *Privacy preservation automation* through automated log collection and *Tackling multi-stakeholder problem* via assuring ownership non-repudiation and trust admissibility. All the aforementioned features are incorporated into the three-layered architecture of PLAF that are; *Layer 1*: Dedicated for Log generation and collection Layer; *Layer 2*: Performs log preservation task at fog level and *Layer 3*: Secure enclave at the cloud to securely store logs data and preserve proof of past logs. We have implemented the test-bed comprising of these three layers using different state-of-the-art technologies, which are: C&C (command and control) bots for autonomous log collection, Docker containers to orchestrate the IoT microservices, and Holochain for preservation.

Our Contributions

We propose an architecture for continuous log collection and preservation for IoT devices in a fog enabled cloud environment for proactive forensic aware logging. Moreover, the proposed

PLAF architecture performs the forensic aware logging and considers automated, secure, and privacy concerned distributed edge node log collection in fog-cloud by tackling the multi-stakeholder collusion problem. The contributions to the proposed scheme are as follows:

1. Privacy-Preserving Automation: For continuous and automated log collection, we used non-malicious botnets at fog level which provides the Privacy-Preserving Automation for IoT environment.
2. Proof of past log (PPL) preservation at fog level via Holochain: Preserving log integrity, privacy, provenance trust admissibility via Holochain distributed network instead of blockchain which is a more power-consuming approach.
3. Secure session management and log storage for log validation and verification are done via SSL mutual authentication with curve25519 and Paillier homomorphic encryption, respectively.

The organization of the paper is as follows: Section 2 discusses related work and background studies; Section 3 defines the threat model and threat scenarios, Section 4 demonstrates the proposed PLAF architecture, Section 7 concludes the paper and outlines plans for future work.

2. Related Work

The existing research proposes many solutions such as web-based management console and read-only API to secure logs for the reliable and efficient forensic process. However, the proposed schemes do not ensure the integrity and confidentiality when CSP is not trustworthy. Some other schemes provided the integrity privacy of preserved logs from the external attacker but failed to provide integrity and confidentiality when the logger itself is untrustworthy or a malicious user. The inevitability of honest and reliable log in the digital investigation is crucial. These issues have been investigated by the researchers in different dimensions.

Logging is a continuous process that keeps a record of each event occurred in the framework. This includes both the equipment and the programming part where numerous records are alluded to as log documents. The reason for log documents varies due to the nature of the product and the application that creates the occasions. Despite adaptation to non-critical failure, logs are a fundamental part of security control and advanced investigations as they can help the associations to identify incidents, security infringement, and noxious exercises.

A log securing scheme is given in [29] which depends on the security logs in the operating system. They used eucalyptus to set up their cloud environment and snort for IP tracing. They examine the performance of eucalyptus components after launching a DDoS attack on the cloud storage using the virtual machines that reside in cloud storage. As a result, they successfully identified the IP of attacking machines, the locks requested by the attacker, and the type of browser used. A management module called a cloud forensic module was implemented in the cloud infrastructure. The purpose of this module is to connect with the kernel, for example, a system call provides access to the network stack and virtual file system to obtain the logs of the machine. The limitation in his work is that the privacy and availability of the required logs haven't been validated [30].

The problem is the infrastructure layer needs modification so Ref. [31] provides a secure logging scheme for cloud infrastructure. The main feature of this work is to provide access logs such as network process logs of read-only API through CSP. Another scheme was proposed in [32] and implemented in FROST; however, this scheme failed to show how to protect the integrity and privacy of users log such as instigators and CSP. Another game based logging scheme was presented in [33] which states a use-case based game. This game-based approach is dependent on the model of the business and logging the security. This research proposed to generate the logs based on the following attributes such as timestamp, user ID, session ID, application ID, severity, and categorization of the morgue.

Variants of bloom filter technique are used in peer to peer networks and are presented in [34] for log integrity verification. The work presented in [35] automatically collects and preserves the logs. It also uses non-malicious botnets as-a-service in the cloud layer with no modification in cloud

infrastructure. A proactive cloud-based cross-reference framework is presented in [36] to verify the integrity of logs. A model is presented in [37] to minimize forensic evidence analysis time in the cloud environment through MapReduce. Another Blockchain-based secure logging framework is analyzed in [38] which provides cloud level integrity checking as storage. Forensic aware ecosystem for IoT termed as FAIoT is proposed in [39] which delivers physical device information in a virtual system for preservation, provenance, and integrity.

Zowoad et al. [40] proposed RESTful APIs to guarantee secrecy and minimize the potential for tampering using Proof of Past Log (PPL). For convenient and proficient proof examination, Kebande et al. proposed a Cloud Forensic Readiness Proof Analysis System that uses appropriate equipment in the cloud environment through MapReduce [41]. The framework contains modules including forensic database and Forensic MapReduce Task (FMT) modules. FMT recovers the potential advanced proof through the MapReduce process, including criminology logs, virtual pictures, and hypervisor mistake logs. All these logs are then placed in the forensic database.

Cloud log assuring soundness and secrecy(CLASS) is proposed in [42] which ensures maintenance of logs in the cloud through deviated encryption. McCabe et al. produced PPL using Rabin's fingerprint strategy and blossom channel [43]. Blockchain-based log preservation is also offered in their proposed scheme. It utilized the unchanging property of blockchain to guarantee the classification and integrity of cloud logs and proposed secure logging-as-an administration in the cloud environment. The plan includes steps such as extraction of logs from a virtual environment; formation of scrambled log passages for each log using open key encryption; and capacity of encoded log sections on the Blockchain.

An open verification model through Blockchain to guarantee the integrity of logs in the cloud is proposed in [44] using an outsider reviewer. The authors used homomorphic and single direction hash capacities to create the labels for log passages and Merkel tree structure for capacity. Another scheme named Probe-IoT ensures the confidentiality, anonymity, and non-repudiation of publicly available pieces of evidence by using a digital ledger [45].

Harvesting and preserving forensically useful data from smart environments is presented as IoT-Dots in [46]. A holistic overview of normal and malicious behavior in log collection [47]. A fog based IoT reediness forensic framework is presented [48]. eCLASS proposed a distributed network log storage and a verification mechanism of edge nodes [24]. Table 1 provides the comparison of related techniques on the bases of adversarial model assumptions, security features, and limitations.

After the detailed analysis of the existing literature on secure logging in a cloud-based environment for IoT and other network environments, we yielded that most of the solutions use Blockchain, RSA, and open key encryption to secure logs in the cloud environment. It is important to note that there is a problem and that is when we need to collect the logs of a smart environment which is multi-hop from the cloud and when the log integrity is threatened, the network latency is tremendously increased during the log sending. Therefore, there is a need for an architecture that preserves the log privacy, provenance, and confidentiality along with scalability of edge node integration in an efficient fashion. The subsequent sections provide the detail of proposed scheme based on discovered solutions along with threat model and security requirements [49,50].

Table 1. Comparison of existing work schemes.

Ref.	Tools and Technique	Adversarial Model Assumption	Security Features	Limitations
[29]	Used eucalyptus for cloud environment	Used snort for malicious activity tracing. Malicious user can eavesdrop messages	Successfully traced the malicious activity and necessary logs	Only provides the acquisition of logs, didn't address integrity.
[30]	Implemented a forensic module in infrastructure layer.	CSP or user is untrustworthy	Implemented a module that gather several logs (network, IoT, kernel, system calls)	Integrity and privacy of logs cannot be validated
[31]	Read only API by CSPs	API can be exploited	Logs (network, process) are available	Integrity and confidentiality is not addressed
[32]	Used FROST	IaaS dependency	Access logs (API, virtual machine and firewall)	Confidentiality and integrity are not addressed
[33]	An architecture based on use-cases to access logs	Use cases malfunctioning or loopholes	Provides different logs of different operations layers	Integrity of data and confidentiality of logs is not maintained
[34]	Hierarchical bloom filer, hash-based payload	Hierarchy may get exploited	Payload information in hierarchy	Complex due to hierarchical structure
[35]	Used botnets as a services	No modification is required in cloud infrastructure	Implanted non malicious botnets to collect and preserve logs	Deficiencies in implementation and prototypical assessment
[36]	Use FTK, EnCase	No security considerations	Cross reference of forensic logs	Neither the prototype of the model is given nor the confidentiality is addressed
[37]	MapReduce	Post -Incident	Reduces forensic proof examination period in the cloud	Private application and exemplary estimation
[38]	Secure logging	PPL can be demolished	Integration with cloud database	Temper resistant threat
[39]	FAIoT	Attacker can manipulate the evidence	Provenance Preservation	The device is physically needed
[40]	Secure logging as-a-service	Colluding among addressing entities	Provides RESTful APIs for integrity verification	Authentication and provenance is missing
[41]	Cloud Forensic Readiness Proof Analysis	PPL is tempered	Recuperated the possible radical evidence	Provenance is not addressed

Table 1. *Cont.*

Ref.	Tools and Technique	Adversarial Model Assumption	Security Features	Limitations
[42]	Cloud Log Assuring Soundness and Secrecy (CLASS)	Deviated encryption of logs	Secure PPL via Rabin's fingerprint strategy and blossom channel	Performance overhead while increasing number of logs Provenance and Authentication
[43]	Unchanging property of Blockchain	Proposed secure logging-as-an administration for the cloud	Integrity Log verification	Formation of scrambled log passages for each log and using open key encryption
[44]	Open verification model via Blockchain	Prevent logs from outsider	Markel tree and Homomorphic single track hash for integrity	Resource consumption
[45]	Probe-IoT	Ensures provenance, integrity , confidentiality	Provides a digital ledger to track all of the records in an IoT system	Performance overhead
[46]	IoT-Dots	Collects forensic data in early stages and then uses in future analysis	Harvest forensically useful log from smart apps of IoT environment	Increased network latency and Provenance
[47]	Prov-Things	Provides holistic overview of malicious and normal behavior	Preserve forensically needed data on devices	only collect logs of provenance and confidentiality
[48]	Fog based IoT forensic framework	Consider cyber-attacks on IoT systems	Provides early bird alarm of attack	System flow must be auxiliary assessed
[24]	eCLASS	Threat to log integrity in centralized in cloud storage	Edge cloud log securing via distributed network	Authentication Edge directs the cloud

3. Threat Model and Security Requirement Modelling

For a forensic data collection, an edge cloud environment must be taken into account for possible security risks, such as log leakage or forensic data removal. Unlike traditional cloud infrastructure, the edge-cloud services are supported using fog nodes. Security monitoring is inadequate with edge nodes delivering services because they are locally isolated from the cloud [51]. This geographical division of management is the main target of malicious users or attackers. Moreover, the log data of edge nodes cannot be securely stored and maintained at cloud storage. Furthermore, several manipulated log data can be entered, and log removal may be performed to prevent forensic investigators for the reincarnation of crime scenarios using the logs.

3.1. Modelling Attack Possibilities

We used four metrics to model the attack possibilities based on multi-stakeholder (CSP, investigator, and attacker) colluding issue and possible integrity theft from multi-stakeholder. These four metrics are: threat, the threat to the asset, possible safeguards, and security controls. Table 2 maps the actions of threats scenarios due to possible threats to assets and thus provides the essential security control requirements. The illustration of the threat model discussed in Table 2 is given in Figure 2 that provides an overview of the relationship amongst the threat agents and the cloud logging system. The terms used in threat modeling and attack possibilities are explained below.

- Asset: Log generated form edge nodes and sent to cloud and fog nodes for log preservation. These logs are the main asset to be protected from manipulation, tampering, and removal.
- Threat Agent: Multi-stake holders are all the corresponding entities that are assumed as threat agents which are, CSP, investigator, an attacker, or malicious user neighboring with the cloud storage in a cloud environment. These threat agents are characterized based on zero trust policy.
- Threat to Asset: Possible collusion among threat agents to asset, integrity theft to logging from multi-stakeholders.
- Possible Safeguards: Mechanisms and designed practices to mitigate threat possibilities.
- Security Controls: Those implemented and applied security requirements which are being applied to prohibit and minimized the threats and attacks.

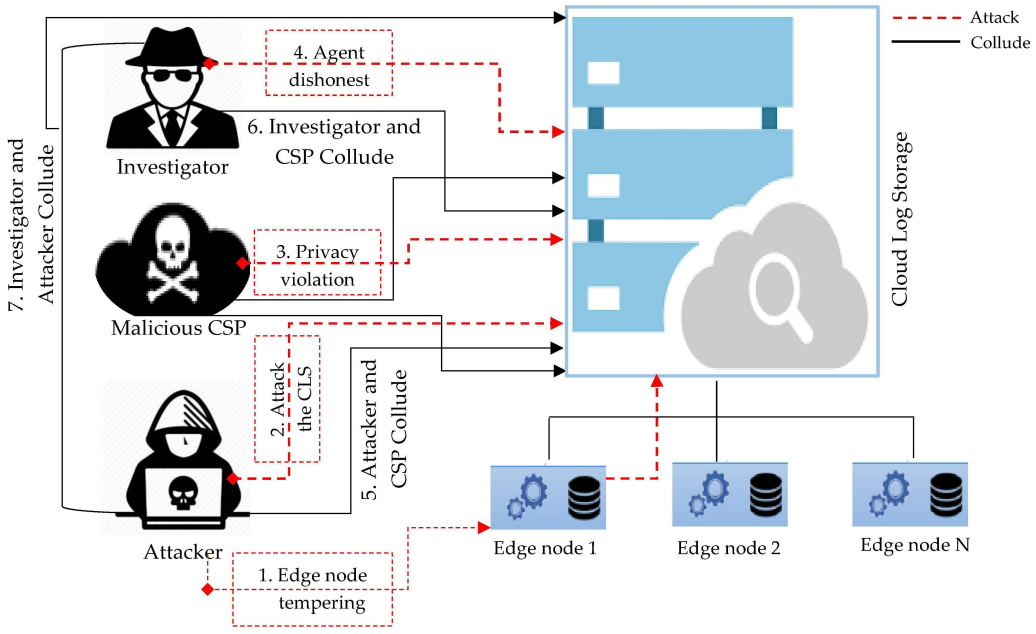

Figure 2. Threat model.

Table 2. Modelling attack possibilities.

Threat	Threats to the Assets	Possible Safeguards	Security Controls
T01: Edge node tampering. Computation overhead in edge node	*TA01:* Attacker abolish the device · The data of edge node device is altered	*PS01:* Secure and continuous log collection · Backup the Device orchestration image	*SR01:* Automation of log collection · Orchestration of edge device at fog level
T02: Attacker launches an attack to the cloud storage	*TA02:* Remove or manipulate the sensitive data. Launch the side channel attack to compromise CLS	*PS02:* Intervene and secure the path between edge node and cloud node	*SR02:* Secure the fog level log data integrity· Ensure the integrity of the data at cloud storage node.
T03: Privacy violation at CSP and confidentiality violation	*TA03:* Denial from CSP of log manipulation, tempering or removal	*PS03:* service confidentiality must present at fog level to store proof of past logs	*SR03:* Non repudiation and service confidentiality
T04: agent becomes dishonest or compromised	*TA04*: authentication threats while accessing the logs	*PS04:* mutual entity authentication scheme must be used	*SR04:* Public key cryptography for mutual authentication
T05: The CSP and the attacker colludes (CSP may be compromised or malicious). Ownership repudiation	*TA05:* attacker is able to get the PPL or learn the information from the logs	*PS03:* CSP would not be capable to disprove the available PPL. Attacker is not able to recover log material from distributed log	*SC05:* Log verifiability, trust admissibility and provenance
T06: CSP and investigator colludes. Ownership repudiation	*TA06:* CSP is malicious or untrustworthy then deny or hide provenance of the proof of logs after publishing them.	*PS06:* It's not credible for the detective to erase any log from the tenacious storing before giving it to the court of law	*SC06:* Non-Repudiation, temper resistant PPL, and provenance
T07: Attacker and investigator colludes	*TA07:* Tampering the PPL to modify the evidences to protect the criminal or to frame the honest user.	*PS07:* There must be secure enclave between path of CLS and edge node none of log entries can be changed at device level	*SC07:* Provenance of PPL. Light weight Hash block to secure the PPL· Reduce performance overhead at edge level to secure PPL

3.2. Modeling Security Requirements

We have identified the essential security requirements from the results of the previous section which are: ownership non-repudiation, trust admissibility, provenance of PPL, log verifiability, log integrity, temper resistance, and privacy-preserving automation. We modeled these security requirements (SR) using secure tropos methodology (STM) evaluated using the SecTro tool as well. The STM is a security-oriented extension of Tropos methodology and is reinforced by SecTro tool [52]. In STM, there are standards that are used to model the security requirement. These standards are illustrated in Figure 3. In what follows, the identified security requirements are explained and their modeling is illustrated via SecTro Tool.

1. Ownership Non-Repudiation: Since cloud servers contain information for other users, the malicious cloud users may access the data logs of other users. The CSP can deny reporting the logs' publication. The CSP creates and maintains logs for all users on certain logging systems, but users can not recognize logs. As a consequence, the implication of log integrity and ownership becomes vague and the CSP can also dispute the data. Figure 3 illustrates the security modeling of this requirement.
2. Trust Admissibility: The forensic data of logs generated in edge clouds should be used in court. For legal admissibility, logging, manipulation, or omission during log-data generation, management, and verification processes should be reviewed. Moreover, log collection on the basis of legal security policies for edge clouds should be performed because unauthorized log data collection infringes admissibility. Trust admissibility is modeled with ownership non-repudiation in Figure 4.
3. Provenance of PPL: Logs generated on cloud servers can only be read by users and the service provider on certain cloud logging systems, but other CSPs, users, and investigators can access logs that include service information, user status, user access frequency, and location. As the service log is indeed a confidential asset, it is imposing on the secrecy of the service to reach other parties (other customers, CSPs, investigators). Threats to this requirement are modelled in Figure 5.
4. Log Verifiability: Subsequently, the architecture PLAF is designed to produce fog-activated logs from the distributed edge-nodes network, log accuracy, consistency, and integrity require verification measures. Thus, a fog cloud logging architecture should be able to review the created logs and to verify data modulation and corruption.
5. Log Integrity: Provides log chains in a whole unchanged manner during the log collection state. The automated forensic analysis architecture should be correctly logged, and log details should show that no fraud has taken place. Distributed logs will indicate that they are connected and record the behavior that the requested service is using. Eventually, it is important to check that the reports are properly reported by users, researchers, and auditors.
6. Temper Resistance: This feature provides resistance to deliberate modification and manipulation of log data via unauthorized channels, both between storage and during transit.
7. Privacy-Preserving Automation: To avoid the computation overhead in edge node devices, where real-time IoT devices are producing data continuously, we used microservices orchestration for managing the integration of IoT devices at fog node. It provides continuous integration and less downtime during service migration. We used C&C automated bots in containers for autonomous log collection and preservation. Figure 6 gives the security feature modeling of privacy-preservation automation.

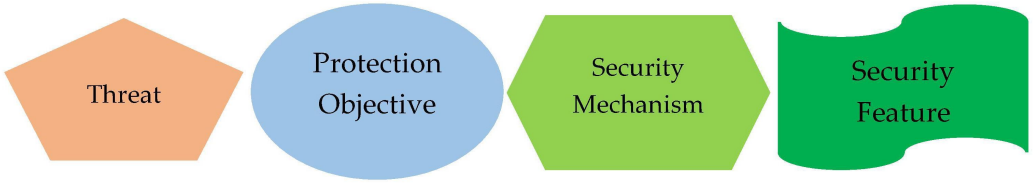

Figure 3. Secure Topos methodology.

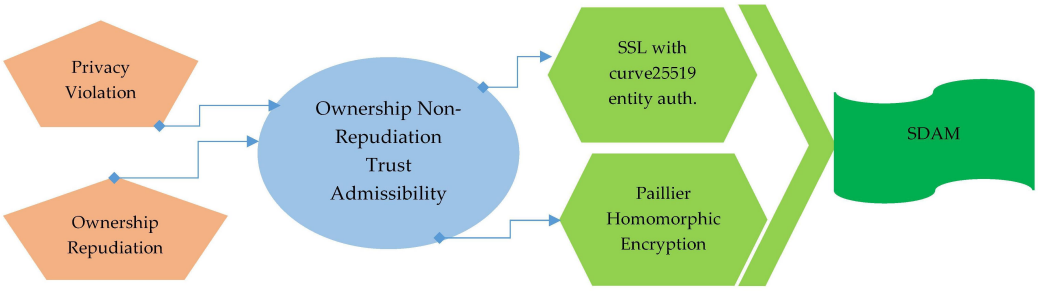

Figure 4. Ownership non-repudiation and trust admissibility modelling.

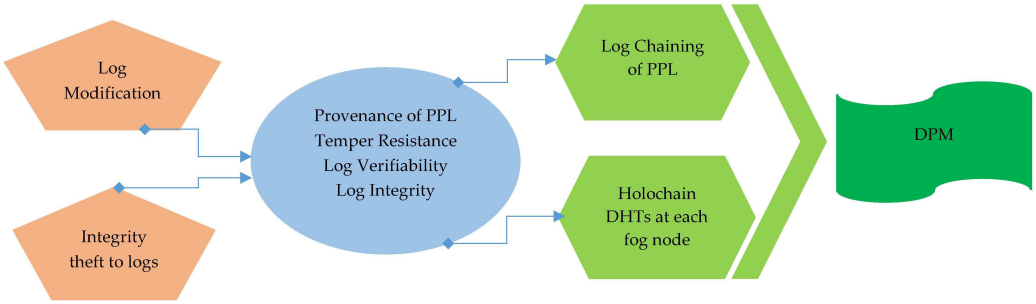

Figure 5. Log integrity and modification threats modelling.

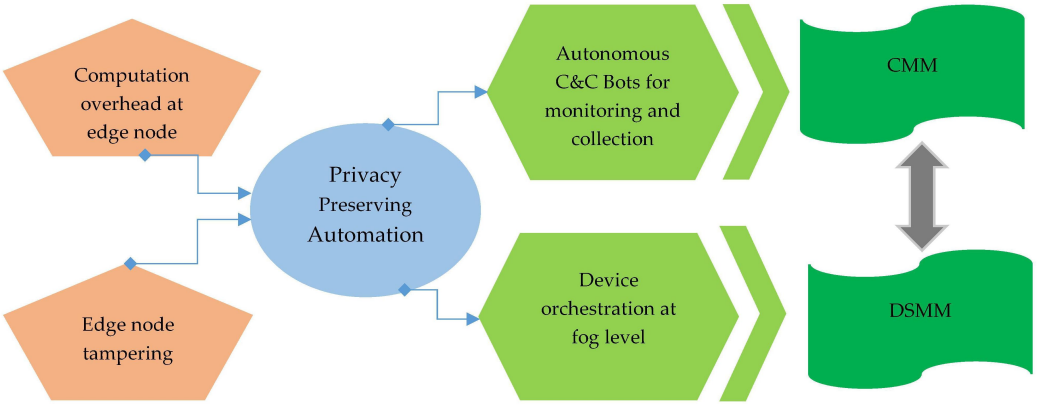

Figure 6. Privacy preserving automation modelling.

4. Proposed Architecture PLAF

We proposed a holistic architecture that addresses the main issues in our findings and implements the related security requirements in the proposed architecture PLAF. The main idea of PLAF is: "Automated and secure log collection and preservation of smart environment logs from distributed edge nodes in fog enabled cloud environment". To securely preserve and transfer logs from IoT devices

to cloud while considering the task offloading and delay sensitivity, we assisted the architecture of PLAF by intervening fog layer amid the IoT and cloud layer.

Fog computing lessens the cloud connotation by dint of pre-processing information generated by the sensors and further IoT devices. The logging and pre-processing in PLAF is performed at fog node and governed by fog node controller to offload the tasks of the edge layer. Conversely, fog node only collects and preserves the PPLs in dedicated format and transfer to the cloud log storage for further secure log archiving and log verification operations. All of the three layers offers different security controls for log secrecy and privacy to tackle multi-stakeholder issue and log alteration. In the following section, all three of the layers are described in detail. All of the operations and information flow of PLAF architecture are given in Figure 7 and described in the steps below:

1. Create an IoT environment.
2. Initiate Docker Swarm Management Module (DSMM) and Central Management Module (CMM) for the placement and orchestration of IoT microservice using docker swarm manager at fog worker node.
3. Placement of bots in a container beside the microservice to collect logs.
4. Initiate the Data Preservation Module (DPM) for log preservation at fog worker nodes via Holochain and send all the Proof of Past Logs (PPL) to the central manager node.
5. Fog controller node administrates the DSMM, CMM, and DPM. In addition, prepare the PPL to transfer from the central manager node to cloud log storage.
6. Archiving the logs from fog node controller to cloud log storage.
7. Secure the logs at Cloud Log Storage (CLS) via Piallier Homomorphic Encryption (PHE).
8. Investigator initiates the secure session to privately query the logs for forensic investigation via SSL with curve25519 mutual authentication.
9. Then, for validating the logs at CLS, secure sessions are established amid fog node and investigator to get PPLs and verify log integrity.

PLAF is comprised of three layers which are: Log generation and collection Layer, Log Preservation Layer, and Log Archiving layer. The security threats that are described in the threat model are addressed in these three layers of PLAF by implementing the security requirements, which are ownership non-repudiation, trust admissibility, temper resistance, log integrity, log verifiability, provenance of PPLs, and privacy-preserving Automation. The mapping of threats and corresponding security features align with essential security requirements in different layers of PLAF are elaborated and discussed in Table 3. The details of all activities performed with protocol-specific information in the three layers are illustrated in activity diagram of PLAF, see Figure 8. The overview of three layers of PLAF is described below:

Log Generation and Collection Layer: This layer offers the log harvesting and collection, The orchestration of IoT microservices is continuous Integration of new arrivals performed. In addition, it provides the autonomous log collection generated by IoT devices via C&c bots which are placed in the container of microservice. This layer includes two modules DSMM and CMM for privacy-preserving automation.

Log Preservation Layer: Secure Log preservation and generation of Proof-of-past logs is achieved in this layer. We used Holochain to perform the log Preservation in fog nodes. DPM is the module of this layer to generate PPL.

Log Archiving Layer: Secure log Storage and secure session establishments for querying the log from storage are performed here. Logs are stored here via Paillier Homomorphic Encryption and SSL with curve25519 is used for secure session establishment for querying. Two aforementioned operations are performed via SDM.

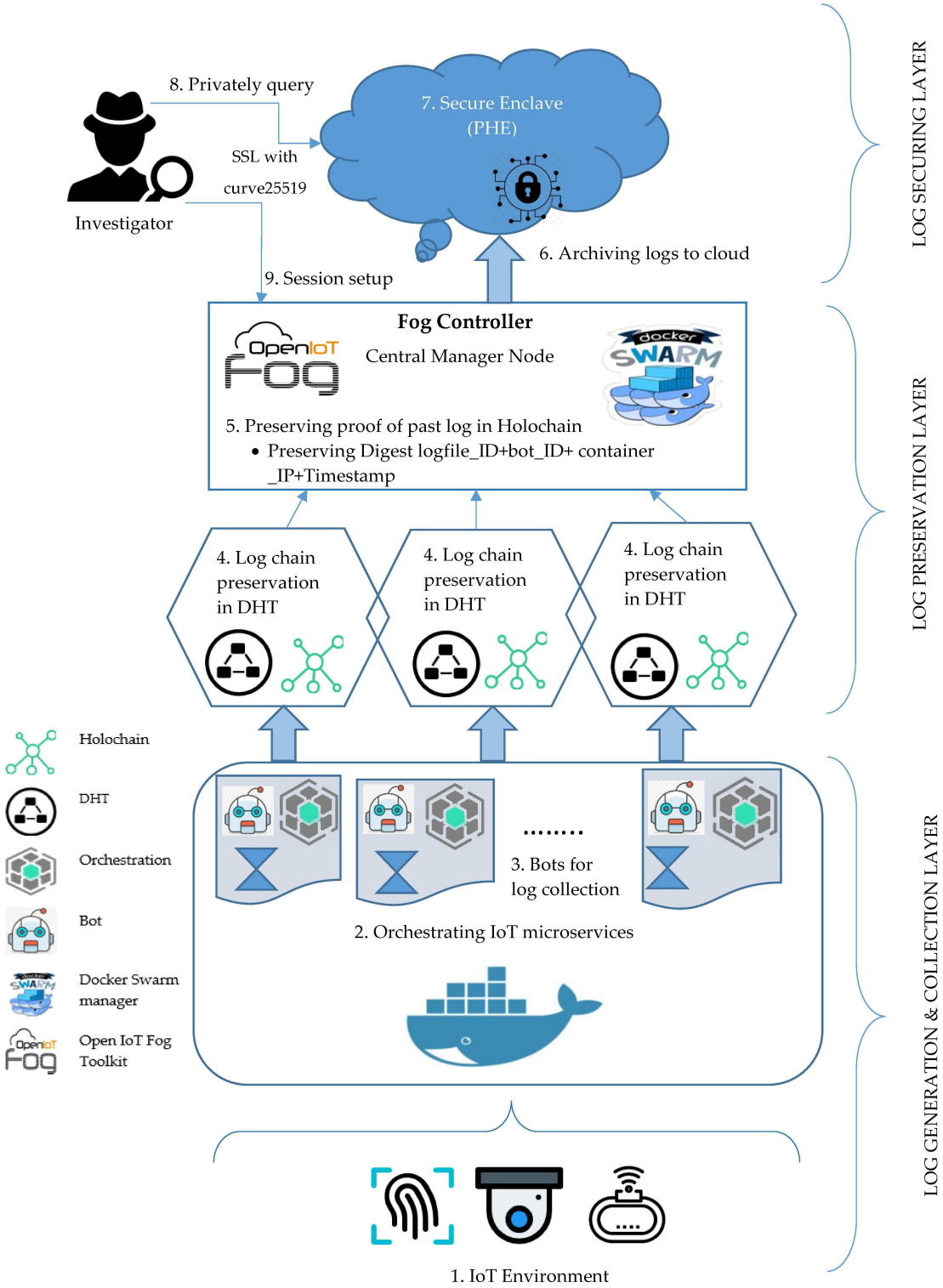

Figure 7. PLAF workflow.

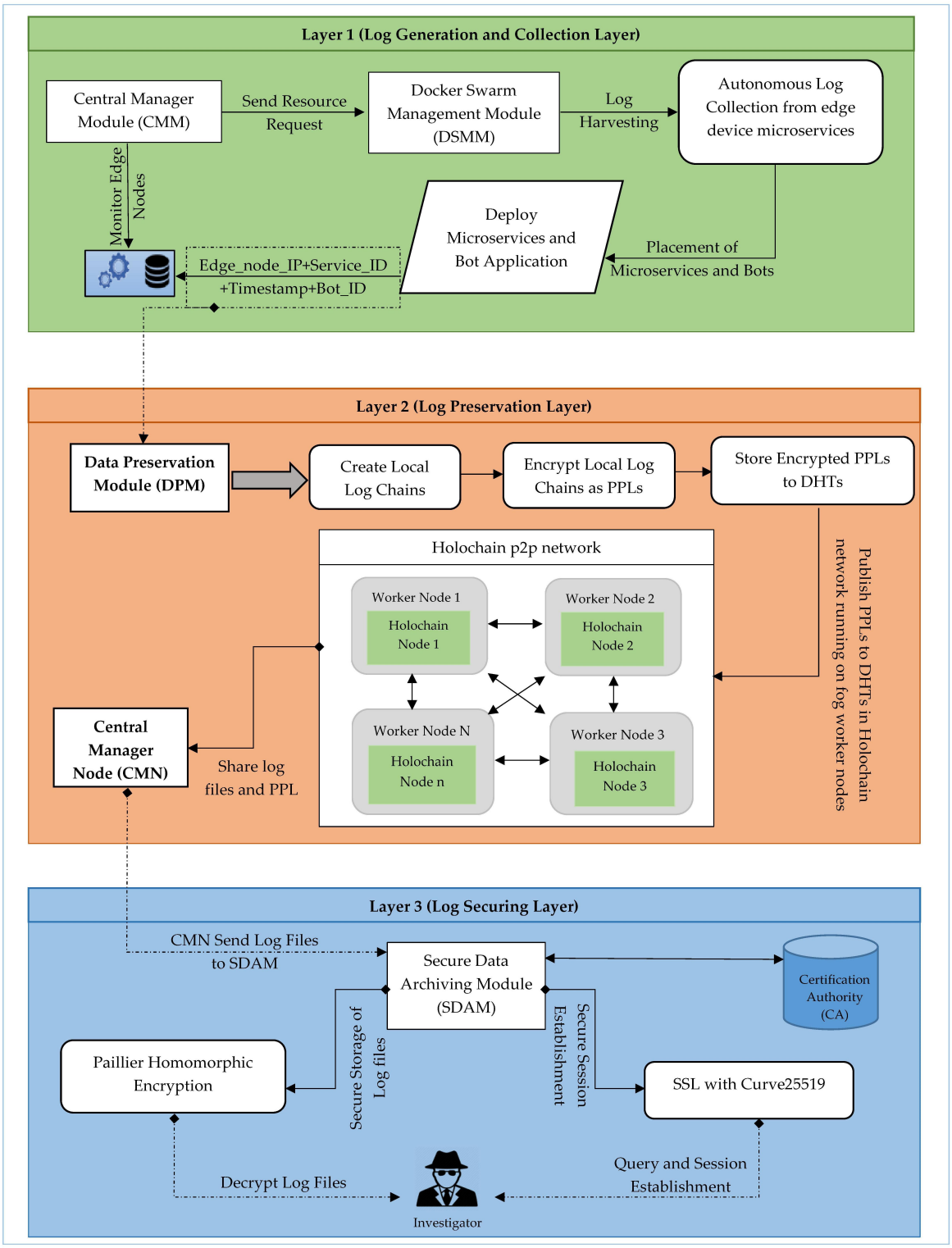

Figure 8. Details of procedures in PLAF layers.

Table 3. PLAF security layers.

Security Layer	Threat	Security Requirement/Protection Objective	Security Mechanism	Security Feature Module
Layer 1 Log Generation and Collection Layer La	Edge node tampering Computation overhead	Privacy-Preserving Automation by scalable orchestration of edge devices and autonomous log collection	Docker container orchestration for continuous integration of microservices. Automated C&C bots to collect logs from microservices	Docker Swarm Management module (DSMM) Central Manger Module (CMM)
Layer 2 Log Preservation Layer	Manipulation in Logs Storage	Log Integrity Temper Resistance Provenance of PPLs Preserving log integrity	Creates Log Chains of PPL and shared in Distributed Hash Tables (DHTs) of the Holochain Network	Data preservation (DPM)
Layer 3 Log Archiving Layer	Investigator may Impersonated or Compromised. Attacker steals logs from CLS	Log Verifiability, Trust Admissibility for Entity authentication, and Ownership non-Repudiation. Secure storage of logs	SSL with Curve25519 and Paillier Homomorphic Encryption	SDAM (Secure Data Archiving Module)

4.1. Log Generation and Collection Layer (Layer One)

This is the first driving part of the architecture where the logs are harvested and sent to the next layer of the PLAF. This layer performs the log generation and collection mechanism and these two operations are administered by DSMM and CMM. The DSMM administrates the other container worker nodes along with CMM. Both the CMM and DSM run on a fog node controller so having a central fog node controller provides the functionality to get the fine-grained control of running services in terms of both orchestration and monitoring. Other fog nodes called worker fog nodes run the app-containers and device microservices along with automated log collection bots.

To understand the DSMM and CMM operation, we illustrate an example. An Xbee-Stick is a connected IoT device node, after its connection, CMM is notified and the ZigBee application is started as swarm worker node and map the device in swarm cluster. CMM sends the notification to Docker swarm orchestration service to create the containerized microservices of a respective node based on the requirement of IoT devices such as Xbee-Stick. The app-container of IoT device node downloads the Docker image of respective microservices and starts. Following these two, DSMM and CMM of Layer one are discussed. Figure 9 provides the placement of CMM and DSMM in the controller node and their communication with worker nodes. We can see in Figure 9 that the central manager node communicates with the fog worker nodes in a back and forth manner. This is an illustration of layer two mechanisms where manager nodes send commands to worker node and receive their responses from them as well.

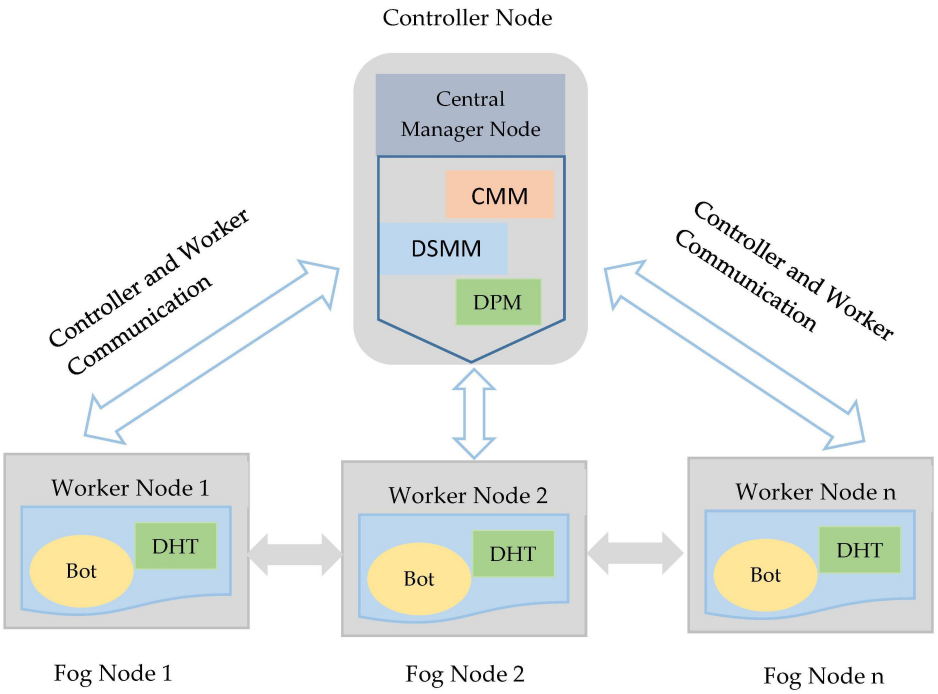

Figure 9. Placement of CMM and DSMM in a controller node.

4.1.1. Central Management Module (CMM)

We developed the OpenIoTFog Central Manager Module (CMM) which runs beside Docker Swarm Manager (DSM). The edge devices orchestrated in containers at fog worker nods may require some software packages for computing and data storage capabilities. Therefore, we deployed CMM to monitor the resource requirement of edge device microservice running in a container and inform the DSMM about updates in terms of software packages and resource allocation. The orchestration or virtualization of microservices is established in the context of the OpenIoTFog toolkit [53]. CMM manages and executes the service functionalities of each via following four operations which are: monitoring the health of fog node by and its resources (CPU, Storage, memory); providing the glimpse of "things" connected to manager; detecting the changes and autonomously execute commands on connected devices; and managing the access control of the device to the container for provisioning the swarm services.

4.1.2. Docker Swarm Service Manager Module (DSMM)

DSMM takes the information from edge devices from CMM in terms of resource requirements for scalable and continuous device integration. We used the Docker container orchestration tool for IoT microservice orchestration. Docker container technology shrinks the application overhead in deployment at fog node and utilizes the device resources and delivers application scalability. The Docker container orchestration environment is the best choice for IoT ecosystem orchestration, where the first layer of PLAF deploys, creates, starts, and stops the microservices running in a container or whole container. We built IoT microservices as a series of separately built, modified, and escalated microservices with autonomous log collecting bots. That microservice is used as an individual container and CPU, bandwidth, RAM, and storage specifications are specified as a resource necessity for each microservice.

The other application of DSMM provides autonomous log collection via automated C&C bots other than the microservice orchestration. The principal purpose of placing automated bots besides the microservice is that the first layer of PLAF is capable of autonomously capturing the logs in a container environment. To collect the logs autonomously in such a way that the intervention of an

outsider is avoided, the automated bots are deployed in the respective IoT device container running at fog worker node. These bots are the inspiration of malicious bots that works on the phenomenon of C&C (command and control) to collect desired information automatically. These automated bots are the automated command and control Python scripts, which collects the microservices application logs of the corresponding container in which they are running.

DSMM deploys the C&C bots besides the microservice in the container of edge device running on the fog worker node. To collect the log from IoT microservice, bots gather the stored logs in a specific directory provided on edge device microservice application. Logs are kept in storage in a predefined format along with the service information for forensics. Equation (1) provides the digest of information format harvested in the microservice of the edge device. These bots are assigned with nonce identifiers so that the corresponding logs must be identified separately. This will also be useful in making information digest during the log preservation chain. Figure 10 provides the inside view of bots working in containers along with microservices.

$$InformationDigest = Edge\_node\_IP + Bot\_ID + Timestamp + Microservice\_IP \quad (1)$$

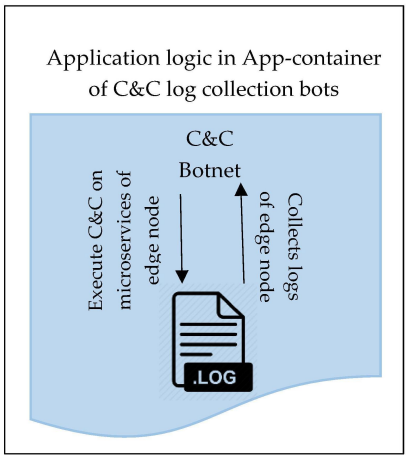

Figure 10. C&C bots running inside container.

4.2. Log Preservation Layer (Layer Two)

The importance of preserving digital information is to make sure that none of the malicious entity can perform modification and tempering to digital assets generated by layer one. In this layer, the aforementioned objective is achieved by preserving the integrity of logs and generating secure PPLs that builds resistance to tempering and provides log verifiability as well. This layer provides crucial safeguarding mechanisms to ensure log integrity, temper resistance, log verifiability, and provenance.

To ensure all the aforesaid security requirements, the proposed architecture PLAF provides a holistic solution that is secure and scalable at once which can easily be executed on a fog node even comprises of a Raspberry-Pi. In layer two, the Data Preservation Module (DPM) is deployed on a fog node controller to perform log preservation via Holochain for secure, scalable, and robust logs preservation. DPM runs beside the CMM and DSM at fog node controller as shown in Figure 9. In layer two, DPM sends the request to fog worker nodes and receives the responses from them. These worker nodes run the instances of Holochain nodes. These Holochain nodes manage the DHTs of log metadata. Following the details about fog worker and controller nodes executing the DPM, module procedures are given.

4.2.1. Data Preservation Module (DPM)

This module governs log data preservation to ensure log integrity, privacy, and peer-to-peer data sharing via Holochain. Holochain used for secure and scalable log preservation schemes which

establish the peer-to-peer network communication of all Holochain nodes distributed among fog worker nodes. The following sections describe the Holochain technology and different operations executed by Holochain at fog worker node [54,55].

4.2.2. Holochain

Holochain offers an agent-centric and relativistic environment to create the underlying validity of data. Holochain guarantees data integrity for distributed applications by carefully collecting data from the local immutable chain of each fog node. This effectively allows for an update to conventional double accounting with the use of cryptographic signatures as accounts are linked to unchanging chains. Each node manages its local transaction chain in this double-entry accounting system instead of a global coins leader as in Blockchain.

Holochain does not waste computing power as it is wasted in Blockchain because it is not dependent on some kind of global leader consensus. Moreover, Holochain is not dependent on the references to the proof-of-work, the proof-of-stake, or leading selection algorithms to ensure the data integrity for peer-to-peer applications.

This means that each balance of a fog worker node is stored on its chain, and when two fog worker nodes are transacted, they only have to test the background of their counter-party to make sure that they have credits. Third-party authorization or consensus is not needed in this case. Figure 11 illustrates the peer-to-peer validation mechanism of Holochain for privacy preservation in distributed fog worker nodes.

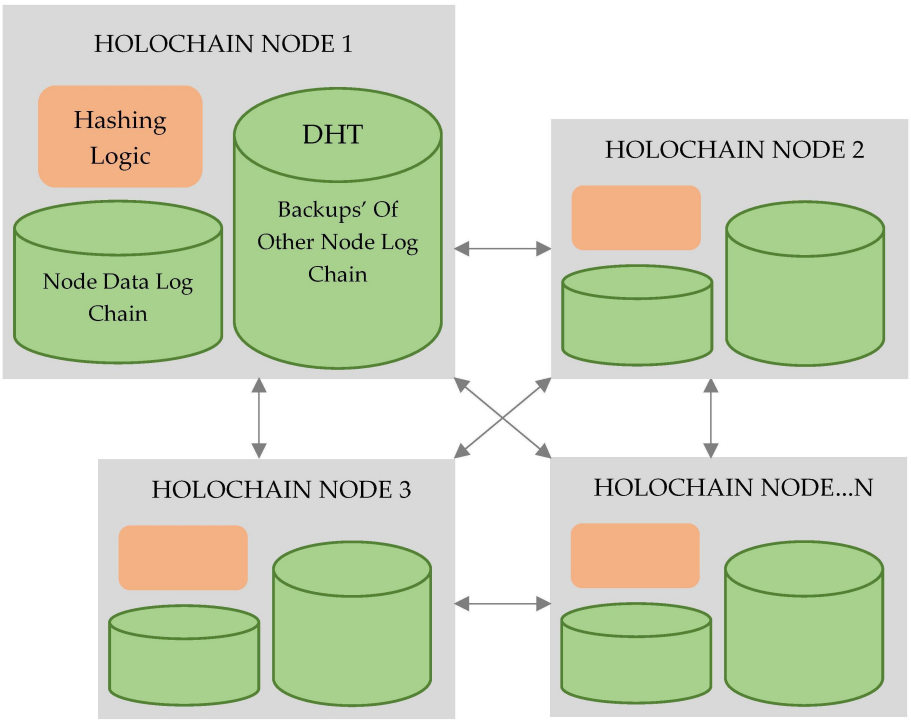

Figure 11. Log preservation in Holochain.

4.2.3. Creating Log Chains as Proof of Past Logs

The fog worker node receives the edge nodes logs from automated bots and builds the log info digest. Log info digest comprises of some attributes which are log_file_ID, bot_ID, container_IP, and timestamp. These log info digests being stored at local storage of respective fog worker node along with their private keys and peer node log chain ID. Finally, all of the fog nodes stores their respective encrypted local log chains. Figure 12 illustrates the procedure of the digest building of local log chains.

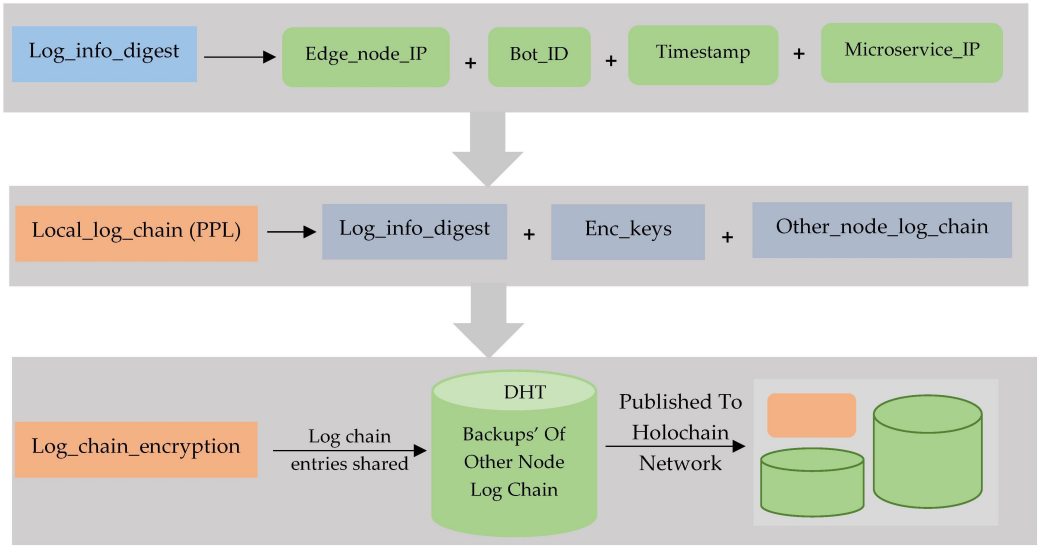

Figure 12. Preserving proof via log chaining.

4.2.4. Distributed Hash Tables (DHTs)

Fog worker logs the local backup of log info digest and then exchanges its local log chain to the peer network fog worker node. In DHTs, a single key that will be "IP:Port" of that node with the SHA-1 algorithm will be allocated to each node. A node may construct a DHT ring and other nodes may join the ring through the IP:Port of this node. When a node enters or joins a ring, some threads are created to ensure correct entries in a finger table. The successor and predecessor fog nodes share and acknowledge the local log chain of each other. There is a list of successors with "R" entries in each node. This list is used when a successor node leaves the ring, and the next node in the successor list will automatically be assigned as the successor to it. A growing node periodically asks the successor and predecessor to remember that they remained in the ring. If no acknowledgment is earned, the ring is left and stabilization is carried out accordingly. Every node maintains a map to store pairs of the key values. When a key is accessed, a single ID is often allocated and stored in a node whose ID is only greater than the ID of that key. If a node again leaves the ring, it gets all the keys from its successor.

4.2.5. Log Archiving Layer (Layer Three)

In this layer, all of the collected logs are securely stored at CLS. This layer affects the session authentication when log retrieval is requested and secure storage of logs as well. The aforementioned operations are executed under the Secure Data Archiving Module (SDAM). These stored logs can be used for testimonial in the court of law, in the recreation of footprints and for threat intelligence analytics. SDAM performs two following operations of log securing and secure session establishment.

Under the considered system model and security requirements for secure and incremental log storage, our goal is to propose a security and privacy-aware storage scheme based on homomorphic encryption in the cloud. The cloud storage is susceptible to side-channel attacks, fake information injection attacks, and data forgery attacks. To prohibit previously mentioned threats in the multi-tenant and black-box nature of the cloud, we used a Partial homomorphic encryption mechanism based Paillier Homomorphic Encryption (PHE) scheme for privacy and security of logs. The purpose of using PHE is to attain the privacy and security of logs in cloud storage while the continuous updates of logs from the fog node controller. PHE is fast and provides additive operations upon logs file increment in cloud storage and also provides semantics security [56].

The public and private keys of CLS are directly stored in the premises of the law enforcement agencies to avoid any kind of conspiracy. There are three main steps involved in this scheme namely Key generation, Encryption, and Decryption. The underlying steps in each of these steps are explained

as follows; in the key generation step, the public (encryption) and private (decryption) are generated and stored; in step two, the computation performs on encrypted data, in step three, decryption of both plain text is performed via private keys, this phase is only done at the agent's machine. Figure 13 provides a simple explanation of PHE working in the context of log storage. The benefit of using PHE is that, even if the attacker intercepts the communication between layer two and layer three, she cannot obtain the sensitive information belongs to logs.

Whenever the investigating entity makes requests to fetch and validate the integrity logs from CLS, a secure session is initiated amid entity, cloud, and fog node controller. These sessions are authenticated by SSL (Secure Socket Layer) using curve25519 as a cryptographic protocol for mutual authentication. The SSL with curve25519 cryptographic protocol offers better security with faster performance compared to RSA, it is compact, and uses only 68 characters compared to RSA 3072 that has 544 characters. RSA is the most widely used public-key algorithm for SSH keys. However, compared to curve25519, it is slower and even considered not safe if it is generated with the key smaller than 2048-bit length [57].

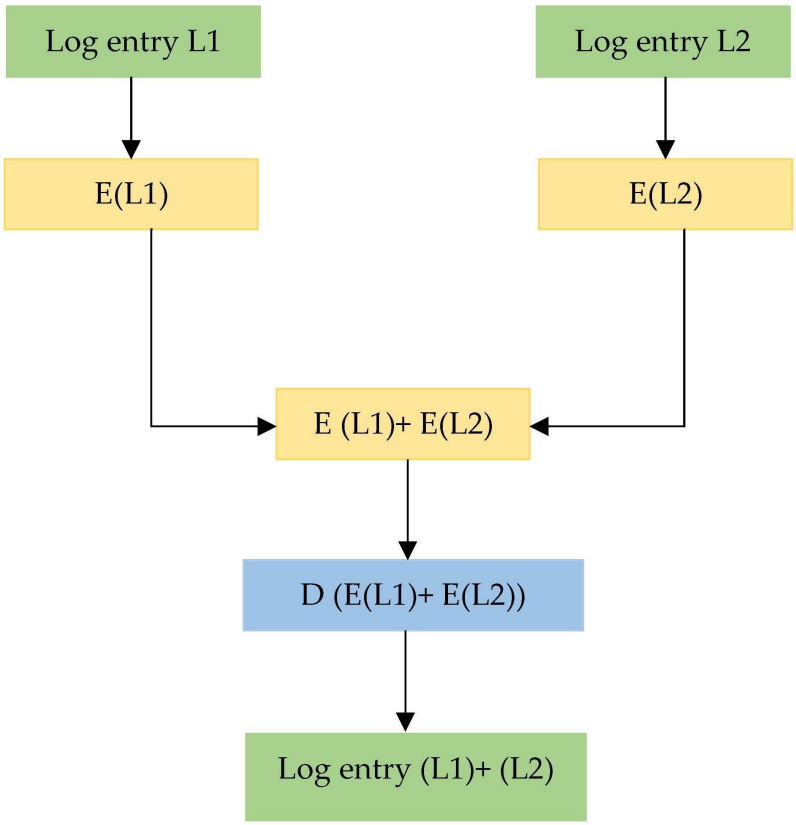

Figure 13. Paillier Homomorphic Encryption (PHE) mechanism in cloud CLS.

The well-thought-out mutual authentication mechanism executes by SDAM in the following steps; the first session of authentication is established among investigators and CLS. When the investigator is verified as a legitimate entity at the CLS verification point, it will be able to request the PPL from the fog node controller via the same authentication scheme used at the cloud level. After this phase of entity authentication, the CLS regenerates the tokens and certificates specifically for the fog node controller. This mutual entity public-key authentication process is presented in Figure 13. In two-way SSL authentication, the SSL client (agent) verifies the identity of the SSL server (CLS), and the SSL server (CLS) verifies the identity of the SSL client (agent). The mutual authentication scheme provides in Figure 14 works in subsequent steps;

1. SSL client (agent) sends a secure connection request to CLS

2. CLS reply with its certificate (CA) and request for agent certificate (CB)
3. Agent verifies the certificate (CA)
4. Key exchange from SSL client (agent) side
5. Agent sends its certificate (CB) signed by its private key
6. CLS receives and verifies the certificate (CB) from public keys of agent
7. Exchange of cipher specs at both SSL client and server-side and fog node controller
8. Secure session initiated.

After the session begins to initiate, the agent gets the encrypted logs and decrypted them on its machine. Here, the third phase of PHE occurs and log validation as well.

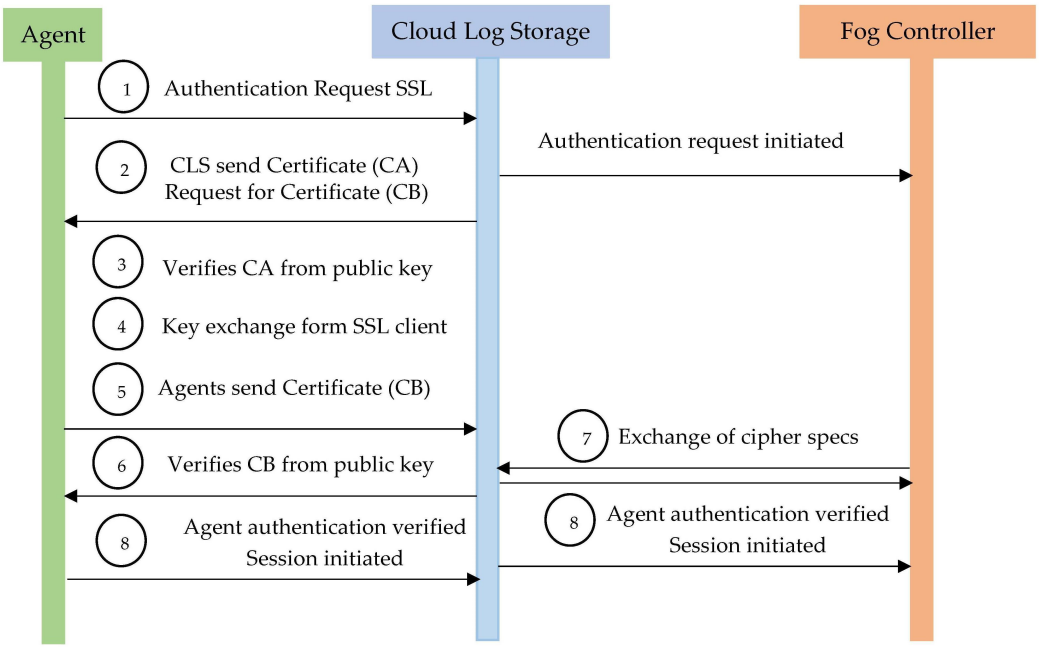

Figure 14. SSL mutual authentication amid agent, cloud and fog node controller.

5. Performance Evaluation and Security Analysis

This section explains the implementation details, performance evaluation, and security analysis of PLAF. The performance analysis is performed based on stress testing and log preservation processing analysis. In security analysis, the formal log integrity verification and validation are performed. In addition, the comparison of PLAF with existing literature is provided.

5.1. Implementation

To implement a prototype environment, the technical PLAF specifications for the host machine are Ubuntu 20.04 LTS system with 16 GB RAM, and a Core i7 Processor. We used the OpenStack cloud platform to host a fog node controller and the Docker swarm management console is used to orchestrate IoT services in fog node; C&C scripts are written in Python to fetch and collect the logs automatically. For the verification of the log hash chain based scheme, Holochain rust API is implemented to create the PPL. To achieve the confidentiality of data, we used PHE for encryption. The authentication of the user is achieved through SSL with curve25519 for mutual entity authentication. The real-time testbed is implemented using IoT devices, the OpenStack cloud platform, and a Raspberry Pi. This testbed is cross-verified from the given threat model and also the simulation results are compared with existing literature. In the following steps, the testbed is described.

Step 1: Layer one is implemented in this phase where different logs of IoT environment are generated and collected. We used different IoT devices such as IP cameras, fingerprint scanners,

and Wi-Fi enabled devices to get the logs. In the second phase of layer one simulation, the orchestration of IoT device node microservices was performed via the Docker swarm platform for continuous integration and privacy-preserving automation. In the Docker swarm environment, different containers are deployed to orchestrate microservices of IoT devices. For continuous and autonomous log collection, the C&C bots application was deployed in the container of each microservice. These automated bots are actually the Python script which uses the C&C mechanism to get the logs continuously. Each bot has a unique ID which is allotted via a nonce key so no malicious bot can access the container.

Step 2: Layer two which is a log proof preservation layer has been instigated in step two of implementation. This step is performed at fog node layer PLAF. We used a Raspberry Pi as a fog node because our adopted preservation scheme can easily run on less resource fog nodes. We used Holochain container-API written in rust for log preservation in different fog nodes. We used the OpenStack cloud platform to simulate the fog node controller which governs the phenomenon of Docker swarm management, fog node monitoring, and DHTs management of different fog nodes. In fog node controller, monitoring of the following modules was implemented; DSMM for microservice orchestration, OpenIoTFog agent (CMM) to monitor the health of IoT devices, data preservation module to effect Holochain mechanism and secure migration of log from fog to cloud.

Step 3: Secure log archiving is deployed in dedicated cloud log storage and is created in the cloud platform. We used Paillier Homomorphic Encryption to secure the log storage and private keys are directly stored in the respective monitoring agent server. SSL with curve25519 for mutual authentication is used for entity authentication when the session is established at CLS and fog node.

5.2. Performance Analysis

This section provides the evaluation of securing logs in PLAF. Fast data processing and data integrity checking are critical because it aims to collect log data in fog enabled cloud environments. Furthermore, the minimal computational power and overhead at fog node have to be calculated during log preservation. The efficiency is measured using four success metrics: fog node automation processing, stress testing of bot running in container besides microservices, privacy preservation processing, and log validation analysis.

5.2.1. Use Case

We built a testbed workload of IoT scenarios to evaluate the efficiency of PLAF. The testbed is modeled on the IoT application environment based on microservices which automatically collects data and sends it to the fog node. There are situations in which a fog node has to be attached physically to the computers. Applications also require access to other resources than CPU, memory, and storage, and serial ports. When requirements for containers are created by DSMM, together the DSMM and CMM ensure that the required resource container is available. When an IoT device connects to the PLAF environment, the requested IoT microservices are placed in the container as shown in Figure 15. The containerized program will also be designed for that particular fog node. The DSMM must facilitate this form of container positioning and CMM must be conscious of the robustness of the device.

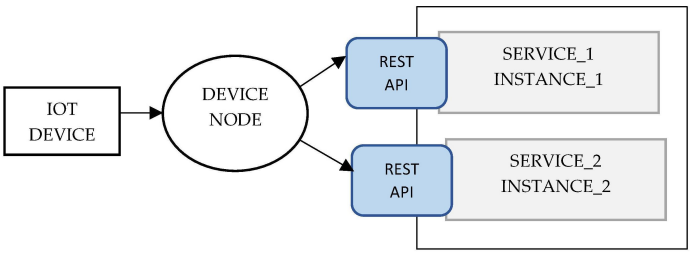

Figure 15. Placement of microservice IoT device.

5.2.2. Fog Level Privacy Automation Processing and Testing

A resource-constrained computer such as a Raspberry Pi is used as a fog node. The DSMM must be able to automatically identify and connect the fog node to the cluster, and only the required program packages are included in this node. In terms of latency and network utilization of the device after deployment, we have tested our scenario with microservices utilizing the autonomous bots and contrasted it with a cloud-based application. The placement methodology is tested based on the time required in the placement of microservices in the fog layer and the cloud. The details are presented in Figure 16.

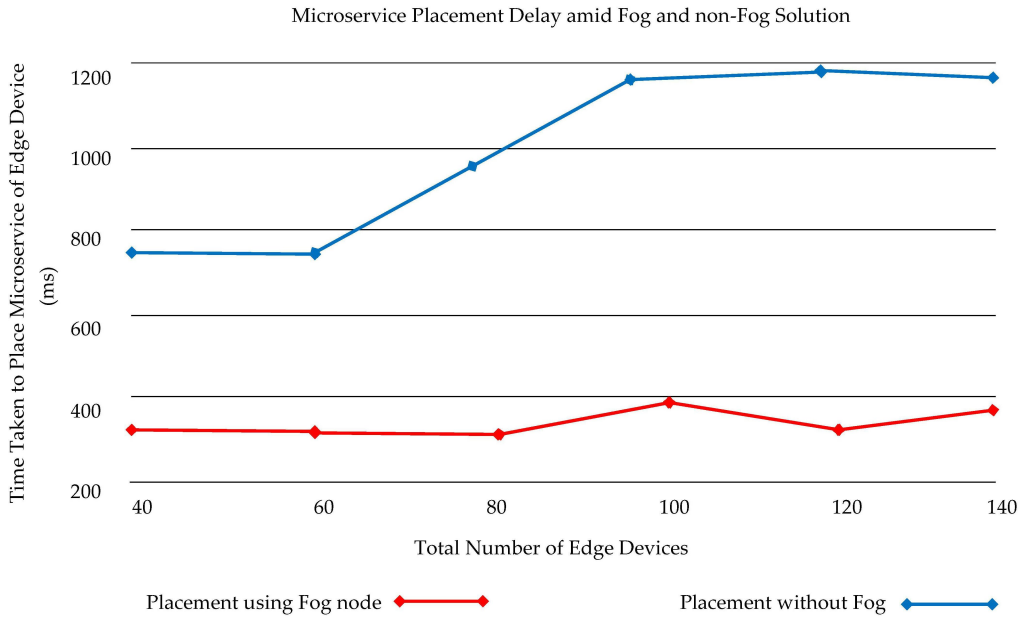

Figure 16. Microservice placement delay in the fog and cloud environment.

5.2.3. Stress Testing of Bots and Containers

The proposed architecture PLAF aims to automate the deployment of the bots in the fog environment which is composed of heterogeneous fog nodes with fewer resources. Therefore, the log harvesting mechanism of PLAF has been evaluated using two scenarios. Scenario 1: In the first scenario, the ZigBee devices were launched with bot application directly running on the host operating system, The time required to reach the ZigBee application is evaluated.

Scenario 2: In the second scenario, we did the calculations in a Docker container while executing the isolated bot program. Finally, in the containers, all gateways and bot applications were running together. The cycle has been replicated twenty times. Docker Containers have observed an overall overhead of about 0.039 sec. The confidence interval for both the scenarios is 95.7%.

The statistics provided in Table 4 demonstrate the outcomes of observations of two scenarios. We used a stress testing method for Linux where -c, -m, -io are all CPU, memory, I/O workload generator. Note that this lag is only noticed when the device is first accessed. The more stress exerted on the PLAF in the simulations, the more time it takes to launch the (Xbee) program. It was observed that, initially, the latency in usage is milliseconds of stress tests. This interruption takes place only once before the program begins and does not create additional overhead in this situation of the Docker container.

In our experiments, we have increased and decreased the number of sensors in the fog layer to evaluate the performance. We also experimented with the deployment of more edge devices attached to each fog node and additionally allocated a few more resources. Table 5 specifies the criteria of simulation for a minimal period to start a container and microservices in the architecture of PLAF.

Figures 17 and 18 provide the delay computed after increasing the number of IoT devices and the placement of microservices in them.

Table 4. Communication delay with bots in container with respect to stress parameters.

Scenario 1: Placement of Bots application without container		
Device Configuration	**Average Time (sec)**	**Confidence Interval (95%)**
Gateway APP in container	0.064144	0.066 < x < 0.069
Bot + APP in container	0.09100	0.091 < x < 0.099
Scenario 2: Placement of Bots application with container		
Bot in container stress (-c)	0.0789271	0.088 < x < 0.089
Bot in container stress (-m)	0.0896683	0.081 < x < 0.093
Bot in container stress (-io)	0.0995469	0.066 < x < 0.096

Table 5. Simulation Parameters.

Parameter	**Value**
IoT device to fog level (one device)	6.5 ms
IoT device to fog level (four device)	22 ms
IoT device to fog level (eight device)	33.2 ms
Container startup time	199 ms
Placement of Microservice in fog node	2 ms

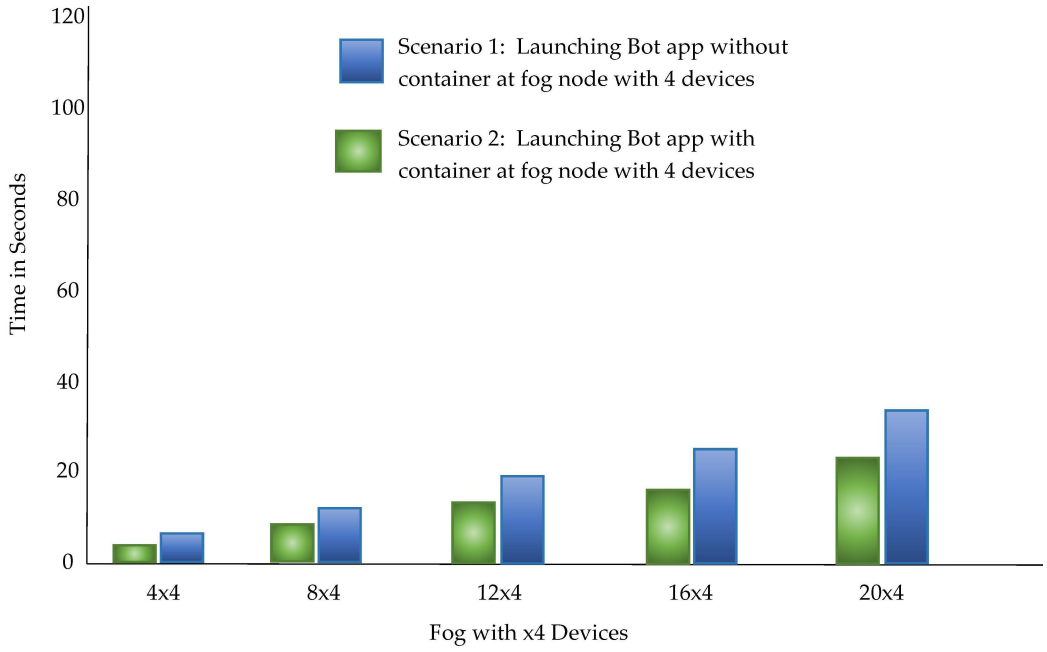

Figure 17. Time taken to place microservice in fog node with four devices.

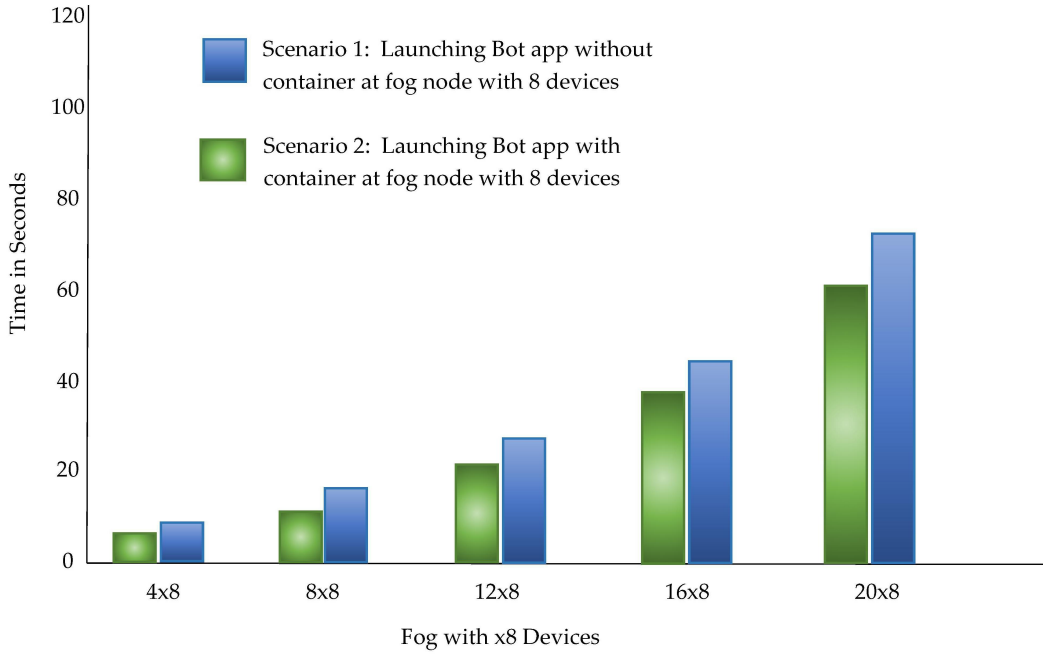

Figure 18. Time taken to place microservice in fog node with eight devices.

5.2.4. Log Preservation Processing Analysis

As previously described, the proposed architecture PLAF applies a Holochain mechanism for the preservation of logs at distributed fog nodes. We compare the PPL preservation performance in terms of CPU resources for the RSA, Blockchain, and Holochain. We used 10, 100, and 1000 KB of the log files with 200 log entries each. The log preservation duration in fog node is shown in Figure 19, which describes that the log protection using Holochain was found to be better than the blockchain and RSA encryption.

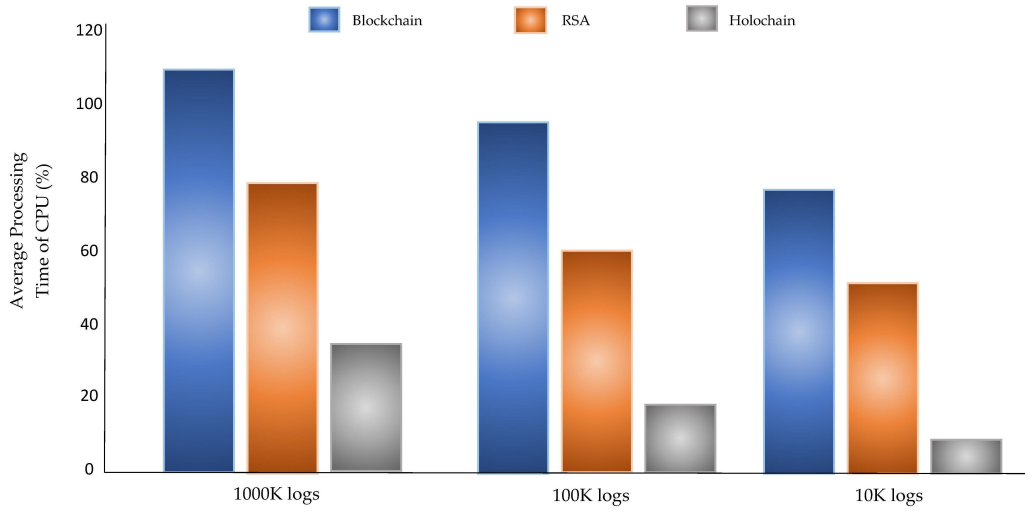

Figure 19. Log preservation performance in RSA, Blockchain, and Holochain.

5.2.5. Secure Log Storage Processing Analyses

PLAF stores the actual log data at the third layer of architecture via PHE (Paillier Homomorphic Encryption) and provides the secure session establishment for log retrieval as well. We have analyzed the privacy-aware secure log storage at CLS and the time taken to perform the incremental addition.

The analysis of average time execution for PHE to execute incrementally is illustrated in Figure 20. Time taken to perform mutual authentication via SSL certificates using curve25519 is also shown in Figure 20.

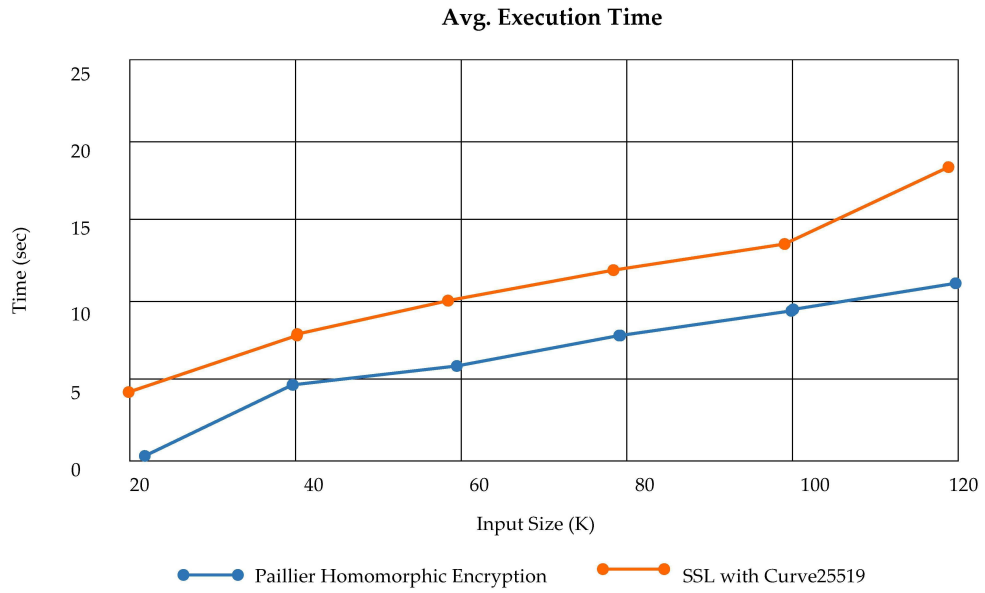

Figure 20. Secure log storage processing session establishment analyses.

5.2.6. Performance Validation of PPL Processing

Only the data created by geo-distributed sensors in cloud location is sent to a central cloud that is multi-hops from the edge of the network, Because all data produced are submitted to the cloud, the number of data streams through the central network arises, significantly increasing the network jamming. Due to these reasons, proof of past logs fetching services deployed in the cloud is recovered and validated considerably late. In addition, data size increases when encrypting and decrypting the data. Encrypting cloud logs is approximately three times expensive and larger files take more storage and time to validate them. In PLAF, the size of the log data during validation is not increased during the performance testing of the validation processing. Figure 21 provides the time consumed to validate the PPL in two scenarios: (i); fetching and validating the PPL from fog node; (ii) fetching and validating the PPL directly from the cloud.

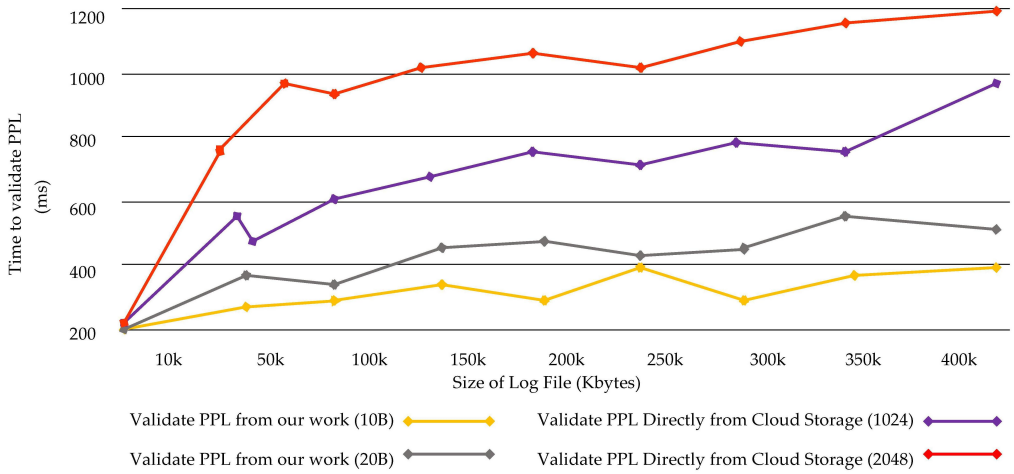

Figure 21. PPL validation processing performance among fog and cloud.

5.2.7. Computing Resource Allocation Trade-Offs

This work proposes a new log preservation architecture named PLAF offering many security aspects for log preservation in an automated manner. On the other hand, it also affects some computational trade-offs in terms of CPU and memory utilization. in this section, these trade-offs and computational costs are presented and elaborated.

CPU Utilization: The CPU utilization during experimentation and simulations were compared. The average CPU utilization for secure log storage using PHE and session establishment via SSL on log size from 10-120KB log size is given in Figure 22. We can see that at the cloud layer the average CPU utilization is approximately 40% of total CPU, which is an overhead. The CPU utilization at the fog level has also been analyzed based on the usage of Docker containers and automated bots for log collection. We have independently and separately analyzed the total CPU usage of Docker containers in fog nodes which is shown in Figure 23. As the Bots are running continuously in containers, they thus create an additional overhead in a container environment; in Figure 24, it is shown that the presence of bots in containers seeks more CPU usage.

Memory Utilization: The memory utilization of docker container without bots is very low as shown in Figure 25. As the data size increases after the deployment of bots, it requires more memory allocation than containers. The memory allocation of containers and bots has separately analyzed and described in Figure 26. The average memory utilization at the third layer of PLAF of log size up to 120 K shows that PHE and SSL both use memory with a minimum difference, which can be seen in Figure 27.

Complexity of Using Docker Containers in OpenStack: We have demonstrated the CPU and memory utilization of Docker containers and bots. Here, the resource usage terms of RAM of a docker container in a fog node controller are presented, which is an OpenStack cloud virtual machine. To build a fog node controller, we have used a virtual machine with the following specifications: Quad-core processor, 32 GB RAM, and 75 GB of storage. To test the docker usage according to OpenStack VM image specifications, we have executed two tests, which are resource complexity analysis on a large Host and resource complexity analysis on a small host. Results of testing using a large Host (OpenStack) shows that we can deploy a large number of containers, which means that a fog node controller must be equipped with maximum resource allocation. This phenomenon is shown in Figure 28.

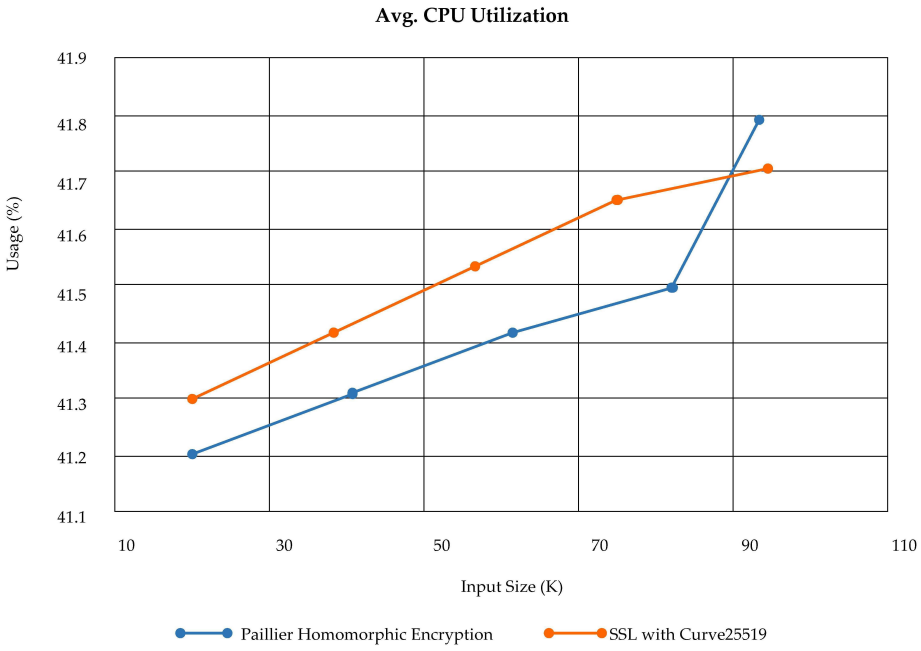

Figure 22. Average CPU utilization for secure log storage using PHE.

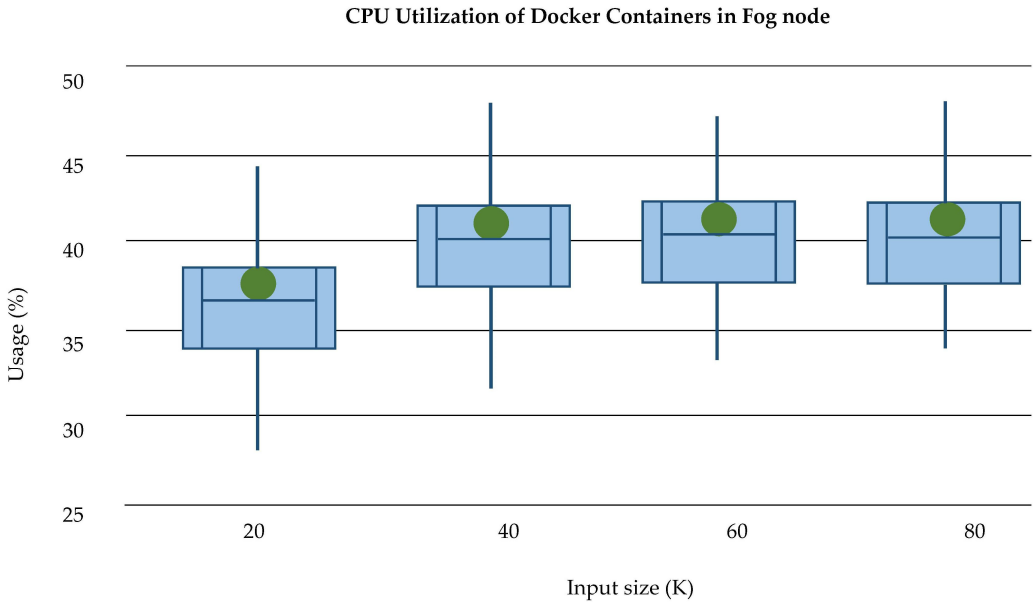

Figure 23. CPU utilization of docker containers in fog nodes.

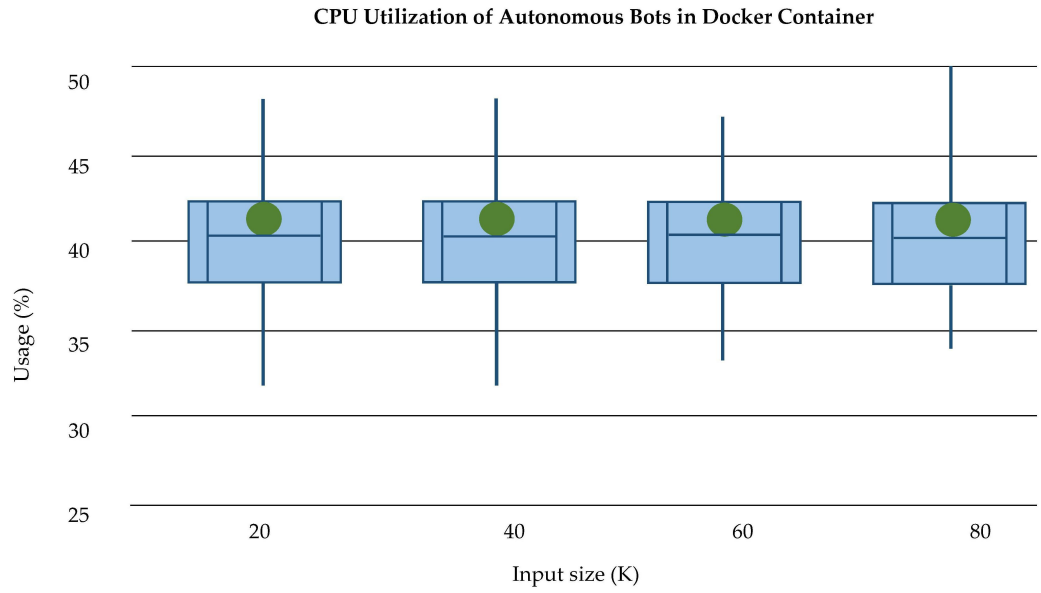

Figure 24. CPU utilization of autonomous bots in docker containers.

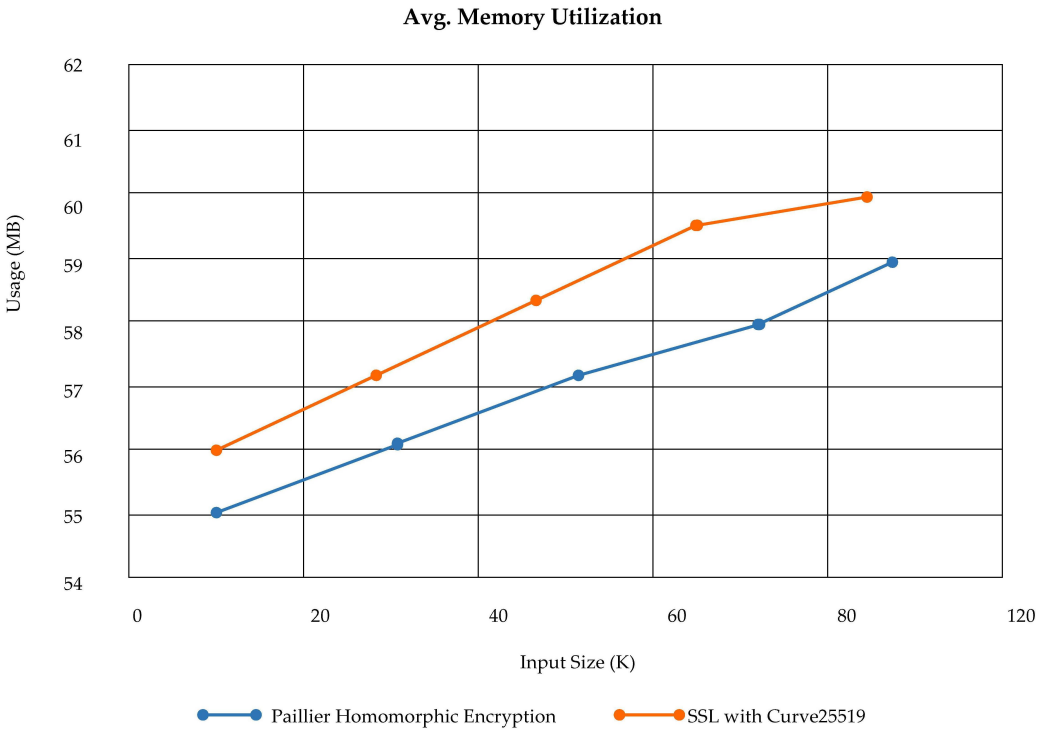

Figure 25. Average memory utilization of docker containers.

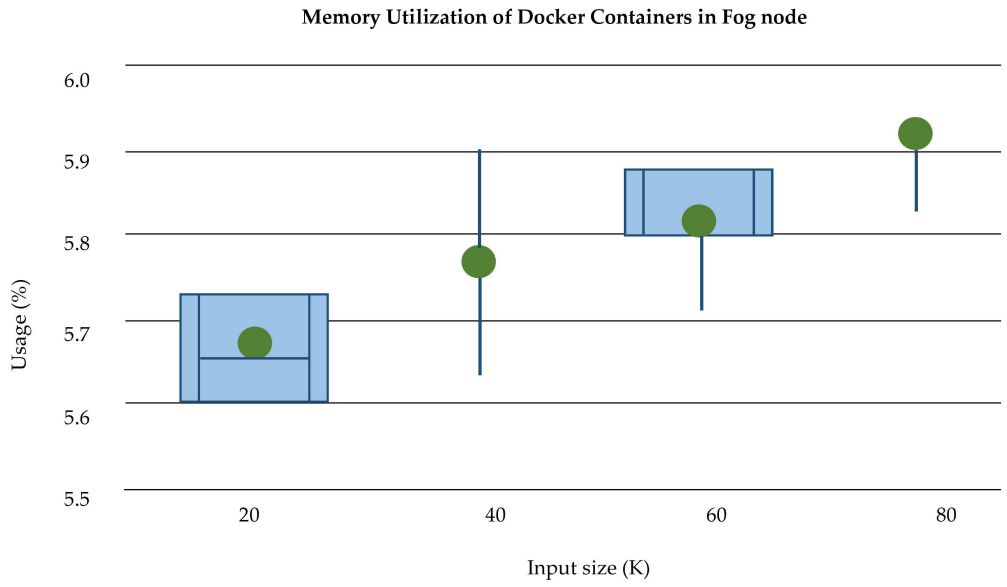

Figure 26. Average memory utilization of docker containers in fog nodes.

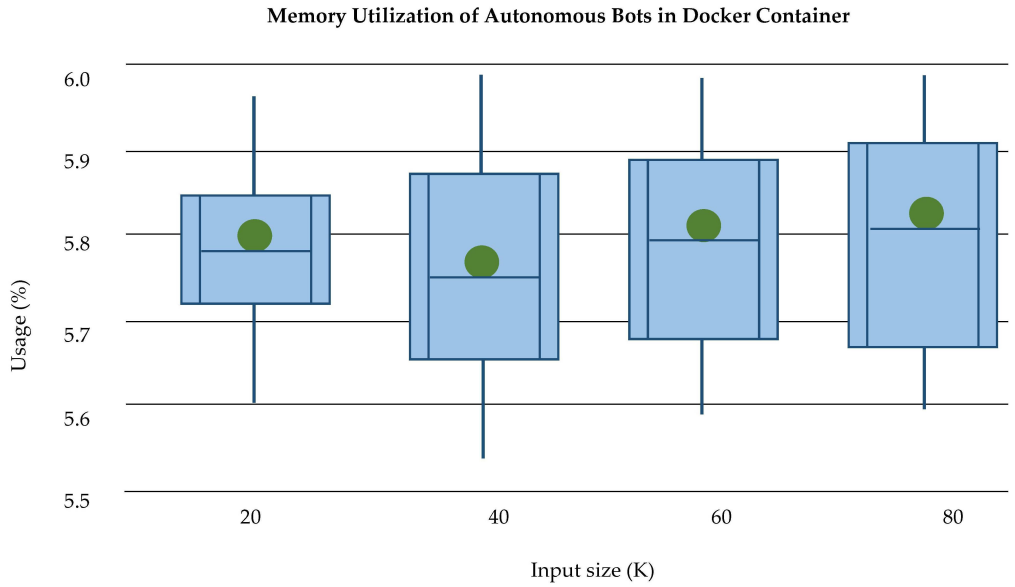

Figure 27. Average memory utilization of autonomous bots in docker containers.

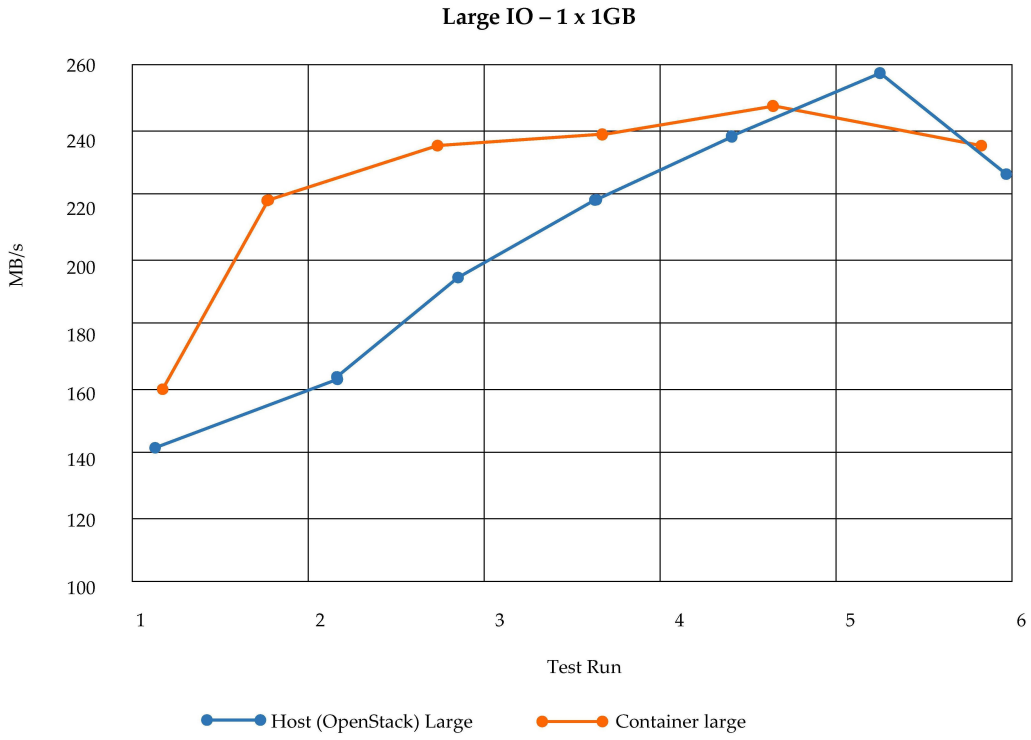

Figure 28. Container usage in Large Host (OpenStack).

Conversely, the deployment of a small host (OpenStack) shows that the inverse results as compared to a large host. We can conclude here that a docker container doesn't use many resources but is adaptable to host machine resources. The results of small host usage are shown in Figure 29.

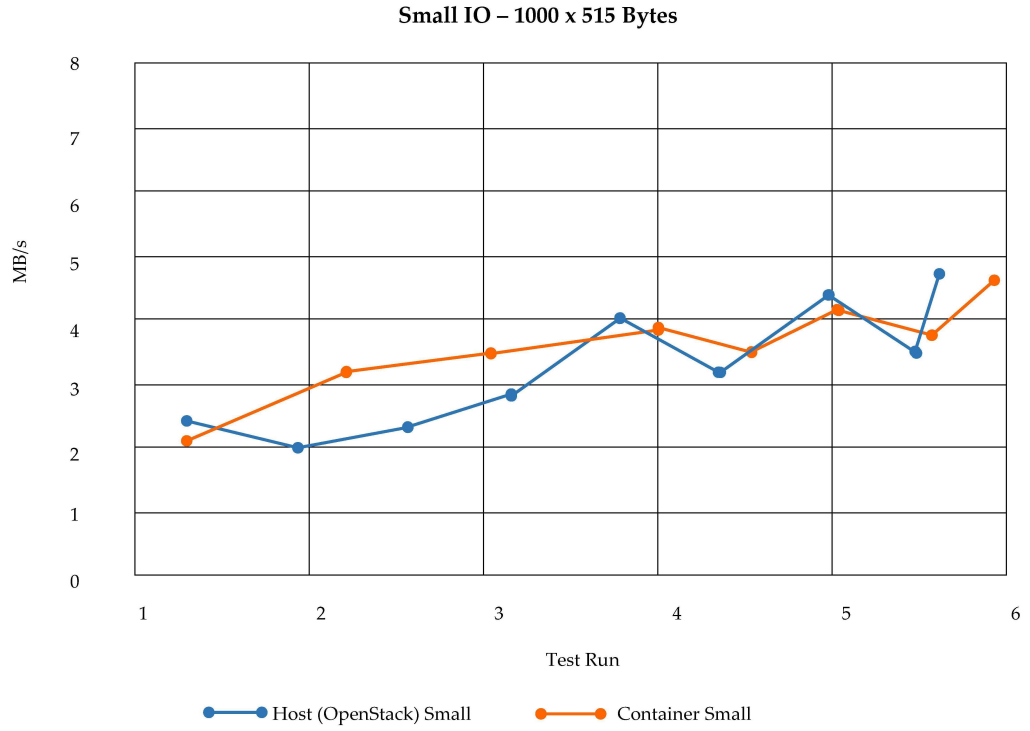

Figure 29. Container usage in Small Host (OpenStack).

5.3. Security Analysis

PLAF is designed for all the security attributes discussed previously and can address the issues in the existing literature. We summarize the limitations in existing schemes and explain how PLAF tackle these.

In the case of a typical cloud attack incident, an agent demands the CSP for a log database for integrity-based validation as forensic evidence. If log validity is not checked, it cannot be accepted and is not credibly acceptable as forensic evidence. However, PLAF guarantees log integrity, log verification, provenance, trust admissibility, temper resistance, ownership non-repudiation, and privacy-preserving automation using Holochain to check the log integrity over a distributed network of hash table at a fog level. We have performed and analyzed the security comparison of PLAF with all the existing literature techniques addressed in Section 2. From the analysis of literature, we have comprehended the essential security requirements for privacy and security-aware log preservation. The scrutinized security requirements were as follows; Ownership Non-Repudiation, Trust admissibility, the provenance of PPL, Temper Resistance, Log Integrity, Log Verifiability, and Privacy-Preservation Automation.

PLAF acquires all the aforementioned security requirements and can defeat each aspect of the threat model given in Section 3. A comprehensive comparison of PLAF with the existing literature is provided in Table 6. The essential security requirements are placed in columns; therefore, we aimed to comprehensively analyze the gains of PLAF with a lack of existing literature. The tick and cross signs indicate the presence and absence of security requirements. The security comparison in Table 6 describes the novice contributions offered by PLAF that outperform with all other existing work. In addition, Table 6 provides the security comparison in the context of essential scrutinized security requirements for log preservation. It can be observed that PLAF provides the following distinctions over existing techniques which are used for the securing of logs in the CLS, which is described in the following points:

Table 6. Security comparison PLAF with existing literature.

Scheme	Ownership Non-Repudiation	Trust Admissibility	Provenance of PPL	Temper Resistant	Log Integrity	Log Verifiability	Privacy Preservation Automation
[29]	✗	✗	✗	✗	✗	✗	✗
[30]	✗	✗	✗	✗	✗	✗	✗
[33]	✗	✗	✗	✗	✗	✗	✗
[31]	✗	✗	✗	✗	✗	✗	✗
[32]	✗	✗	✗	✗	✗	✗	✗
[34]	✗	✗	✗	✓	✓	✓	✗
[35]	✗	✗	✗	✓	✓	✓	✗
[36]	✗	✗	✓	✓	✓	✓	✗
[37]	✗	✗	✗	✗	✗	✗	✓
[38]	✗	✗	✗	✗	✓	✓	✗
[39]	✓	✗	✗	✗	✓	✗	✗
[40]	✗	✗	✓	✓	✓	✓	✗
[41]	✗	✗	✗	✓	✓	✓	✗
[42]	✓	✗	✗	✗	✓	✗	✗
[43]	✗	✓	✗	✓	✓	✓	✗
[44]	✗	✗	✓	✓	✓	✓	✗
[45]	✗	✗	✓	✓	✓	✓	✗
[46]	✗	✗	✓	✓	✓	✓	✗
[47]	✗	✗	✗	✓	✓	✓	✗
[48]	✗	✗	✗	✗	✗	✗	✗
[24]	✗	✓	✗	✓	✓	✓	✗
PLAF	✓	✓	✓	✓	✓	✓	✓

1. Independent autonomous forensic log collection framework assisted via automated bots.

   Previous workers secure the log data through data encryption and deception methods but heavily rely on their respective CSPs. PLAF incorporates the automation of edge node log collection, which provides integrity and privacy management.
2. Logging scheme that incorporates the security of edge nodes logs into the account.

   PLAF is formulated to hash the log chain and proof for log replication in such a way that log modification could be supervised throughout the log verification process.
3. Preventing the log compromise due to collusion between the owner CSP, the customer, and the investigators.

   Because all CSPs have encrypted database and PPL data, there is no possible way to avoid or check the involvement between owners, investigators, and CSPs with DB and the PPL records. However, in PLAF, the log chains of all DHTs cannot be modified from distributed fog nodes because it shares encrypted and distributed log chains with other network fog nodes.

5.3.1. Log Integrity Verification

In the case of an accident affecting classic cloud protection, a forensic authority first needs an integrity-based report from CSP for the processing of technical evidence and the interpretation of records. If the validity of the documents is not proven, they are not legitimately accurate and cannot be regarded as forensic evidence. Moreover, PLAF preserves the log integrity with the hash function of log chain digest and tests the accuracy logs using a distributed storage network of Holochain.

Fog nodes create log chain hash values Hashes of encrypted log chain (H(eLC)) to ensure the log integrity. Figure 30 describes how to log data from cloud and fog nodes can be verified for integrity. The investigators collect distributed log digest data and recover the log data using log chain validation to verify the validity of the log data. As the cloud storage data recovered comprises of encrypted log record (eLR)) and Hash of Log chain (H(eLR)), the collected (H(eLC)) data chain from fog node is pondered to see whether they match (H(eLR)). The validity of integrity will be validated if both information complements each other.

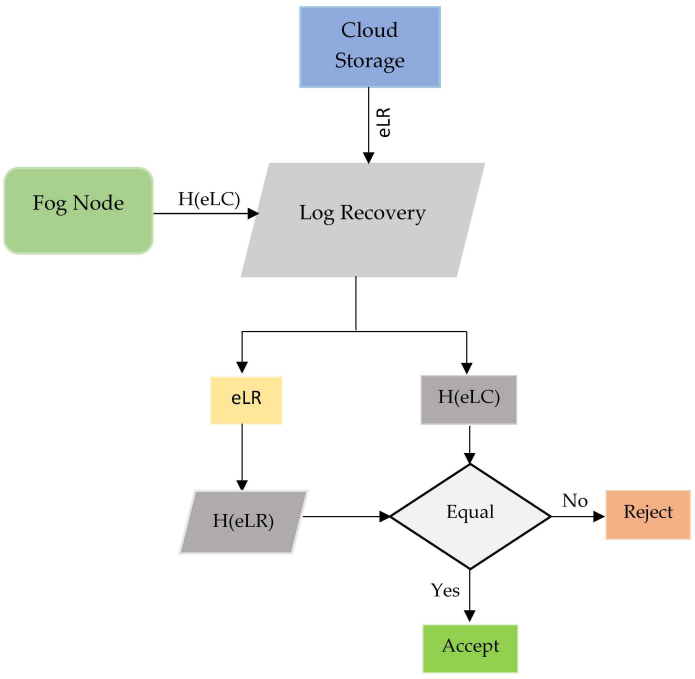

Figure 30. Log verification analysis.

5.3.2. Log Chain Validation

To ensure the correct order of each log file, we used Precision Time Protocol (ptp) to sync the same time. PLAF provides the accurate sync up time among fog nodes of log digest via PTP. The Precision Time Protocol (PTP) implementation [58] has ensured synchronization between clocks from all fog nodes. Furthermore, it should be noted that, before updating the log digest, the end-to-end time synchronization is achieved. This method permits the timestamps to be shared with fog worker nodes that can verify if the timestamp is altered or not. It also prohibits the tempered timestamp addition and publication.

Log Validation algorithm (LVA) with time complexity is the timestamp sharing and comparison method as described in Algorithm 1. Timestamp comparison is done to validate the correct order of log files. The input variables are: TBI (timestamp block identifier), $\text{PathGTB}_{1...n}$ (Path identifier of groups of all timestamp block), and MTID (Multi-timestamp path identifiers). LVA works on two phases, which are timestamp generation and sharing, and timestamp blocks for comparison or validation.

Algorithm 1: Log Validation Algorithm (LVA)

Result: Timestamp Comparison
Input : Input : TBI, $\text{PathGTB}_{1...n}$
Input : TBI (timestamp blocks Info)
Input : GTB (groups of timestamp block info)
Output: multiple timestamp comparison (MTID)

Input : timestamp generation and sharing
while *fog node* **do**
    TID = (BotID+TBI+logfileID);
    BTID = (TID | | $\text{PathGTB}_{1...n}$);
    send TID to Investigator;
end

Input : Timestamp Blocks for validation
while *fog node* **do**
    MTID = (Ordering(TID_1, TID_2, ... , TIDX_n) | | ($\text{PathGTB}_{1...n}$)) ;
    if $MTID_{1...n} \leftarrow BTID$ **then**
        log timestamps are SAFE ;
        publish results;
    else
        information is tempered or manipulated;
        publish results;
    end
end

The first phase of LVA generates TID (timestamp identifiers) using Bot_ID, Timestamp, logfile_ID. BTID (block of timestamps) used to publish these Timestamps to investigators. BTID is composed of TID and $\text{PathGTB}_{1...n}$ of all identifiers.

Log validation and comparison is done in phase two of LVA. Here, the correct orderings of all timestamp are compared to published BTIDs in a while loop. This phase provides information about log validation by confirming the order of timestamps. If the order and given timestamp is equal to timestamp under comparison, then it is declared as SAFE otherwise tempted:

$$T(n) = n(3+2) = 5n = O(N) \tag{2}$$

$$S(n) = n(1+1) + n = 3n = O(N) \tag{3}$$

We calculated the time and space complexity of both phases of LVA. The time complexity of phase one and two is described as in Equation (2), which shows $T(n)$ = each fog node(timestamp blocks

generation) + (timestamp comparison). The space complexity of phase one and two is $O(n)$ = each edge node(timestamp blocks generation) + (timestamp comparison) as given in Equation (3).

6. Discussion

Investigating various logs such as process logs and network logs plays a vital role in computer forensics. Collecting logs from a cloud is a challenging task due to the black-box nature and the multi-tenant cloud models, which can impinge on consumer privacy when accumulating logs. Moreover, the shifting paradigm of cloud computing toward fog computing brought new challenges for digital forensics. Smart nodes are more susceptible to security threats and with less computing resources, making this impossible to gather logs.

This work proposed a forensic aware logging architecture for security and privacy-aware log autonomous log preservation in fog enabled cloud federations. We also considered the secured and privacy concerned distributed edge node log collection by tackling the multi-stakeholder collusion problem. To address the problem domain of PLAF, we also compared the security requirements with the threat model and analyzed the required security controls. Based on scrutinized security controls, we have designed the security requirements via Secure Tropos methodology.

PLAF offers the seven essential security requirements needed against some security threats which are privacy violation, owner repudiation, log modification integrity theft, edge node tempering, and computation overhead at edge level. The following are the addressed security requirements: Ownership Non-Repudiation, Trust admissibility, provenance of PPL, Temper Resistance, Log Integrity, Log Verifiability, and Privacy Preservation Automation. PLAF architecture is comprised of three layers to mitigate the above-mentioned threats and affect the security requirements. The first layer provides privacy preservation automation to avoid edge level threats. The second layer mitigates the log integrity and privacy threats by implementing the Holochain distributed network that is not acquainted with existing Blockchain technology, which is a power-consuming approach. Ownership repudiation, privacy violation issues, and multi-stakeholder issues are dulled at the third layer of PLAF via Paillier Homomorphic Encryption for secure storage and SSL with curve25519 mutual authentication, respectively.

7. Conclusions and Future Directions

We have developed a testbed (PLAF Source Code—https://github.com/CanVel00/SALPAF) to implement the aforementioned specifications and requirements of PLAF via incorporating many state-of-the-art technologies at one place. We have analyzed PLAF in three dimensions, which are: testbed performance, computation resource allocation, and security. The results obtained from our experiments show a scalable and adaptable implementation of PLAF in terms of performance and security. On the other hand, PLAF shows computation trade-off in terms of CPU and memory utilization, especially for continuous log collection.

This paper presented new security and privacy-aware logging scheme in fog enabled clouds for digital forensics. However, there were several limitations to this study. Thus, potential future extensions may include advancing the operation of bots via machine learning techniques to detect the rouge edge devices via log analyses at the fog level. Similarly, the Design and Implementation of a real-world prototype of PLAF by incorporating the real-world scenarios and experimentation can be a promising direction. This will not only ensure but also define the multiple log formats to cover the log data of many services.

Author Contributions: Conceptualization, M.A.S. and K.J.; Formal analysis, M.A.S., K.J., and H.A.K.; Funding acquisition, A.A.; Investigation, K.J.; Methodology, M.A.S., K.J., and H.A.K.; Resources, I.U.D. and H.A.K.; Validation, I.U.D., A.A., M.A.S., and K.J.; Writing—original draft, K.J., C.M., and H.A.K.; Writing—review and editing, I.U.D., C.M., and H.A.K. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by King Saud University, Riyadh, Saudi Arabia, through Researchers Supporting Project number RSP-2020/184.

Acknowledgments: The authors acknowledge the support of King Saud University, Riyadh, Saudi Arabia, through Researchers Supporting Project number RSP-2020/184

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

Abbreviations

The following abbreviations are used in this manuscript:

Terms	Descriptions
LR	Log Record
eLR	Encrypted Log Record
H(eLR)	Hash of Encrypted Log Record
CLS	Cloud Log Storage
CSP	Cloud Service Provider
DSMM	Docker Swarm Management Module
CMM	Central Management Module
DPM	Data Preservation Module
SDAM	Secure Data Archiving Module
SSL	Secure Socket layer
PHE	Piallier Homomorphic Encryption
PPL	Proof of Past Log
C&C	Command and Control
PTP	Precision Time Protocol
LVA	Log Validaion Algorithm
DHTs	Distributed Hash Tables
STM	Secure Tropos Methodology
SecTro Tool	Secure Tropos Tool

References

1. Khattak, H.A.; Farman, H.; Jan, B.; Din, I.U. Toward integrating vehicular clouds with IoT for smart city services. *IEEE Netw.* **2019**, *33*, 65–71. [CrossRef]
2. Haseeb, K.; Almogren, A.; Ud Din, I.; Islam, N.; Altameem, A. SASC: Secure and Authentication-Based Sensor Cloud Architecture for Intelligent Internet of Things. *Sensors* **2020**, *20*, 2468. [CrossRef] [PubMed]
3. Haseeb, K.; Islam, N.; Almogren, A.; Din, I.U. Intrusion prevention framework for secure routing in WSN-based mobile Internet of Things. *IEEE Access* **2019**, *7*, 185496–185505. [CrossRef]
4. Servida, F.; Casey, E. IoT forensic challenges and opportunities for digital traces. *Digit. Investig.* **2019**, *28*, S22–S29. [CrossRef]
5. Vines, R.L.K.R.D.; Krutz, R. *Cloud Security: A Comprehensive Guide to Secure Cloud Computing*; Wiley Publishing, Inc.: Hoboken, NJ, USA, 2010.
6. Pourvahab, M.; Ekbatanifard, G. Digital Forensics Architecture for Evidence Collection and Provenance Preservation in IaaS Cloud Environment Using SDN and Blockchain Technology. *IEEE Access* **2019**, *7*, 153349–153364. [CrossRef]
7. Ali, Z.; Shah, M.A.; Almogren, A.; Ud Din, I.; Maple, C.; Khattak, H.A. Named Data Networking for Efficient IoT-based Disaster Management in a Smart Campus. *Sustainability* **2020**, *12*, 3088. [CrossRef]
8. Haseeb, K.; Islam, N.; Almogren, A.; Din, I.U.; Almajed, H.N.; Guizani, N. Secret sharing-based energy-aware and multi-hop routing protocol for IoT based WSNs. *IEEE Access* **2019**, *7*, 79980–79988. [CrossRef]
9. Awan, K.A.; Din, I.U.; Almogren, A.; Guizani, M.; Khan, S. StabTrust—A stable and centralized trust-based clustering mechanism for IoT enabled vehicular ad-hoc networks. *IEEE Access* **2020**, *8*, 21159–21177. [CrossRef]
10. Nieto, A.; Rios, R.; Lopez, J. Iot-forensics meets privacy: Towards cooperative digital investigations. *Sensors* **2018**, *18*, 492.10.3390/s18020492. [CrossRef]
11. Yaqoob, I.; Hashem, I.A.T.; Ahmed, A.; Kazmi, S.M.; Hong, C.S. Internet of things forensics: Recent advances, taxonomy, requirements, and open challenges. *Future Gener. Comput. Syst.* **2019**, *92*, 265–275. [CrossRef]

12. Din, I.U.; Guizani, M.; Hassan, S.; Kim, B.S.; Khan, M.K.; Atiquzzaman, M.; Ahmed, S.H. The Internet of Things: A review of enabled technologies and future challenges. *IEEE Access* **2018**, *7*, 7606–7640. [CrossRef]
13. Din, I.U.; Guizani, M.; Kim, B.S.; Hassan, S.; Khan, M.K. Trust management techniques for the Internet of Things: A survey. *IEEE Access* **2018**, *7*, 29763–29787. [CrossRef]
14. Asmat, H.; Din, I.U.; Ullah, F.; Talha, M.; Khan, M.; Guizani, M. ELC: Edge Linked Caching for content updating in information-centric Internet of Things. *Comput. Commun.* **2020**. [CrossRef]
15. Roman, R.; Lopez, J.; Mambo, M. Mobile edge computing, Fog et al.: A survey and analysis of security threats and challenges. *Future Gener. Comput. Syst.* **2018**, *78*, 680–698. [CrossRef]
16. Toor, A.; Ul Islam, S.; Sohail, N.; Akhunzada, A.; Boudjadar, J.; Khattak, H.A.; Din, I.U.; Rodrigues, J.J. Energy and performance aware fog computing: A case of DVFS and green renewable energy. *Future Gener. Comput. Syst.* **2019**, *101*, 1112–1121. [CrossRef]
17. Mukherjee, M.; Matam, R.; Shu, L.; Maglaras, L.; Ferrag, M.A.; Choudhury, N.; Kumar, V. Security and Privacy in Fog Computing: Challenges. *IEEE Access* **2017**, *5*, 19293–19304. [CrossRef]
18. Ali, W.; Din, I.U.; Almogren, A.; Guizani, M.; Zuair, M. A Lightweight Privacy-aware IoT-based Metering Scheme for Smart Industrial Ecosystems. *IEEE Trans. Ind. Inform.* **2020**. [CrossRef]
19. Haseeb, K.; Din, I.U.; Almogren, A.; Islam, N.; Altameem, A. RTS: A Robust and Trusted Scheme for IoT-based Mobile Wireless Mesh Networks. *IEEE Access* **2020**, *8*, 68379–68390. [CrossRef]
20. Din, I.U.; Guizani, M.; Rodrigues, J.J.; Hassan, S.; Korotaev, V.V. Machine learning in the Internet of Things: Designed techniques for smart cities. *Future Gener. Comput. Syst.* **2019**, *100*, 826–843. [CrossRef]
21. Masood, F.; Almogren, A.; Abbas, A.; Khattak, H.A.; Din, I.U.; Guizani, M.; Zuair, M. Spammer detection and fake user identification on social networks. *IEEE Access* **2019**, *7*, 68140–68152. [CrossRef]
22. Khattak, H.A.; Islam, S.U.; Din, I.U.; Guizani, M. Integrating fog computing with VANETs: A consumer perspective. *IEEE Commun. Stand. Mag.* **2019**, *3*, 19–25. [CrossRef]
23. Cha, H.J.; Yang, H.K.; Song, Y.J. A study on the design of fog computing architecture using sensor networks. *Sensors* **2018**, *18*, 3633. [CrossRef] [PubMed]
24. Park, J.; Huh, E.N. eCLASS: Edge-cloud-log assuring-secrecy scheme for digital forensics. *Symmetry* **2019**, *11*, 1192. [CrossRef]
25. Fan, Q.; Bai, J.; Zhang, H.; Yi, Y.; Liu, L. Delay-aware Resource Allocation in Fog-assisted IoT Networks Through Reinforcement Learning. *arXiv* **2020**, arXiv:2005.04097.
26. Wang, Q.; Chen, S. Latency-minimum offloading decision and resource allocation for fog-enabled Internet of Things networks. *Trans. Emerg. Telecommun. Technol.* **2020**, e3880. [CrossRef]
27. Mukherjee, M.; Kumar, S.; Shojafar, M.; Zhang, Q.; Mavromoustakis, C.X. Joint Task Offloading and Resource Allocation for Delay-Sensitive Fog Networks. In Proceedings of the IEEE International Conference on Communications, Shanghai, China, 20–24 May 2019. [CrossRef]
28. Wang, Y.; Uehara, T.; Sasaki, R. Fog computing: Issues and challenges in security and forensics. In Proceedings of the 2015 IEEE 39th International Computer Software and Applications Conference, Taichung, Taiwan, 1–5 July 2015; Volume 3, pp. 53–59. [CrossRef]
29. Zafarullah.; Anwar, F.; Anwar, Z. Digital forensics for Eucalyptus. In Proceedings of the 2011 9th International Conference on Frontiers of Information Technology, FIT 2011, Islamabad, Pakistan, 19–21 December 2011; pp. 110–116. [CrossRef]
30. Patrascu, A.; Patriciu, V.V. Logging for cloud computing forensic systems. *Int. J. Comput. Commun. Control* **2015**, *10*, 222–229. [CrossRef]
31. Sibiya, G.; Fogwill, T.; Venter, H.S.; Ngobeni, S. Digital forensic readiness in a cloud environment. In Proceedings of the IEEE AFRICON Conference, Pointe-Aux-Piments, Mauritius, 9–12 September 2013. [CrossRef]
32. Dykstra, J.; Sherman, A.T. Design and implementation of FROST: Digital forensic tools for the OpenStack cloud computing platform. *Digit. Investig.* **2013**, *10*, S87–S95. [CrossRef]
33. Marty, R. Cloud application logging for forensics. In Proceedings of the ACM Symposium on Applied Computing, TaiChung, Taiwan, 21–24 March 2011; pp. 178–184. [CrossRef]
34. Kumar, A.; Xu, J.; Wang, J.; Spatschek, O.; Li, L. Space-code bloom filter for efficient per-flow traffic measurement. In Proceedings of the IEEE INFOCOM, Hong Kong, China, 7–11 March 2004; Volume 3, pp. 1762–1773. [CrossRef]

35. Kebande, V.R.; Venter, H.S. A cloud forensic readiness model using a Botnet as a Service. In Proceedings of the international conference on digital security and forensics (DigitalSec2014), Ostrava, Czech Republic, 24–26 June 2014; The Society of Digital Information and Wireless Communication: Ostrava, Czech Republic, 2014; pp. 23–32.
36. Liu, Z.; Zou, H. Poster: A proactive cloud-based cross-reference forensic framework. In Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, Scottsdale, AZ, USA, 3–7 November 2014; pp. 1475–1477.
37. Kebande, V.; Venter, H.S. A functional architecture for cloud forensic readiness large-scale potential digital evidence analysis. In Proceedings of the European Conference on Information Warfare and Security, ECCWS, Hatfield, UK, 2–3 July 2015; pp. 373–382.
38. Lin, C.Y.; Chang, M.C.; Chiu, H.C.; Shyu, K.H. Secure logging framework integrating with cloud database. In Proceedings of the International Carnahan Conference on Security Technology, Taipei, Taiwan, 21–24 September 2015; pp. 13–17. [CrossRef]
39. Zawoad, S.; Hasan, R. FAIoT: Towards Building a Forensics Aware Eco System for the Internet of Things. In Proceedings of the 2015 IEEE International Conference on Services Computing, SCC 2015, New York, NY, USA, 27 June–2 July 2015; pp. 279–284. [CrossRef]
40. Zawoad, S.; Dutta, A.K.; Hasan, R. Towards Building Forensics Enabled Cloud Through Secure Logging-as-a-Service. *IEEE Trans. Dependable Secur. Comput.* **2016**, *13*, 148–162. [CrossRef]
41. Kebande, V.R.; Venter, H.S. On digital forensic readiness in the cloud using a distributed agent-based solution: Issues and challenges. *Aust. J. Forensic Sci.* **2018**, *50*, 209–238. [CrossRef]
42. Ahsan, M.A.M.; 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. [CrossRef]
43. McCabe, J.D. *Security and Privacy Architecture*; Elsevier: Amsterdam, The Netherlands, 2007; pp. 359–383. [CrossRef]
44. Li, J.; Liu, Z.; Peng, H. *Security and Privacy in New Computing Environments: Second EAI International Conference, SPNCE 2019, Tianjin, China, April 13–14, 2019, Proceedings*; Springer: Berlin, Germany, 2019; Volume 284.
45. Hossain, M.; Karim, Y.; Hasan, R. FIF-IoT: A forensic investigation framework for IoT using a public digital ledger. In Proceedings of the 2018 IEEE International Congress on Internet of Things (ICIOT), San Francisco, CA, USA, 2–7 July 2018; pp. 33–40.
46. Babun, L.; Sikder, A.K.; Acar, A.; Uluagac, A.S. IoTDots: A Digital Forensics Framework for Smart Environments. *arXiv* **2018**, arXiv:1809.00745.
47. Wang, Q.; Hassan, W.U.; Bates, A.; Gunter, C. Fear and Logging in the Internet of Things. *Netw. Distrib. Syst. Symp.* **2018**. [CrossRef]
48. Tariq, N.; 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]
49. Al-Garadi, M.A.; Hussain, M.R.; Khan, N.; Murtaza, G.; Nweke, H.F.; Ali, I.; Mujtaba, G.; Chiroma, H.; Khattak, H.A.; Gani, A. Predicting cyberbullying on social media in the big data era using machine learning algorithms: Review of literature and open challenges. *IEEE Access* **2019**, *7*, 70701–70718. [CrossRef]
50. Manzoor, A.; Shah, M.A.; Khattak, H.A.; Din, I.U.; Khan, M.K. Multi-tier authentication schemes for fog computing: Architecture, security perspective, and challenges. *Int. J. Commun. Syst.* **2019**, e4033. [CrossRef]
51. NIST. The NIST Definition of Cloud Computing. *Obs. Econ. EEUU* **2016**, *800*, 1–8. [CrossRef]
52. Pavlidis, M.; Islam, S. SecTro: A CASE tool for modelling security in requirements engineering using Secure Tropos. *CEUR Workshop Proc.* **2011**, *734*, 89–96.
53. OpenIoTFog—Making Your Shop Floor Industry 4.0 Ready. Available online: https://openiotfog.org/ (accessed on 18 July 2020).
54. Brock, A.; Atkinson, D.; Friedman, E.; Harris-Braun, E.; Mcguire, E.; Russell, J.M.; Perrin, N.; Luck, N.; Harris-Braun, W. *Holo Green Paper*; Technical Report March; Holo Host: Holo Chain, CA, USA, 2018.
55. GitHub—Holochain/holochain-Rust. Available online: https://github.com/holochain/holochain-Rust (accessed on 18 July 2020).
56. Makkaoui, K.E.; Ezzati, A.; Beni-Hssane, A.; Ouhmad, S. A swift Cloud-Paillier scheme to protect sensitive data confidentiality in cloud computing. *Procedia Comput. Sci.* **2018**, *134*, 83–90. [CrossRef]

57. Adamantiadis, A. Secure Shell (SSH) Key Exchange Method Using Curve25519 and Curve448. 00000 Library Catalog: tools.ietf.org. Available online: https://tools.ietf.org/id/draft-ietf-curdle-ssh-curves-10.html (accessed on 18 July 2020).
58. GitHub—ptpd/ptpd: PTPd Official Source—Master Branch a.k.a. Trunk. Available online: https://github.com/ptpd/ptpd (accessed on 18 July 2020).

Sample Availability: Samples of the experimental data are available from the authors upon reasonable request.