Research Article

Prasad Purnaye* and Vrushali Kulkarni

BiSHM: Evidence detection and preservation model for cloud forensics

https://doi.org/10.1515/comp-2022-0241
received March 14, 2021; accepted April 12, 2022

Abstract: The cloud market is growing every day. So are cloud crimes. To investigate crimes that happen in a cloud environment, an investigation is carried out adhering to the court of law. Forensics investigations require evidence from the cloud. Evidence acquisition in the cloud requires formidable efforts because of physical inaccessibility and the lack of cloud forensics tools. Time is very crucial in any forensic investigation. If the evidence is preserved before the cloud forensic investigation, it can give the investigators a head start. To identify and preserve such potential evidence in the cloud, we propose a system with an artificial intelligence (AI)-based agent, equipped for binary classification that monitors and profiles the virtual machine (VM) from hypervisor level activities. The proposed system classifies and preserves evidence data generated in the cloud. The evidence repository module of the system uses a novel blockchain model approach to maintain the data provenance. The proposed system works at the hypervisor level, which makes it robust for anti-forensics techniques in the cloud. The proposed system identifies potential evidence reducing the effective storage space requirement of the evidence repository. Data provenance incorporated in the proposed system reduces trust dependencies on the cloud service provider (CSP).

Keywords: artificial intelligence, cloud computing, machine learning, evidence acquisition, digital forensics

1 Introduction

Cloud forensics is an investigation of a cyber-crime committed in a cloud cyber-space. Cyber-crimes in cloud-cyber space can be committed by using the cloud as a subject, as an object, or as an environment. The rate at which data is being generated is as huge as 2.5 quintillion bytes per day [1]. The majority of these data are generated in a cloud platform. A cloud forensic investigation needs to collect evidence from such a huge pile of data. Knowing only the precise data that can potentially contribute as evidence to the forensic investigation can save a lot of data processing as well as time. This gives the advantage of the time to the investigation. Such classification is also valuable when it is used to preserve volatile data as evidence. Evidence acquisition presents challenges of data gravity and secure storage [2–4]. The chain of custody must be preserved throughout all the forensic investigation stages, which can also be difficult in a cloud scenario.

We propose a system equipped with an AI agent, which incorporates a Binary classification of Evidence using Smart Hypervisor Monitoring (BiSHM) to classify the data as evidence data. Our novel approach, unlike most of the current solutions in the market, does not require an agent to be installed inside the virtual machine (VM) to monitor data or metadata. The monitoring sensors of the modelled AI agent reside at the hypervisor level. This placement of the sensors of the AI agent ensures minimum privacy intrusion in the cloud. Malicious behaviour of the VM is detected with hypervisor level monitoring [5]. All the data including volatile data of the VM generated during the time of the event is collected as evidence. The evidence is then stored securely with a blockchain framework to ensure its integrity.

The BiSHM model will preserve only the required evidence data generated during an attack originating from or terminating at a VM. The chain of custody of the evidence is safeguarded by the blockchain framework. The authors have also contributed a novel dataset for cloud forensics.

The rest of this article is organized as follows: Section 2 discusses the cloud forensic scenario in the cloud. Section 3 details the problem statement. Section 4 describes the proposed system. Modelling of the proposed agent is also elaborated in Section 4. In Section 5, we have explained the dataset, which was used as a part of the

* **Corresponding author: Prasad Purnaye,** School of Computer Engineering, MIT World Peace University, Pune, India, e-mail: prasad.purnaye@mitwpu.edu.in
Vrushali Kulkarni: School of Computer Engineering, MIT World Peace University, Pune, India, e-mail: vrushali.kulkarni@mitwpu.edu.in

experimentation. Section 6 discusses the results from the experiments. The conclusion and future scope of the research are explained in Section 7.

2 Forensics in cloud

According to Gartner, the fastest-growing market will be one of the infrastructure as a service (IaaS) cloud segment, with a projected growth of up to \$38.9 billion by 2022, and the platform as a service (PaaS) market will be reaching around \$31.8 billion by 2022 [1]. The adoption of the cloud is increasing day by day. This fast adoption also attracts criminals to the cloud platform [6]. According to the 2015 Trustwave Global Security Report [7], the volume of attacks on cloud services more than doubled in 2019 and accounted for 20% of the investigated incidents. The IaaS and PaaS adoptions are widely used in the IT industry. There is a high priority need for forensics tools and techniques for these cloud models [8,9], and while we are at it, we need to understand if and why the forensics investigation approach for each of the cloud service models is different.

Each service model of the cloud has a different level of virtualisation implementation [10]. The level of virtualisation specifies which components of the cloud are under the control of the cloud service provider (CSP) and which components are under the control of the cloud user [11,12]. The relative term used to specify this level of control is called the degree of control. In the PaaS model of the cloud, the CSP has a control over and up to the platform level component, which includes networking, storage, servers, operating system, middleware, and runtime environment of the entire stack of cloud servers. However, applications and data related to the applications are managed by the cloud tenant. The degree of control in PaaS is comparatively more than the IaaS and less than the software as a service (SaaS) model of the cloud. We are proposing a solution for the IaaS model based on the considerations of the least degree of control and the most popular adoption in IT industries. However, the proposed system can also serve the need for the SaaS and PaaS models, which are discussed in the future scope section at the end of this article.

In a cloud model, artefacts such as disk snapshots, application logs, access logs, memory dumps, network logs, and network packets can potentially contribute as evidence for the investigation [13–15]. Cyber-attacks in the cloud scenario leave their inkling in some or all of these artefacts. Hence, these artefacts collected at the time of the event are potential evidence. The sources where this evidence is found are virtualized networks, virtualized disks, and virtualized memory. If this evidence data is to be collected and preserved continuously as proposed in refs. [11,16,17], it will take up large amounts of cloud space. Not to mention acquiring some of these artefacts from the VM may demand elevated system privileges [12,18]. To add to these existential problems, sophisticated anti-forensics techniques can obfuscate and corrupt the artefact at sources of evidence.

To mitigate and overcome the aforementioned problems, we propose a system with BiSHM, with a novel evidence detection approach. BiSHM monitors activities linked to sources of evidence. The monitoring is done at the hypervisor level, so for an undesirable event, there are fewer chances of escaping the monitoring. This also mitigates anti-forensics techniques at the VM level. Based on the hypervisor monitoring, only the artefact that is classified as evidence is preserved instead of all the data generated in the cloud. The artefacts to be preserved vary as per the cloud service model because of the degree of control [19].

3 Problem definition

Not all the data which are generated in the cloud are evidence. The data that are generated during an attack or malpractice in the cloud have the potential to contribute as evidence for the investigation. If D_{H} is the data (including logs) generated by the hypervisor and D_{V} is data generated by the virtual machine in a PaaS or IaaS cloud, then the total data generated in the cloud D_{C} can be depicted in equation (1):

$$D_{\mathrm{C}} = D_{\mathrm{H}} \cup D_{\mathrm{V}}. \tag{1}$$

Potential evidence data D_{E} is a subset of D_{C}. Referring to equation (1), we can depict this relation with equation (2):

$$D_{\mathrm{E}} \subseteq (D_{\mathrm{V}} \cup D_{\mathrm{H}}). \tag{2}$$

Evidence detection in traditional cloud forensics is a manual process. Data generated in cloud (D_{E}) are huge. This makes the detection of D_{E} from D_{C} a tedious process. It is like finding a needle in the haystack problem [19]. To differentiate D_{E} from D_{C}, we propose a novel approach to evidence detection and preservation using an AI agent. The proposed AI agent will use a classifier to predict D_{E} given D_{C} using the monitored data generated at the hypervisor level. This classification can be defined in a function depicted in equation (3):

$$f : D_{\mathrm{C}} \rightarrow D_{\mathrm{E}}. \tag{3}$$

The objectives of this research problem are as follows:
- To model an AI agent for evidence detection and preservation in a cloud environment.
- To propose a novel feature set to improve the performance of the proposed AI agent.
- To use a blockchain to preserve the chain of custody of acquired evidence in the cloud.

4 Proposed system

The proposed system consists of a hypervisor monitoring module (HMM), an AI agent, and an evidence provenance module (EPM) as shown in Figure 1. The HMM continuously monitors the memory, disk, and network activities from the virtual device drivers. This is possible with the use of the Libvirt-domain virtualisation Application Programming Library (API) library [20]. Libvirt is a virtualisation library that has drivers for Xen, ESX, Hyper-V, VMware, etc. The AI agent is equipped with a classification module that uses the output from the live monitoring module and classifies the data generated in the cloud as evidence. The evidence acquisition module triggers evidence acquisition and preservation. Acquisition and preservation of the evidence are taken care of by the EPM.

The instrumental key components of our system are the AI Agent, HMM, and EPM. The placement of the cloud is shown in Figure 1. The hypervisor level monitoring is done using two different libraries, namely, OpenNebula and Kernel virtual machine (KVM). KVM provides more detailed activity monitoring than OpenNebula, which is explained in Section 5. These monitoring data that are generated are stored in the monitoring dataset. We propose a novel feature set, which is derived from the existing features of KVM and the OpenNebula dataset. The monitoring data are used for training the AI agent for evidence detection. Our novel feature-set acts as sensor data for the AI agent. The AI agent is modelled for evidence data detection. The trained classification model enables the AI agent to trigger the evidence acquisition using Libvirt libraries for evidence acquisition. The evidence is pushed to the provenance module where it is securely stored using a simple blockchain mechanism. Thus, continuous monitoring of the data yields selective evidence detection and acquisition.

4.1 Hypervisor monitoring module

The HMM is the core component of the framework which monitors the hypervisor activities and profiles the virtualized resources of the cloud. The HMM module monitors each VM. There are two submodules of the HMM. Monitoring is done using OpenNebula and KVM. In a private cloud deployment,

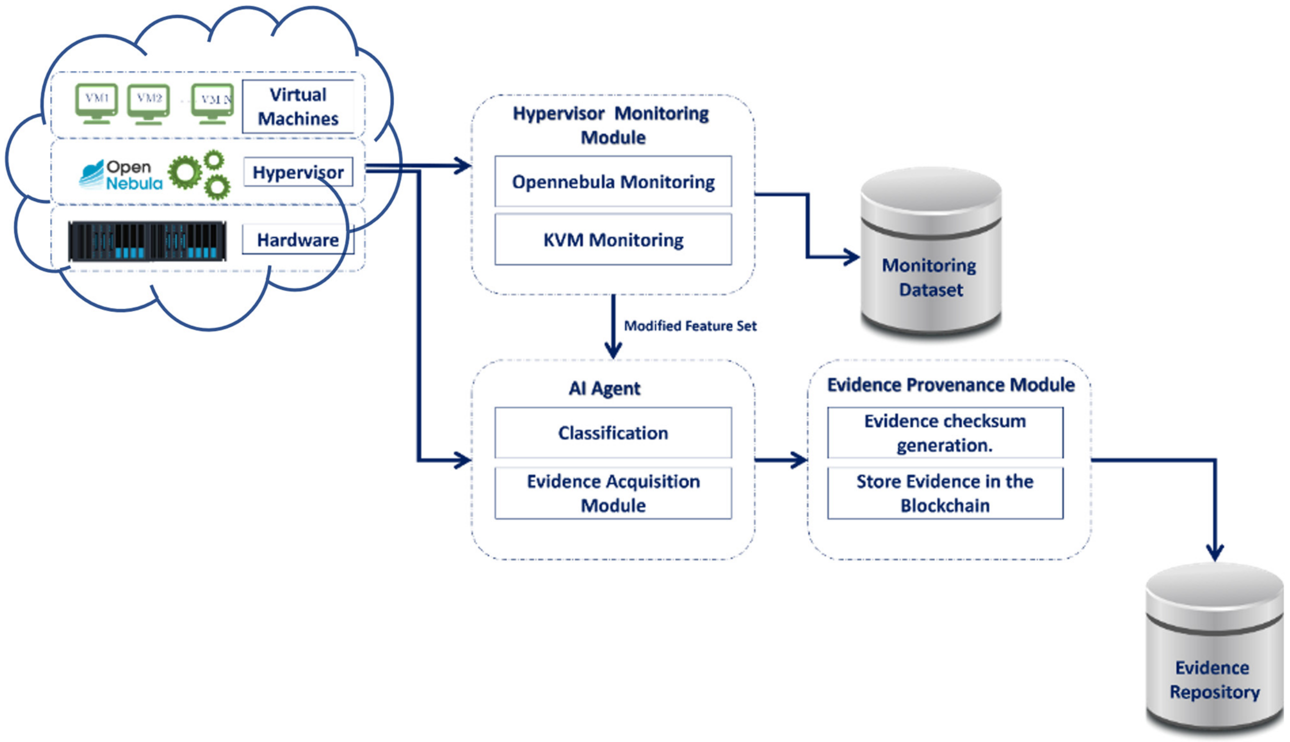


Figure 1: Proposed system.

VM monitoring is done at the KVM hypervisor level using the Libvirt API. Libvirt supports KVM and Xen virtualisation. The API provides diverse parameter monitoring of a VM. The pseudocode of the monitoring using OpenNebula and KVM is shown in Algorithms 1 and 2, respectively. The parameters and their descriptions, which are collected as a dataset, are given in Section 4.2.

Algorithm 1: Monitoring using OpenNebula

```
Result: Get Monitoring information
conn=Connect_to_oned_daemon(one_auth);
conn.GetActiveVMsList XML file;
for <for each VM in the XML> do
    if V is live then
        getXMLInformationFile();
        parseMemoryInfo();
        parseDiskInfo();
        parseNetworkInfo();
        StoreInMySQL(OpennebulaMonitoring);
    else
        SkipMonitoring();
    end
end
```

Algorithm 2: KVM_Monitoring

```
Result: MonitoringDatabase
Establish connection to QEMU_Hypervisor();
for VM deployed on the cloud do
    find devices of VM;
    get_LAST_POLL_value();
    get_interfaces_stats();
    get_CPU_stats();
    get_memory_stats();
    get_blocks_stats(hda);
    get_blocks_stats(vda);
    store_data_mysql();
end
```

4.2 Model of the proposed AI agent

In designing an agent, the first step must be always to specify the task environment as fully as possible [21]. We have modelled the AI agent by specifying the performance measure, the environment, and the agent's actuators and sensors (PEAS). The precise definition of the task environment is given in Sections 4.2.1–4.2.4 by the PEAS description.

4.2.1 Performance

Response time and evidence acquisition time are the performance parameters of our proposed agent. The agent has two submodules, namely, classification and evidence acquisition. The classification module identifies evidence data from the generated cloud data. Response time of the identification of evidence depends on the detection of an attack in the cloud environment [22]. The detection of evidence data triggers the acquisition module. The evidence acquisition module acquires the data and stores it as evidence. The performance of evidence acquisition is technology dependent. The size of the evidence and the specifications of the hardware components limit the performance of the evidence acquisition process.

4.2.2 Environment

Environment profiling of the agent is documented in this section. The agent's sensors are deployed at the hypervisor level. The agent environment is fully observable. It is a single-agent model where only a single agent is in the action. The state of the environment is categorized as deterministic. The classification agent experience is atomic; hence, the environment state is episodic. The unchanging state of the environment makes it static. The parameters of the environment, which are being monitored continuously, have a continuous distinction that is applied to the state of the environment making it a continuous environment. The environment is known for the outcomes of all actions.

4.2.3 Actuators

If evidence is identified by the agent, it is stored in the evidence repository. This is done by using triggers to fetch the volatile data from the hypervisor level API. The actuators for the proposed AI agent are the evidence detection and evidence acquisition submodules.

4.2.4 Sensors

The activities at the sources of evidence are used for the placement of the sensors. These activities can be quantified using the HMM. Hypervisor level APIs are used by the sensors to collect the monitoring data about the resources. The collected data are used to predict D_E from the main sources of the virtualized resources such as a virtualized disk, network, and memory.

4.3 Classification module

The classification module of an AI agent is trained on the generated training dataset. Machine learning modules have a binary classification agent. The agent takes the monitoring data Mt at every time instance t from the HMM and uses it to classify t to be an attack scenario or a normal scenario. This classification triggers the evidence tagging module. The evidence tagging module starts to acquire, tag, and store the data (D_C) generated at the t time instance as evidence data (D_E). We have explored the following models for the evidence classification problem:

- k-Nearest neighbour (kNN)
- Support vector machine (SVM)
- Naive Bayes
- Tree
- Random forest

4.3.1 kNN

Euclidean distance and Mahalanobis distance are commonly used distance metrics. The most basic instance-based method is the kNN algorithm [23]. This algorithm assumes that all instances correspond to points in the n-dimensional space. The nearest neighbours of an instance are defined in terms of the standard Euclidean distance. More precisely, let an arbitrary instance x be described by the feature vector described by equation (4):

$$\{a_1\,(x),\,a_2\,(x),....,a_n(x)\} \tag{4}$$

where $a_1(x)$ denotes the value of the rth attribute of instance x. Then, the distance between the two instances x_i and x_j is defined to be $d(x_i, x_j)$, as shown in equation (5):

$$d\,(x_i, x_j) = \sqrt{\sum_{r=0}^{n} (a_r\,(x_i) - a_r\,(x_i))^2}\,, \tag{5}$$

Unlike the Euclidean distance though, the Mahalanobis distance accounts for how correlated the variables are to one another. The Mahalanobis distance calculation differs only slightly from Euclidean distance as shown in equation (6):

$$d(\vec{x}, \vec{y}) = \sqrt{(\vec{x} - \vec{y})^T s^{-1}(\vec{x} - \vec{y})}\,. \tag{6}$$

where $\vec{x}$, $\vec{y}$ represent attributes, and s^{-1} is the covariance matrix of $\vec{x}$ and $\vec{y}$. Evaluation results indicate that the performance of the Mahalanobis distance is better compared to the Euclidean distance.

4.3.2 SVM

SVMs are particular linear classifiers that are based on the margin maximisation principle. They perform structural risk minimization, which improves the complexity of the classifier intending to achieve excellent generalization performance [24]. The SVM accomplishes the classification task by constructing, in a higher-dimensional space, the hyperplane that optimally separates the data into two categories. SVM does not deliver as good results for our problem data as other algorithms. The reason could be that the size of the dataset and the dimensions are larger.

4.3.3 Naïve Bayes

Naïve Bayes is a simple learning algorithm that utilizes the Bayes rule together with a strong assumption that the attributes are conditionally independent, given the class. This classifier refers to a group of probabilistic classifiers. It implements the Bayes theorem for classification problems. The first step of the Naive Bayes classifier is to determine the total number of classes (outputs) and calculate the conditional probability for each dataset class. Then, the conditional probability is calculated for each attribute.

4.3.4 Decision tree

A decision tree refers to both a concrete decision model used to support decision-making and a method to construct such models automatically from data. As a model, a decision tree refers to concrete information or a knowledge structure to support decision-making [25]. Selection of the best attribute for the root node and subnodes is crucial while implementing a decision tree. Attribute selection measure (ASM) is used to select the nodes and subnodes. We have used the information gain technique for ASM to select attributes, and the results are explained in Section 6.

4.3.5 Random forest

A random forest is an ensemble of random decision tree classifiers, which makes predictions by combining predictions of individual trees [26]. There are different approaches to introduce randomness in the decision tree construction method. A random forest can be used to make predictions over nominal (classification) or numeric target attributes (regression). Random forests are one of the best performing predictive models. For our datasets, the random forest technique has shown improvement in the sensitivity evaluation metrics with the help of a combinational approach of random forest.

4.4 Evidence provenance module

Memory is stored in a secure folder with limited access policies using the Libvirt memory dump API using the raw format [20]. The file location is also recorded against the memory dump. The algorithm for evidence provenance using blockchain is explained in Algorithm 3.

Algorithm 3: BlockChain of EvidenceRepo

```
Result: Evidence Provenance
OpenConnectionwithMYSQL;
for <for each evidence in the EvidenceRepo> do
    prev_data=previous_data();
    calculateHash(curr_data);
    EncryptUsingSHA256(prev_data, Data, Timestamp);
    AppendBlockToList();
end
```

4.4.1 Evidence acquisition module

Electronic evidence can be found in a disk, network, and memory. However, in a cloud environment, evidence data in memory are highly volatile in nature and difficult to acquire [19]. Volatile memory in most cases no longer exists if the VM is shut down or undeployed [8,27]. But volatile data are very crucial in forensics for crime scene reconstruction. Important artefacts such as process id, password, and unencrypted data can be found in the volatile memory. However, fetching volatile memory data are difficult mainly because of the following reasons: (i) The software approach of fetching volatile memory needs a program to run in the memory affecting the original contents of the memory. (ii) Volatile memory acquisition requires live forensics. Memory being the critical information source and highly volatile, we have discussed the implementation of evidence collection with memory dumps acquisition module. Nonetheless, disk and network data acquisition modules can also work smoothly with the proposed system.

The evidence detection agent of the proposed system triggers the evidence acquisition module. The memory acquisition module uses a kernel-level API to acquire and dump the memory data associated with the VM; hence, it does not require any agent installed on the VM. When the BiSHM agent is running at the hypervisor level, it can trigger the acquisition module to take memory dumps of the VM at a specific time instance *t*. This dumped data are preserved as evidence in the evidence repository. The evidence acquisition module of the proposed system uses the Libvirt API [20] to acquire the dump. The dump is then processed for evidence provenance. The algorithm for memory evidence acquisition is shown in Algorithm 4.

Algorithm 4: Acquisition of MemoryDumps with KVM

```
Result: Acquire Memory Evidence
Open Connection to QEMU System through Libvirt;
List Active Domains using Libvirt;
for <for each domain dom in the list> do
    if V is deployed then
        SaveMemoryContentstoFile();
        GenerateSHA256hash();
        CompressTheFileUsingGZIP();
        RecordFileLocation();
        RecordFileCreationTime();
        RecordFileModificationTime();
        RecordFileAccessTime();
        StoreAllInMySQL();
    else
        SkipMonitoring();
    end
end
```

4.4.2 Blockchain

Blockchain is a tamper-resistant timestamp-based distributed ledger that often adapts for sharing and storing

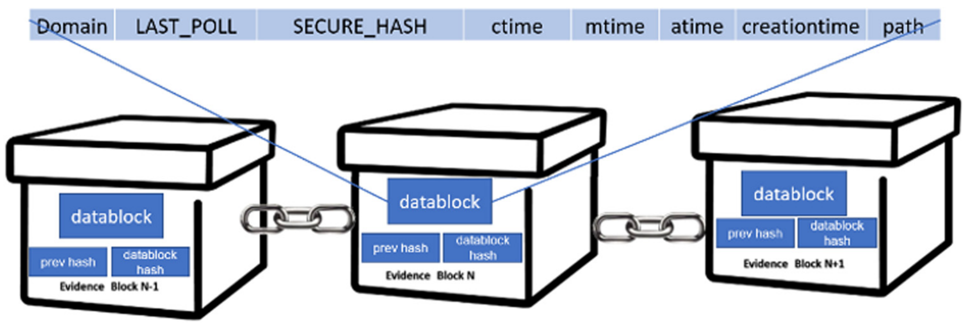
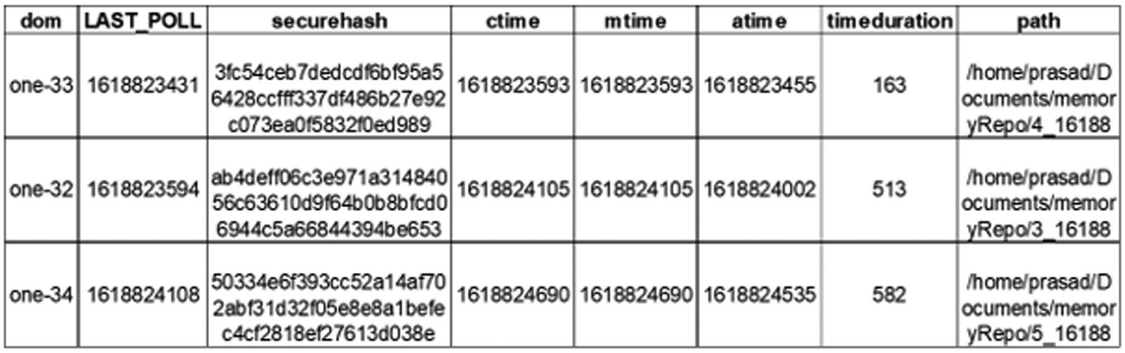
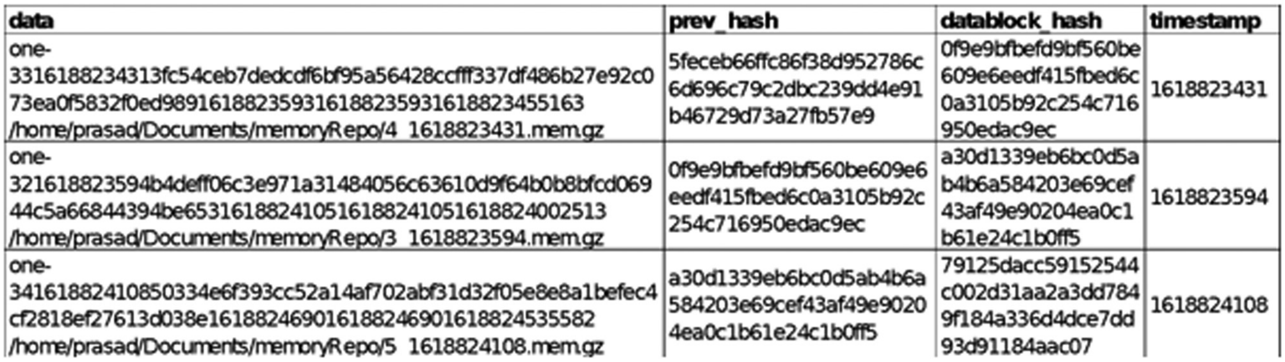


Figure 2: Blockchain model for evidence repository.



dom	LAST_POLL	securehash	ctime	mtime	atime	timeduration	path
one-33	1618823431	3fc54ceb7dedcdf6bf95a5 6428ccfff337df486b27e92 c073ea0f5832f0ed989	1618823593	1618823593	1618823455	163	/home/prasad/D ocuments/memor yRepo/4_16188
one-32	1618823594	ab4deff06c3e971a314840 56c63610d9f64b0b8bfcd0 6944c5a66844394be653	1618824105	1618824105	1618824002	513	/home/prasad/D ocuments/memor yRepo/3_16188
one-34	1618824108	50334e6f393cc52a14af70 2abf31d32f05e8e8a1befe c4cf2818ef27613d038e	1618824690	1618824690	1618824535	582	/home/prasad/D ocuments/memor yRepo/5_16188

Figure 3: Snapshot of the data to be stored in the blockchain.



data	prev_hash	datablock_hash	timestamp
one- 3316188234313fc54ceb7dedcdf6bf95a56428ccfff337df486b27e92c0 73ea0f5832f0ed989161882359316188235931618823455163 /home/prasad/Documents/memoryRepo/4 1618823431.mem.gz	5feceb66ffc86f38d952786c 6d696c79c2dbc239dd4e91 b46729d73a27fb57e9	0f9e9bfbefd9bf560be 609e6eedf415fbed6c 0a3105b92c254c716 950edac9ec	1618823431
one- 321618823594b4deff06c3e971a31484056c63610d9f64b0b8bfcd069 44c5a66844394be653161882410516188241051618824002513 /home/prasad/Documents/memoryRepo/3 1618823594.mem.gz	0f9e9bfbefd9bf560be609e6 eedf415fbed6c0a3105b92c 254c716950edac9ec	a30d1339eb6bc0d5a b4b6a584203e69cef 43af49e90204ea0c1 b61e24c1b0ff5	1618823594
one- 34161882410850334e6f393cc52a14af702abf31d32f05e8e8a1befec4 cf2818ef27613d038e161882469016188246901618824535582 /home/prasad/Documents/memoryRepo/5 1618824108.mem.gz	a30d1339eb6bc0d5ab4b6a 584203e69cef43af49e9020 4ea0c1b61e24c1b0ff5	79125dacc59152544 c002d31aa2a3dd784 9f184a336d4dce7dd 93d91184aac07	1618824108

Figure 4: Blockchain database snapshot.

data. The fundamental elements of the blockchain are listed as follows:

- Decentralization: In blockchain architecture, control is not retained by a centralized authority. Instead, it is distributed among peers. In the chain, each node is free to join or leave the blockchain network [28].
- Collective verification: All other nodes verify the transactions made on the blockchain. In addition, it also provides tamper assistance, which enables the chain of custody of evidence. The data that are stored in the blockchain cannot be modified or deleted.
- Security and integrity: All transactions are verified and stored in blocks under strong cryptographic functions. The involvement of digital signatures preserves data security and integrity [29]. Figure 2 depicts a blockchain model for evidence repository [28].

Table 1: Category and description of features of OpenNebula dataset

Sr No	Category	Parameter	Description
1	Meta-data	VMID	The ID of the VM
2		LAST_POLL	EPOCH timestamp
3		STATE	State of the VM (explained below)
4		MAC	The physical address of the VM
5		IP	IPv4 addresses associated with the VM at LAST_POLL time instant
6	Memory	CPU	Percentage of CPU consumed
7		MEMORY	Memory consumption in kilobytes
8	Network	NETRX	Received bytes from the network
9		NETTX	Sent bytes to the network
10	Disk	DISKRDBYTES	Read bytes from all disks since last VM start
11		DISKWRBYTES	Written bytes from all disks since last VM start
12		DISKRDIOPS	Read IO operations from all disks since the last VM start
13		DISKWRIOPS	Written IO operations all disks since the last VM start

The file size of the evidence file for a VM is as huge as the memory size of the VM, which is several Giga Bytes. To store such a huge file in the blockchain would require calculating the hash of these large files. To tackle this problem, the EPM compresses the evidence and stores it at an encrypted location. Before compressing the memory file, it also calculates the hash value of the file using SHA-256. The metadata of the memory dump including the access timestamp (atime), the modified timestamp (mtime), and the changed timestamp (ctime) and path are stored in the repository dataset. A snapshot of the generated data is shown in Figure 3. These data are then stored in a blockchain. A snapshot of the saved data is shown in Figure 4.

Table 2: Category and description of features of KVM dataset

Sr No	Category	Feature	Description
1	Meta-data	LAST_POLL	Epoch timestamp
2		VMID	The ID of the VM
3		UUID	Unique identifier of the domain
4		Dom	Domain name
5	Network	Rxbytes	Received bytes from the network
6		rxpackets	Received packets from the network
7		rxerrors	Number of received errors from the network
8		rxdrops	Number of received packets dropped from the network
9		txbytes	Transmitted bytes from the network
10		txpackets	Transmitted packets from the network
11		txerrors	Number of transmission errors from the network
12		txdrops	Number of transmitted packets dropped from the network
13	Memory	timecpu	Time spent by vCPU threads executing guest code
14		timesys	Time spent in kernel space
15		timeusr	Time spent in userspace
16		state	Running state
17		memmax	Maximum memory in kilobytes
18		mem	Memory used in kilobytes
19		cpus	Number of virtual CPUs
20		cputime	CPU time used in nanoseconds
21		memactual	Current balloon value (in KiB)
22		memswap_in	The amount of data read from swap space (in KiB)
23		memswap_out	The amount of memory written out to swap space (in KiB)
24		memmajor_fault	The number of page faults where disk IO was required
25		memminor_fault	The number of other page faults
26		memunused	The amount of memory left unused by the system (in KiB)
27		memavailable	The amount of usable memory as seen by the domain (in KiB)
28		memusable	The amount of memory that can be reclaimed by balloon without causing host swapping (in KiB)
29		memlast_update	The timestamp of the last update of statistics (in seconds)
30		memdisk_cache	The amount of memory that can be reclaimed without additional I/O, typically disk caches (in KiB)
31		memhugetlb_pgalloc	The number of successful huge page allocations initiated from within the domain
32		memhugetlb_pgfail	The number of failed huge page allocations initiated from within the domain
33		memrss	Resident set size of the running domain's process (in KiB)
34	Disk	vdard_req	Number of read-requests on the vda block device
35		vdard_bytes	Number of read-bytes on the vda block device
36		vdawr_reqs	Number of write requests on the vda block device
37		vdawr_bytes	Number of write requests on vda the block device
38		vdaerror	Number of errors in the vda block device
39		hdard_req	Number of read requests on the hda block device
40		hdard_bytes	Number of read bytes on the had block device
41		hdawr_reqs	Number of write requests on the hda block device
42		hdawr_bytes	Number of write bytes on the hda block device
43		hdaerror	Number of errors in the hda block device

5 Datasets

We have deployed a private cloud setup using OpenNebula [30]. There are two ways of monitoring resources using OpenNebula. One of the ways is to use Abstract APIs provided by OpenNebula to monitor the VM. The second way is to use the Libvirt API to monitor the virtualized domains. One of the datasets is generated using OpenNebula Monitoring API and the other dataset using Libvirt API. We have considered both datasets for the experimentation. The OpenNebula dataset has fewer parameters as compared to the Libvirt dataset. The list and the description of the parameters are provided in Tables 1 and 2 for OpenNebula and KVM, respectively. The OpenNebula API provides seven parameters, whereas the KVM parameter provides 43 parameters. These parameters are associated with the disk, network, and memory attributes of the VM. The datasets are stored in the MYSQL database at a regular time interval. The time interval of OpenNebula is greater than the time interval of KVM. Both the datasets showed good classification accuracy. However, for the final evaluation, we have considered the KVM dataset as it gives more detailed features of the disk, memory, and network. We have published both datasets on IEEE Dataport [31,32].

6 Experimentation

6.1 Setup

The author did not find any suitable and feasible existing dataset, which has the parametric data of a cloud platform along with a mechanism to store the evidence data. So, to generate a synthetic dataset, a private cloud was set up for the proof of concept. The system configuration included Intel® Core™ i5-4590 Processor with 12 GB of RAM with 1 TB of HDD. The private cloud setup was done using a KVM type-1 hypervisor along with OpenNebula (version 5.12) as the cloud management platform. To simulate a real-time cloud environment a script generating, a synthetic workload was deployed on the virtual machines of the cloud.

Table 3: Information gain ranking of features from OpenNebula dataset

Sr No	Parameter	Information gain
1	NETTX	0.545711414
2	DISKWRBYTES	0.208363871
3	DISKWRIOPS	0.205281549
4	NETRX	0.165385768
5	CPU	0.020941001
6	DISKRDBYTES	0.004170941
7	DISKRDIOPS	0.003862696

Table 4: Information gain ranking of features from KVM dataset

Sr No	Parameter	Information gain
1	txpackets	0.623739395
2	txbytes	0.507514798
3	rxpackets	0.195322086
4	rxbytes	0.193009701
5	vdawr_reqs	0.186630402
6	timesys	0.163283136
7	vdawr_bytes	0.167101488
8	timecpu	0.095914082
9	cputime	0.095914082
10	memlast_update	0.000636577
11	timeusr	0.056836573
12	hdard_req	0.00047371
13	hdard_bytes	0.00047371
14	memminor_fault	0.000294259
15	state	0.002081396
16	memunused	0.000247844
17	memusable	0.000247844
18	memrss	0.01690513
19	vdard_bytes	0.001638032
20	vdard_req	0.001422896

Table 5: Pearson correlation scores for parameter pairs from the OpenNebula dataset

Sr No	Pearson correlation	Parameter pair	
1	0.99	DISKRDBYTES	DISKRDIOPS
2	0.969	DISKRDBYTES	DISKWRIOPS
3	0.944	DISKRDIOPS	DISKWRIOPS
4	0.94	DISKRDIOPS	NETRX
5	0.937	DISKRDBYTES	NETRX
6	0.878	DISKWRIOPS	NETRX
7	0.778	DISKWRBYTES	NETRX
8	0.722	DISKRDIOPS	DISKWRBYTES
9	0.678	DISKRDBYTES	DISKWRBYTES
10	0.574	DISKWRBYTES	DISKWRIOPS
11	0.31	NETRX	NETTX
12	0.251	DISKRDIOPS	NETTX
13	0.211	DISKRDBYTES	NETTX
14	0.199	DISKWRIOPS	NETTX
15	0.186	CPU	DISKWRIOPS
16	0.179	CPU	DISKRDBYTES
17	0.171	CPU	DISKRDIOPS
18	0.129	CPU	NETRX
19	0.126	DISKWRBYTES	NETTX
20	0.046	CPU	DISKWRBYTES
21	0.046	CPU	NETTX

Table 6: Pearson correlation scores for parameter pairs from the KVM dataset

Sr No	Pearson correlation	Parameter pair	
1	0.998	hdard_bytes	hdard_req
2	0.992	rxbytes	rxpackets
3	0.789	vdard_bytes	vdard_req
4	0.697	vdawr_bytes	vdawr_reqs
5	0.654	timeusr	txpackets
6	0.593	hdard_bytes	vdard_req
7	0.579	hdard_req	vdard_req
8	0.575	txbytes	txpackets
9	0.518	timesys	txpackets
10	0.376	timeusr	txbytes
11	0.306	timesys	txbytes
12	0.303	timesys	timeusr
13	0.255	hdard_bytes	vdard_bytes
14	0.248	hdard_req	vdard_bytes
15	0.166	vdard_req	vdawr_reqs
16	−0.125	txpackets	vdawr_reqs
17	0.122	vdard_bytes	vdawr_bytes
18	0.119	vdard_bytes	vdawr_reqs
19	0.104	vdard_req	vdawr_bytes
20	0.088	rxpackets	vdawr_bytes
21	0.085	timecpu	txpackets
22	−0.075	timeusr	vdawr_reqs
23	−0.074	txbytes	vdawr_reqs
24	0.061	timecpu	txbytes
25	0.055	rxpackets	vdawr_reqs
26	0.054	rxbytes	vdawr_bytes
27	−0.052	timesys	vdawr_reqs
28	−0.051	txpackets	vdawr_bytes
29	0.048	rxbytes	vdawr_reqs
30	0.039	timecpu	timesys
31	−0.037	timecpu	vdawr_bytes
32	−0.032	rxpackets	txpackets
33	−0.03	timeusr	vdawr_bytes
34	−0.029	txbytes	vdawr_bytes
35	0.025	timecpu	timeusr
36	−0.019	rxpackets	txbytes
37	−0.018	timesys	vdawr_bytes
38	−0.018	txpackets	vdard_req
39	−0.018	rxpackets	timeusr
40	−0.017	txpackets	vdard_bytes
41	−0.017	rxbytes	txpackets
42	−0.015	rxpackets	timesys
43	−0.013	rxpackets	timecpu
44	0.011	timecpu	vdawr_reqs
45	−0.01	txbytes	vdard_req
46	−0.01	rxbytes	txbytes
47	−0.01	txbytes	vdard_bytes
48	0.009	rxpackets	vdard_bytes
49	0.009	rxbytes	vdard_bytes
50	−0.008	hdard_req	txpackets
51	-0.008	hdard_bytes	txpackets
52	−0.008	rxbytes	timeusr
53	0.008	rxpackets	vdard_req
54	−0.007	rxbytes	timesys
55	0.007	rxbytes	vdard_req
56	0.007	hdard_req	vdawr_bytes
57	0.006	hdard_bytes	vdawr_bytes
58	−0.006	timesys	vdard_bytes
59	0.006	rxbytes	timecpu
60	0.006	hdard_req	vdawr_reqs
61	−0.006	timeusr	vdard_bytes
62	0.005	hdard_bytes	vdawr_reqs
63	−0.005	timesys	vdard_req
64	−0.005	hdard_req	txbytes
65	−0.005	hdard_bytes	txbytes
66	0.004	timecpu	vdard_bytes
67	−0.003	hdard_bytes	timecpu
68	−0.003	hdard_req	timecpu
69	−0.002	hdard_req	rxpackets
70	0.002	timecpu	vdard_req
71	0.002	hdard_bytes	timeusr
72	-0.002	hdard_bytes	rxpackets
73	−0.002	timeusr	vdard_req
74	0.002	hdard_req	timeusr
75	−0.001	hdard_req	rxbytes
76	−0.001	hdard_bytes	rxbytes

Table 6: *Continued*

6.2 Proposed modified feature set

Both the generated datasets, OpenNebula and KVM, did not have any missing values. The numeric features of the dataset represent respective cumulative values till time *t*. To obtain the rate of the activities at virtualized resources at a specific time instance *t*, the difference of consecutive data values should be considered. Numeric features representing disk and network activities are not used as they are received from the HMM. The calculation steps of the proposed new feature set are as represented in equation (7):

$$F_{\text{new}} = \frac{\Delta(F_{\text{old}})}{\Delta(\text{LAST\_POLL})/60}, \tag{7}$$

where $\Delta(F_{\text{old}})$ represents the difference of the consecutive values of the feature and $\Delta(\text{LAST\_POLL})$ represents the time duration of these consecutive F values.

6.3 Pre-processing

Normalization of each of the feature parameters is done to mitigate any dominant effect of any specific parameter. Normalization adjusts values to a common scale by using equation (8).

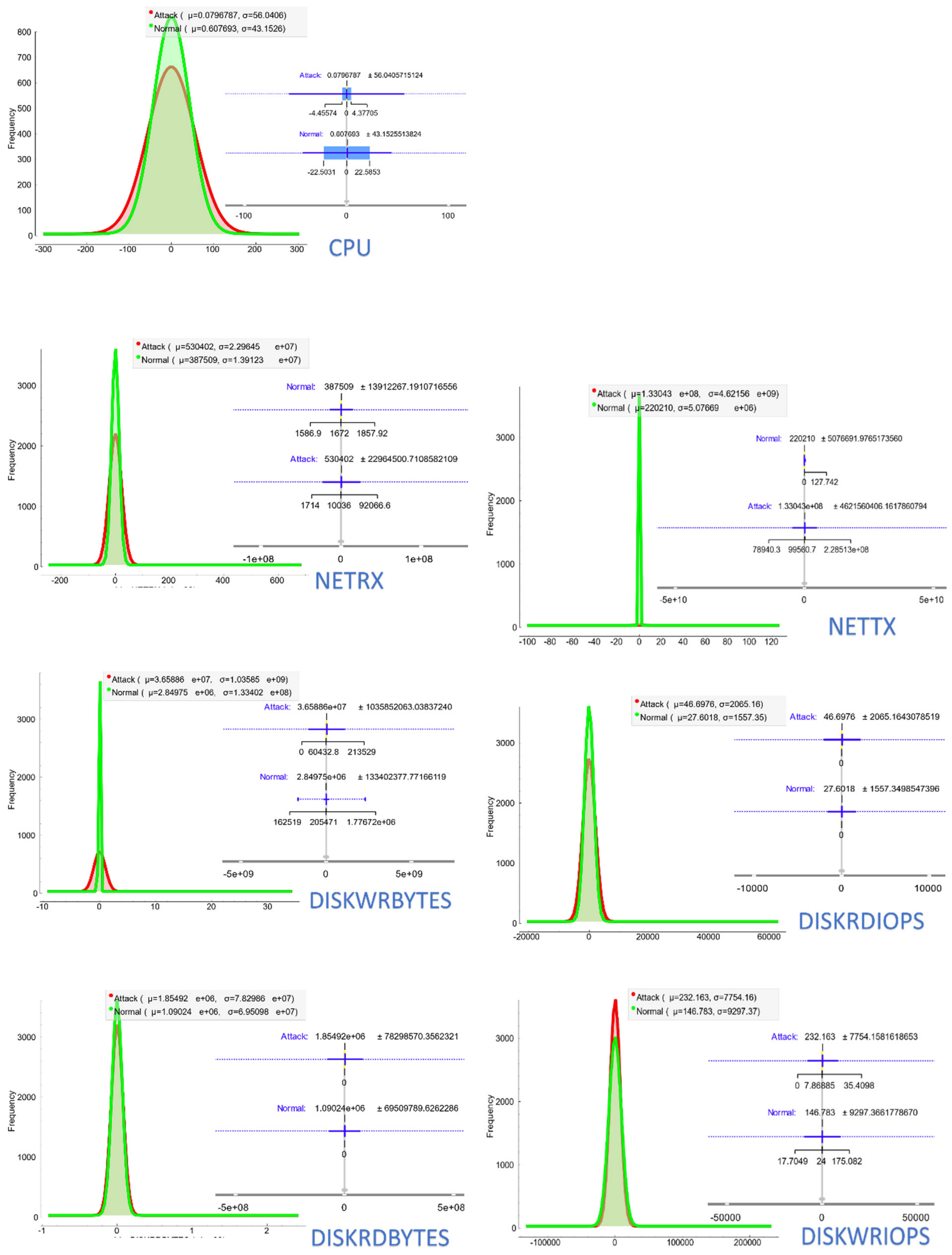


Figure 5: Distribution and box plot of OpenNebula dataset.

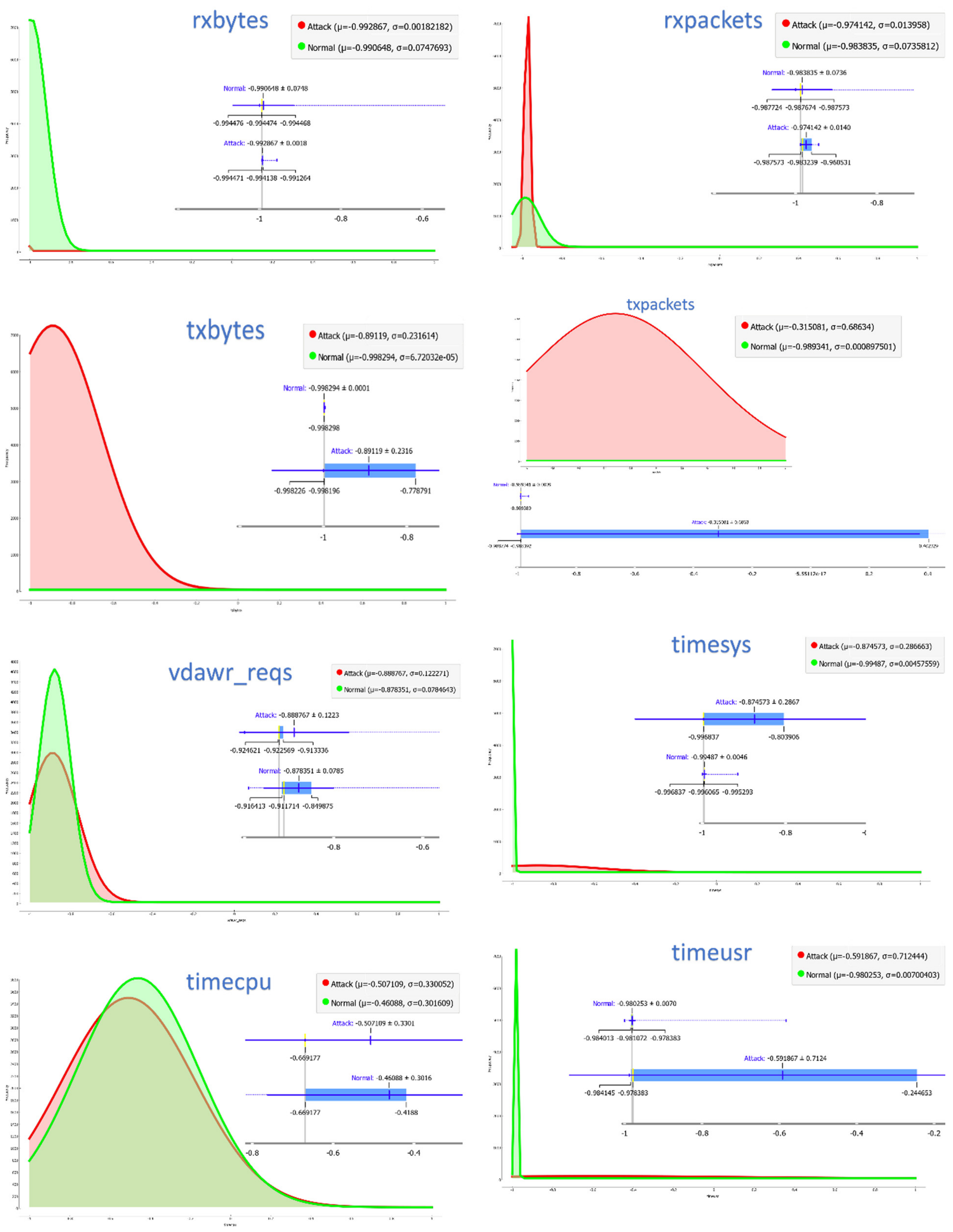


Figure 6: Distribution and box plot of KVM dataset.

Table 7: Parameter setting for classification models

Model	Parameter setting for KVM dataset	Parameter setting for OpenNebula dataset
Random forest	Train:test split = 10%:90%, no of features = 8, no of trees = 8, growth control = do not split subset smaller than 5	Train:test split = 10%:90%, no of features = 7, no of trees = 7, growth control = do not split subset smaller than 5
kNN	Train:test split = 10%:90%, no of features = 8, no of neighbours = 5, metric = uclidean, weight = uniform	Train:test split = 10%:90%, no of features = 7, no of neighbours = 5, metric = uclidean, weight = uniform
SVM	Train:test split = 10%:90%, no of features = 8, SVM type: SVM, $C = 1.0$, $\varepsilon = 0.1$, Kernel: RBF, $\exp(-0.35\|x-y\|^2)$, Numerical tolerance: 0.001, Iteration limit: unlimited	Train:test split = 10%:90%, no of features = 7, SVM type: SVM, $C = 1.0$, $\varepsilon = 0.1$, Kernel: RBF, $\exp(-0.35\|x-y\|^2)$, Numerical tolerance: 0.001, Iteration limit: unlimited

$$F_{\text{normalized}} = \frac{F - F_{\min}}{F_{\max} - F_{\min}}. \tag{8}$$

6.4 Feature ranking

The ranking experiment is carried out on both datasets using information gain [33]. Given a data set X with n cases, $X = \{x_1, x_2, ..., x_n\}$, where each case x_i belongs to exactly one of the k predefined classes $c_j \in \{c_1, c_2, ..., c_k\}$. Then, the information gain of attribute a is the reduction in entropy caused by partitioning X based on the values v_i of that attribute a; it is given by equation (9):

$$\begin{aligned}&\text{Information gain}\,(X,\ a)\\ &= \text{Entropy}\,(X) - \sum_{v\,\in\,\text{values}\,(a)}^{n} \frac{(a = v)}{n} \times \text{entropy}\,(X,\ a = v).\end{aligned} \tag{9}$$

Tables 3 and 4 present the information gain of different parameters of OpenNebula and KVM datasets, respectively, in descending order of the information gain.

6.5 Feature selection

Correlation is a bivariate analysis that measures the strength of association between two variables and the direction of the relationship. In terms of the strength of the relationship, the value of the correlation coefficient varies between +1 and –1 [34]. Pearson r correlation is the most widely used correlation statistic to measure the degree of the relationship between linearly related variables. So, we have experimented with both datasets to find the top parameters for effective machine learning performance. Tables 5 and 6 present the correlation score for the OpenNebula and KVM datasets.

From the results presented in Table 5, we have selected seven parameters for further experiments using the OpenNebula dataset, and referring to Table 6, we have considered eight parameters for the KVM dataset. This parameter selection is based on high-decorrelated features with high information gain. The distribution of these features is explained in Section 6.6.

6.6 Distributions

The simplest form of Gaussian distribution is the one-dimensional standard Gaussian distribution [25]. All the selected parameters of the dataset show Gaussian distribution. Box plot is another way to discover any anomalies in the dataset. Figures 5 and 6 represent the distribution and box plot of the top features that were selected from the earlier stage. The distribution happens to be a

Table 8: Classification evaluation results for OpenNebula dataset

Model	AUC	CA	F1	Precision	Recall
Random forest	0.984	0.987	0.987	0.987	0.987
Tree	0.942	0.975	0.974	0.975	0.975
kNN (Mahalanobis distance)	0.931	0.921	0.922	0.925	0.921
Naive Bayes	0.974	0.914	0.915	0.917	0.914
SVM	0.74	0.874	0.855	0.891	0.874
Logistic regression (Lasso)	0.498	0.764	0.661	0.583	0.764

Table 9: Classification evaluation results for KVM dataset

Model	AUC	CA	F1	Precision	Recall
Random forest	0.998	0.998	0.996	0.998	0.994
Logistic regression (Ridge)	0.959	0.889	0.701	0.995	0.541
SVM	0.997	0.949	0.883	0.995	0.793
Logistic regression (Lasso)	0.995	0.992	0.983	0.995	0.971
kNN (Mahalanobis distance)	0.996	0.995	0.991	0.993	0.988

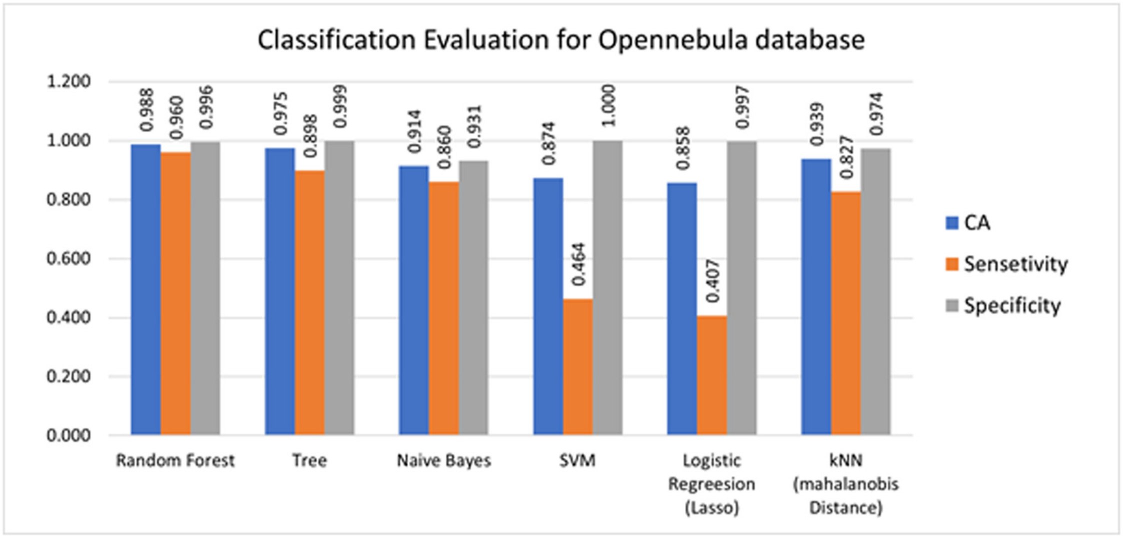


Figure 7: Classification evaluation for KVM dataset.

Gaussian distribution. Figure 5 shows the results for the OpenNebula dataset, and Figure 6 shows the results for the KVM dataset.

6.7 Evaluation results

The OpenNebula dataset [31] and KVM dataset [32] have a total of 4869 and 9610 records, respectively. By using 10% of the fixed sampling method, we have trained SVM, Naïve Bayes, Tree, and Random forest models. The parameter setting, which gives better performance, is selected through extensive empirical experimentation as shown in Table 7. The focus of the research is not to modify any of these classification models. However, the proposed modified feature set used to train the models has revealed better classification results ultimately improving the performance of the AI agent in the proposed system.

The proposed system is tested with training data sampling of 10, 20, 30, 40, 50, 60, and 70%. The proposed agent was trained using SVM, Naïve Bayes, Tree, and random forest models out of which the random forest model gives better classification accuracy in almost every

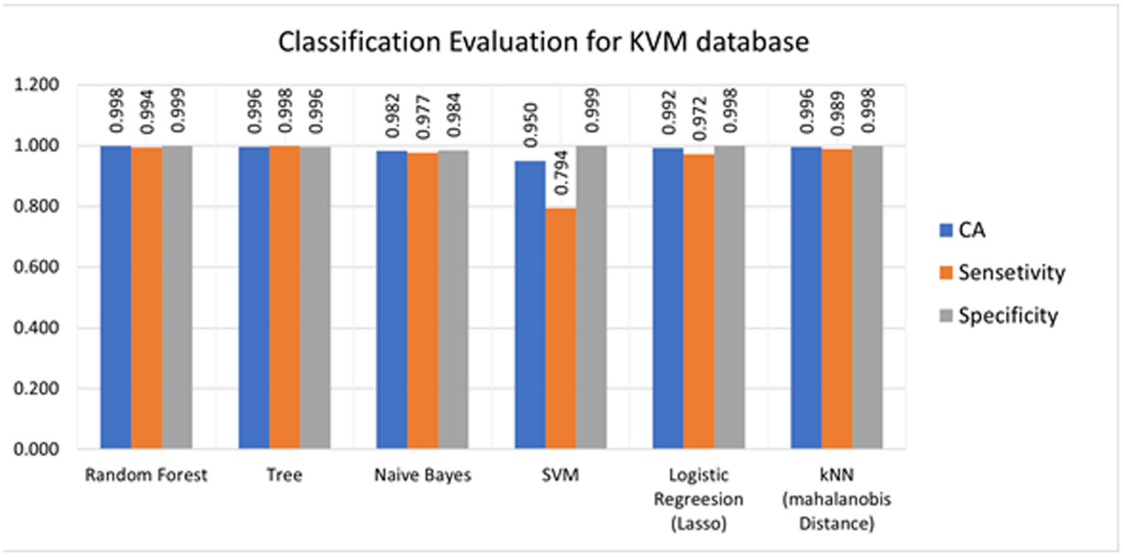


Figure 8: Classification evaluation for KVM dataset.

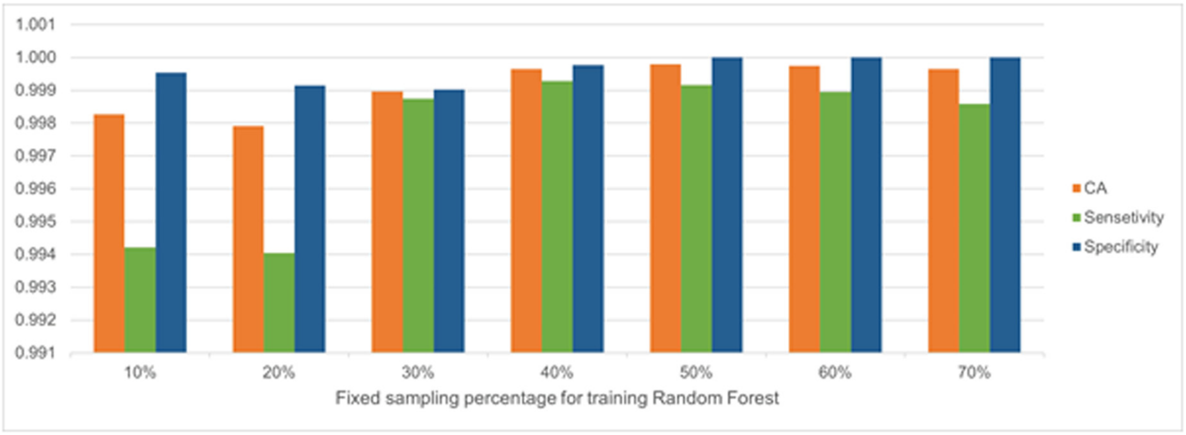


Figure 9: Graph showing the comparison of evaluation parameters for classification using KVM dataset.

training iteration from 10% to 70%. The data exhibit a class imbalance problem. However, without using any explicit control over the data balance ratio, the Naïve Bayes provides better overall classification accuracy for both the OpenNebula and KVM datasets. Because of the class imbalance problem, we have considered sensitivity and specificity to measure the performance through the evaluation results. Table 8 presents the classification evaluation results for the OpenNebula dataset, and Table 9 shows the classification evaluation results for the KVM dataset. Figures 7 and 8 show the respective graphical depiction of the results for OpenNebula and KVM.

Extensive experimentation shows that the random forest classifier gives the best classification accuracy, sensitivity, and specificity combination altogether. Random forest handles the imbalance of the dataset well. Upon increasing the training data sample size, the AI agent shows improvement in the performance parameters as shown in Figure 9.

6.8 Storage efficiency

The evidence repository module saves only the evidence data which is generated during the attack. This reduces the space required to preserve volatile evidence. For the experiment we carried out, we implemented the volatile data dump module which generated around 360 VM memory dump images with an average size of 1 GB each. These data files are compressed to a 79.5 GB memory evidence dataset. Of these preserved and stored memory dump datasets, 79 files of size 17.3 GB were generated during the attack. This means that only 21.76% of the data (in size) is potential evidence, and this can improve the storage and the processing time during the forensics investigation. However, these statistics only depict the current scenario of the simulation and are subject to change as per the duration of the attack. The results of our experiment regarding the storage requirement are presented in Table 10.

Table 10: Improvement in the storage efficiency

Description	Number of memory dumps	Size in GB
Data generated in cloud	360	281
Data generated during attack (potential evidence)	78	60

7 Conclusion and future scope

A traditional cloud forensics process involves manual detection and acquisition of evidence. In this article, we proposed a system equipped with an AI agent that automates evidence detection and acquisition. We have modelled the AI agent. The system classifies the data generated in the cloud as evidence or non-evidence data by monitoring and profiling the VM at the hypervisor level. We propose a novel feature-set of KVM-based hypervisor monitoring, which is highly de-correlated, giving better classification. The hypervisor-level monitoring ensures minimum privacy intrusion to a naïve user's VM and decreases the chances of malicious activities escaping the monitoring. The acquired evidence is stored using a simple blockchain mechanism to safeguard the evidence from tampering.

The proposed system is implemented and evaluated on the KVM private cloud platform. The evaluation results show that our system reduces the size requirement of the evidence repository by using AI agents and selective evidence preservation. The process provenance enhances the trustworthiness of the chain of custody for cloud forensics.

The current scope of the research presented in this article is limited to the acquisition of memory evidence. Future work can be undertaken to extend the scope to disk and network artefact acquisition using the Libvirt library. Autosklearn [35] can also be integrated to develop a cloud forensics tool that will aid the AI agent to select classification models and hyperparameter tuning for better results. Our next work is focused on developing a web-based cloud forensics tool that will include the proposed AI agent model for memory, disk, and network evidence detection and acquisition.

Conflict of interest: Authors state no conflict of interest.

Data availability statement: The datasets generated and analysed during the current study are available in the IEEE Dataport repository, "https://ieee-dataport.org/open-access/evidence-detection-cloud-forensics." DOI: https://dx.doi.org/10.21227/2yr5-7z67.

References

[1] Gartner. Gartner forecasts worldwide public cloud revenue to grow 17.5 percent in 2019. [Online]. 2019. https://www.gartner.com/en/newsroom/press-releases/2019-04-02-gartner-forecasts-worldwide-public-cloud-revenue-to-g

[2] M. M. H. Onik, C. S. Kim, N. Y. Lee, and J. Yang, "Privacy-aware blockchain for personal data sharing and tracking," *Open. Computer Sci.*, vol. 9, pp. 80–91, 2019.

[3] D. Quick and K.-K. R. Choo, "IoT device forensics and data reduction," *IEEE Access*, vol. 6, pp. 47566–47574, 2018.

[4] Y. Wu, Z. Zhang, C. Wu, C. Guo, Z. Li, and F. Lau, "Orchestrating bulk data transfers across geo-distributed datacenters," *IEEE Trans. Cloud Comput.*, vol. 5, no. 1, pp. 112–125, 2015.

[5] A. Aldribi, I. Traoré, B. Moa, and O. Nwamuo, "Hypervisor-based cloud intrusion detection through online multivariate statistical change tracking," *Computers Sec.*, vol. 88, p. 101646, 2020.

[6] K. Shaukat, S. Luo, V. Varadharajan, I. A. Hameed, and M. Xu, "A survey on machine learning techniques for cyber security in the last decade," *IEEE Access*, vol. 8, pp. 222310–222354, 2020.

[7] Trustwave global Security Report 2015. https://www2.trustwave.com/rs/815-RFM-693/images/2015_TrustwaveGlobalSecurityReport.pdf.

[8] A. T. Lo'ai and G. Saldamli, "Reconsidering big data security and privacy in cloud and mobile cloud systems," *J. King Saud. Univ-Computer Inf. Sci.*, vol. 33, no. 7, pp. 810–819, 2021.

[9] Z. Inayat, A. Gani, N. B. Anuar, S. Anwar, and M. K. Khan, "Cloud-based intrusion detection and response system: open research issues, and solutions," *Arab. J. Sci. Eng.*, vol. 42, pp. 399–423, 2017.

[10] M. Compastié, R. Badonnel, O. Festor, and R. Hev "From virtualization security issues to cloud protection opportunities: An in-depth analysis of system virtualization models," *Computers Sec.*, vol. 97, p. 101905, 2020.

[11] G. Meera, B. K. S. P. Kumar Raju Alluri, and G. Geethakumari, "CEAT: a cloud evidence acquisition tool for aiding forensic investigations in cloud systems," *Int. J. Trust. Manag. Comput. Commun.*, vol. 3, no. 34, pp. 360–372, 2016.

[12] D. Gonzales, J. M. Kaplan, E. Saltzman, Z. Winkelman, and D. Woods, "Cloud-trust – A security assessment model for infrastructure as a service (IaaS) clouds," *IEEE Trans. Cloud Comput.*, vol. 5, no. 3, pp. 523–536, 2015.

[13] A. Atamli, G. Petracca, and J. Crowcroft, "IO-Trust: an out-of-band trusted memory acquisition for intrusion detection and forensics investigations in cloud IOMMU based systems," *Proceedings of the 14th International Conference on Availability, Reliability and Security*, 2019.

[14] Z. Qi, C. Xiang, R. Ma, J. Li, H. Guan, and D. S. L. Wei. "ForenVisor: A tool for acquiring and preserving reliable data in cloud live forensics," *IEEE Trans. Cloud Comput.*, vol. 5, no. 3, pp. 443–456, 2016.

[15] L. A. Holt and M. Hammoudeh, "Cloud forensics: A technical approach to virtual machine acquisition," *2013 European Intelligence and Security Informatics Conference*, IEEE, 2013.

[16] S. Zawoad, R. Hasan, and A. Skjellum, "OCF: an open cloud forensics model for reliable digital forensics," *2015 IEEE 8th International Conference on Cloud Computing*, IEEE, 2015.

[17] S. Simou, C. Kalloniatis, H. Mouratidis, and S. Gritzalis, *A Meta-model for Assisting a Cloud Forensics Process*, vol. 9572, Springer Verlag, 2016, pp. 177–187.

[18] S. Khan, A. Gani, AWA Wahab, M. A. Bagiwa, M. Shiraz, S. U. Khan, et al., "Cloud log forensics: Foundations, state of the art, and future directions," *ACM Comput. Surv.* (*CSUR*), vol. 49, no. 1, pp. 1–42, 2016.

[19] P. Purnaye and V. Kulkarni, "A Comprehensive study of cloud forensics," *Arch. Comput. Methods Eng.*, vol. 29, pp. 1–14, 2021.

[20] W. D. Ashley, *Foundations of Libvirt Development*, New york: Apress, 2019.

[21] S. Russell, P. Norvig, and E. Davis, *Artificial Intelligence: A Modern Approach*, 3rd ed., Upper Saddle River, NJ, Prentice Hall, 2010. Print.

[22] N. Rakotondravony, B. Taubmann, W. Mandarawi, E. Weishäupl, P. Xu, B. Kolosnjaji, et al., "Classifying malware attacks in IaaS cloud environments," *J. Cloud Comput.*, vol. 6, pp. 1–12, 2017.

[23] G. Guo, H. Wang, D. Bell, Y. Bi, and K. Greer, et al., "KNN model-based approach in classification." *OTM Confederated International Conferences "On the Move to Meaningful Internet Systems,"* Berlin, Heidelberg, Springer, pp. 986–996, 2003.

[24] A. Hamoudzadeh and S. Behzadi, "Predicting user's next location using machine learning algorithms," *Spat. Inf. Res.*, vol. 29, pp. 379–387, 2021.

[25] H. Liu and H. Motoda, *Feature Selection for Knowledge Discovery and Data Mining*, Vol. 454, New York, NY, Springer Science & Business Media, 2012.

[26] V. Y. Kulkarni, P. K. Sinha, and M. C. Petare, "Weighted hybrid decision tree model for random forest classifier," *J. Inst. Eng.* (*India*)*: Ser. B*, vol. 97, pp. 209–217, 2016.

[27] W.-Z. Zhang, H.-C. Xie, and C.-H. Hsu, "Automatic memory control of multiple virtual machines on a consolidated server," *IEEE Trans. Cloud Comput.*, vol. 5, no. 1, pp. 2–14, 2015.

[28] M. Pourvahab and G. Ekbatanifard, "Digital forensics architecture for evidence collection and provenance preservation in iaas cloud environment using sdn and blockchain technology," *IEEE Access*, vol. 7, pp. 153349–153364, 2019.

[29] K. DeviPriya and S. Lingamgunta, "Multi factor two-way hash-based authentication in cloud computing," *Int. J. Cloud Appl. Comput.* (*IJCAC*), vol. 10.2, pp. 56–76, 2020.

[30] P. Kalyanaraman, K. R. Jothi, P. Balakrishnan, R. G. Navya, A. Shah, and V. Pandey, "Implementing hadoop container migrations in OpenNebula private Cloud Environment," *Role Edge Analytics Sustainable Smart City Development: Challenges and Solutions*. USA, Wiley, pp. 85–103, 2020.

[31] P. Purnaye and V. Kulkarni, "OpenNebula virtual machine profiling for intrusion detection system," *IEEE Dataport*, 2020, doi: 10.21227/24mb-vt61.

[32] P. Purnaye and V. Kulkarni, "Memory dumps of virtual machines for cloud forensics," *IEEE Dataport*, 2020, doi: 10.21227/ft6c-2915.

[33] T. Ito, Y. Kim, and N. Fukuta. I. C. Society, I. A. for Computer Information Science, I. of Electrical, and E. Engineers, *2015 IEEE/ACIS 14th International Conference on Computer and Information Science* (*ICIS*)*: Proceedings: June 28–July 1*, 2015, Las Vegas, USA.

[34] W. Kirch, "Pearson's correlation coefficient," *Encycl. Public. Health*, vol. 1, pp. 1090–1091, 2008.

[35] F. Fabris and A. A. Freitas, "Analysing the overfit of the auto-sklearn automated machine learning tool," *International Conference on Machine Learning, Optimization, and Data Science*, Cham, Springer, 2019.