How to Cite:
Czarnecka, P. (2020). The multi-tenant cloud computing architecture allows the service consumers to share the computing resources. *Tennessee Research International of Social Sciences*, *2*(1), 1-24. Retrieved from http://triss.org/index.php/journal/article/view/17

The multi-tenant cloud computing architecture allows the service consumers to share the computing resources

Paulina Czarnecka
Nicolaus Copernicus University, Toruń, Poland

***Abstract*---**The log integrity is proved by log chains which are created in the implemented system and by the potential electronic evidence of past logs which are posted by the cloud service provider. The proposed system aids in performing the reasonable verifications that the cloud service provider or the forensic investigator is not tampering the logs. The novelty of the research conducted in this paper is a technique which applies the cuckoo filter, to the forensic logs which is supportive in proving the integrity of the evidences at a faster pace in comparison to the other filters. Gathering and scrutinizing the different types of logs are the vital steps in the forensic domain. Logs are commonly gathered by the cloud service providers or by some third party layers governed by the cloud service providers. Security of the logs is a crucial issue as the logs can be tampered accidentally or intentionally by an employee in the cloud service provider's organization or by the forensic investigator, thus maligning the evidence in case a cyber-crime is committed through the cloud service provider's infrastructure. The malicious attacker can also conspire with the cloud service provider or the forensic investigator to erase or malign the logs that are generated for one's own criminal activity. To address such issues, a method is recommended which verifies the tampering of the virtual instance logs; Verification process confirms that the confidentiality and integrity of the logs remains intact.

***Keywords*---**Multi-tenant, Confidentiality, Forensic, Evidences, Scrutinize.

Introduction

Many services are offered by the Cloud Service Providers (CSPs) to millions of its service consumers referred to as tenants or clients or users. The multi-tenant cloud computing architecture allows the service consumers to share the

Tennessee research international of social sciences © 2020.
ISSN: 2766-7464 (Online)
Publisher: Smoky Mountain Publishing
Manuscript submitted: 09 March 2020, Manuscript revised: 18 April 2020, Accepted for publication: 27 May 2020

computing resources. The different services offered by the CSPs are Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS) as listed by National Institute of Standards and Technology U.S Department of Commerce (NIST) [1]. The various deployment models of cloud are private, public, community and hybrid cloud. Cloud environment has been adopted by many people as the business provided by it, is very promising in many countries. The latest research report states that the global cloud computing market having coverage of products, technologies and services is anticipated to cross $1 Trillion by 2024. India's booming cloud market is set to be worth $4.1 Billion by 2020[3].Cloud has many essential characteristics for the service consumers like on-demand self-service, broad network access, resource pooling, rapid elasticity or expansion, and measured service.

The contributions presented in this paper include:

1) A continuous forensics log system is introduced to monitor and collect the log which actually contains the evidences created by the users of this system. The proposed system preserves the confidentiality of the system users. Users of the system can be addressed as the cloud service clients or tenants.
2) The proposed system introduces the collection of Potential Electronic Evidence for Past Logs (PEEPL) which cannot be altered by any of the probable malicious entities like CSP or any employee in the organization or the forensics investigator.
3) The proposed system proves the confidentially (i.e. concealment) and the integrity of the evidences gathered using best key generation, encryption and hashing algorithm as per the need of the law of most of the countries.
4) The proposed system is constructed on upper layers of “Open Source cloud computing platform – OpenStack” where the logs are taken on regular basis and stored, which aids in solving the issues of volatile logs of virtual instances.
5) The modules of the system are built using python scripts which can be integrated later as a part of OS or as a layer of service in the future. Probabilistic data structures are employed like cuckoo filters and bloom filters which are supportive in proving the integrity of the logs which are evidences collected in the past, at a faster pace .The comparison of both the filters is provided.

This paper is organized as below: Section 2 discusses about the digital and cloud forensic process model, Section 3 discusses about related work, background and challenges. Section 4 gives a brief introduction on probabilistic data structures. Section 5 discusses about the threat model terminologies used in this paper. Section 6 discusses about proposed work and prototype implementation. Section 7 explains about the security analysis. Section 8 explains about the results observed. Section 9 includes discussion and section 10 finally concludes this research paper.

Digital and Cloud Forensic Process Model

Digital forensics is the procedure of preserving, collecting, confirming, identifying, analyzing, recording, and presenting the crime related evidence. Digital forensics is an applied science to recognize an incident, gathering, investigation, and

examination of evidence data as defined by NIST [18]. The Digital Forensic Process Model (DFPM) consists of four methods including gathering, examination, and analysis and reporting. The procedures here are expressed as a sequential progressive logic for better elucidations. Roussev et al, Kohn et al and Ahmed Nour Moussa et al have adopted the sequential logic process to capture and visualize the flow processes in the digital forensics world [4, 14 - 15]. The sequential progressive logic process notations employed in this paper is described as follows:

Digital Forensic Process Model = {Start process → Next process → then → End process}

The processes defined by NIST are collection, examination, analysis and reporting. The investigating process involved transmutes the data on the devices into evidences for internal use of the organization or for the usage of the law enforcement [1]. Collection of data happens in the initial stage from the devices and then this data is transmuted in a form useful for the forensic process. Data collection process gathers all possible sources of data after identifying them. The analysis stage is very significant as it transmutes the data into information which then leads to the reporting stage where this information is converted to evidence.

DFPM = {Data Collection → Data Examination → Analysis → Reporting}

Where

Data collection = {Identifying Possible Sources of Data → Acquiring the Data → Incident Response Considerations}.

Cloud forensics is defined as the science of conserving all potential evidences, safeguarding the concealment and integrity of the information, identification, collection, organization, presentation, and verification of evidence data to determine the events about an incident in the concerned cloud environment[1]. The cloud storage forensics process model (CSFPM) was presented by Darren Quick et al based on the intelligence analysis cycle and DFPM model [16]. It includes procedures like commence (scope), preparation and response, identification and gathering, conservation, examination, presentation, feedback and others. The CSFPM is related as:

CSFPM = {Commence → Preparation and Response → Identification and Collection → Preservation, Analysis, Presentation → Feedback → Complete}

The cloud network forensics process model (CNFPM) defined by Gebhardt T. et al has five horizontal processes data gathering, separation, accumulation, examination and reporting that act together with a management process represented by the symbol ← → below [16]. The process models employed here are referenced from DFPM model of NIST defined earlier in this section. The researchers have used OpenNebula-based (IaaS) environment for the implementation of the proposed system. They remotely analyze the Network Traffic moving in and out of the (IaaS) environment. All these process are running

inside the cloud environment and the clients of the system can get their forensic data on request to the respective role.

CNFPM = {{Data Collection → Separation → Aggregation → Analysis → Reporting}} ← → Management}

The Open Cloud Forensic Process Model (OCFPM) suggested by the researcher Shams et al [17] is incessantly supported by the CSPs. The model is from the DFPM model and defines six processes such as preservation, identification (incident and evidence), collection, organization (examination and analysis), conservation and verification. Sub processes are defined by both identification and organization processes This OCFPM is specified as:

OCFPM = {Preservation → Identification → Collection → Organization → Presentation → Verification}

Where

Identification = {*Incident → Evidence*}
Organization = {*Examination → Analysis*}

This section describes the various digital and cloud forensic models. The digital evidence plays a major role in the court of law along with the processes followed for the digital investigation process. These models though are very important and are followed very strongly by well-known researchers in this domain, but they do not include some important phases like preservation and verification stages except OCFPM. These models have been updated and are improvised by different researchers. The preservation and verification stages are important for the implemented system in this paper as the important entities of the system do the major processing in these stages. The research work followed in this paper is based on the processes of OCFPM.

Related Work, Background and Challenges

A solution in the cloud forensics domain has been proposed by many researchers and has been observed in the last few recent years. This section focuses on a few researchers who have provided a solution executed practically in cloud environments and a few who have just discussed the fundamental concepts. Ahmed Nour Moussa et al have proposed a cloud-forensic-as-a-service a live cloud forensic process model, a process which is bilaterally agreed evidence collection where customers can employ it as a service to gather criminological data from cloud. The CSP has the major control over the critical evidences, but the researcher has not addressed the issues of the CSP being dishonest, and the related measures [4]. Noëlle Rakotondravony et al have classified attacks in (IaaS) cloud that can be scrutinized by employing Virtual Machine Introspection (VMI) - based mechanisms. The authors say that VMI is proved to be an operative tool for malware detection and analysis in cloud computing environment. They focused on the attacks on virtual machines deployed in an (IaaS) cloud. The authors have addressed the attacks from the CSP's side, but have failed to address the issues of volatile logs. The authors have just classified the attacks and discussed about

its financial impact [5]. Adam J Brown et al suggested of constructing a balancing test for competing considerations of a forensic investigator acquiring information. The authors claim that maintaining privacy of data on cloud is a complex task. The authors have focused on pre-processing, acquisition and presentation phases and have tried to balance the user privacy and data integrity. The authors also claim that it's a difficult task to gather evidences in criminal cases and also difficult to meet the standards of United States Federal [6]. Nurul et al discussed about integrated cloud event behavior and forensic by design model. The authors have deployed the cloud storage applications in experiments in Drop box, Google Drive, and One Drive. They have collected and analyzed the residual data from cloud storage applications [7]. The experiments done by the authors were divided in two phases simulation phase followed by the packet capturing phase. The authors also have constructed the event scenarios with metadata captured from the past events. The authors have failed to address the issues of data integrity and accidental changes. M Edington Alex et al suggested about a consolidated forensic server and a forensic level called forensic monitoring plane (FMP) external to the specified infrastructure, after procurement of consent from the International Telecommunication Union (ITU). Thus, the investigators need not rely on the CSP for gathering facts as the said entities can be dishonest and thus can delay the process [9].Roberto Battistoni et al designed a CURE architecture that can detect timeline of events alterations to aid a forensic investigation process. A public cloud is used to deploy and test a CURE architecture. It pledges on the validity and trustworthiness of the investigation processes. The authors have not focused on the scaling factor and the integrity verification on distributed timeline [12]. Vassil Roussev formulated a forensic triage as a real-time computation problem with specific technical requirements, and used these requirements to evaluate the suitability of different forensic methods for triage purposes. The authors debated on the issues of forensic processes to be treated as soft real time problem [13]. Michael Kohn et al proposed a standardized digital forensic process model to assists the investigators in following a uniform approach in digital forensic investigations. The uniform approach is a must and a very significant step in investigations. The authors describe all the necessary steps for DFPM and how it needs to be presented [14]. Darren Quick et al an author of a book suggested about cloud forensic storage. The authors have discussed many case studies for analyzing the different cloud storage devices [15]. Gebhardt T. et al presented a general model for network forensics in the cloud and describes an architecture that reports a few issues in the domain of cloud computing. The authors have provided a remote access to the network forensics to the clients, ensures clear distinction between different user spaces in a multi-tenant environment. The system developed also shuns the cost of transferring the network data which is required for the Investigator by developing an internal service in cloud architecture. The authors validate the said architecture with a model for implementation based on the OpenNebula platform and the Xplico analysis tool [16]. Aarafat Aldhaqm discusses about the conceptual examination process model and an outline on database forensic investigation (DBFI) knowledge possessed algorithms, process models, procedures and artifacts forensics, which are very much applicable for DBFI users, practitioners and researchers in researching an imminent and undeveloped discipline [18]. Qiujin Zhang et al have built a model, an index system for confidentiality assessment for the cloud and also claim that no such model is built before. The authors have evaluated the

risks using the Information Entropy theory and Fuzzy logic. The authors have evaluated the privacy security risks in the cloud. The authors have also listed the privacy security risk from the popular CSPs like Google, Microsoft, Apple etc. [32]. Zhao discussed about the architecture which is secure for cloud computing alliance. The author has employed Game Theory and modelled the management of reliable resources of the cloud [33].Shams et al have suggested an Open model for Cloud Forensics which is termed as digital forensics [17]. Zawoad et al defined OCFPM which focuses on preserving sufficient electronically stored information (ESI) required to investigate cases involving clouds and ensuring the trustworthiness of such ESI. The Researcher worked on a case study, which is inspired from an actual civil lawsuit [21]. Zawoad et al also proposed a new accumulator scheme Bloom-Tree, which performs better than the other two accumulators in terms of time and space requirement. The authors claim that the false positive rate for the operations is also reduced in the system modeled by them [21-23]. The Google cloud uses the stack driver logging agent to log the significant elements, but how the security of the logs is handled and especially the forensic part of it is not available [31]. The logs are stored for 30 days, but the integrity issues remain transparent. Most of the commercial clouds keep such issues transparent and the servers are situated in different countries with different laws. All the CSPs may not obey the laws unless it's very legally and strictly followed in their respective country. The domain of cloud forensics is still a little immature and unexplored and has many issues due to which many criminals are getting away after performing the crimes. So the research focus of this paper is the design and implementation of a system which overcomes a few issues like volatile logs, safeguarding the privacy and confidentiality and proving the integrity of the logs [36].

Probabilistic Data Structures

When dealing with large amount of data, many of the classic data structures used to store, read, and update data become cumbersome and impractical to employ. In certain cases, it is better to use a different class of data structures and algorithms which employ randomness to mitigate some of these problems. Such data structures are called Probabilistic Data Structures (PDS). PDS are extremely useful for big data and use hash functions to randomize and compactly represent a set of items. These data structures use very less memory and have a constant query time. Some examples for PDS are bloom filter, cuckoo filter, skip lists, hyperloglog, count-min sketch etc.

Bloom filter is a **"***space-efficient probabilistic data structure that is used to test whether an element is a member of a set*". For instance, testing the presence of login names is a set membership query, where the set is the list of all registered username and can be done faster as compared to the search operations in the database. But the issues with PDS despite of being efficient is that they are probabilistic in nature which means the results of such data structures can be false positive which portrays that said item is previously present which actually may not be. The details of such results or the test conditions for the PDS are illustrated in figure 1. There are different types of bloom filter Standard Bloom filter (SBF), Counting Bloom Filter (CBF), multi-level bloom filter etc. SBF allows adding new elements to the filter and is characterized by a perfect true positive

rate (i.e. 1), but nonzero false positive rate. The false positive rate depends on the count of elements to be stored in the filter and on the filter's parameters, including the number of hash functions and the size of the filter. However, SBF lacks the functionality of deleting an element. Therefore, a CBF [10], providing the delete operation, is commonly used. The advance knowledge of the size of the CBF and the count of the elements to be stored will aid in optimizing the number of hash functions minimizing the false positive rate.

Cuckoo filter stores the fingerprints of a set of items which resolve collision employing a cuckoo hash table. Cuckoo filter is also termed as a compact alternative of a cuckoo hash table storing merely a few bits which are part of the item by employing a hash function for every item to be inserted. The filter efficiently occupies space even when it is densely filled, specifically when close to 100 % [19] [21]. A fingerprint as a hash function Φ: U $\rightarrow$ $\{0, 1\}^f$ which maps each element from the universe to its fingerprint, a short identifying string on f bits, such that if x = y then Φ (x) = Φ (y) and if x! = y, then Φ(x)! = Φ (y) with high probability (! = representing not equal to sign). The pseudo code for the insert and the lookup procedure is discussed later in this section. Bin Fan Et al have proved that "Cuckoo filters are practically better than the bloom filters" [8]. False positive results may be likely, but false negatives may never come up, to be precise, a query can return results which are "possibly in set" or "definitely not in set"[8]. Items or the logs may be inserted to the set. As the number of elements are inserted to the set, the likelihood of false positives increases.

Mareli, B. Twala et al have contributed by investigation of dynamically increasing switching parameter in Cuckoo Search algorithms performance [10]. They have optimized by employing an adaptive cuckoo search algorithm. Prashant Pandey et al have suggested a general-purpose Approximate membership query data structure which is minor and speedy and has good locality of reference and supports various operations like merging, counting (even on skewed data sets), and deleting and supports highly concurrent access [11]. The author claims that Counting Quotient Filter (CQF) performs good lookups even if the system is full up to 95%.

Shams Z. et al have used bloom filter for the implementation and verification of proof of past logs [23]. Bloom filters (BFs) have limitations which are overcome by the cuckoo filter. Cuckoo filters have the major advantages like roping in dynamic insertion and deletion of items. It affords higher lookup performance as compared to traditional BFs, even when it's literally jam-packed (e.g., 95% space exploited). In numerous real-world applications space used by cuckoo filters is far more less than the bloom filters with a very low false positive rate (<3%). A cuckoo filters can be used for some of the few significant key operations like add/ insert, lookup for an item which can return false positive result, delete an item etc. Probabilistic filters are employed in a many of the applications where slow or costly operations can be eluded prior to execution by a referring to a comparatively fast or cheap set membership test. Some of them are: Database Query Optimization (DBQO) and edge filtering. DBQO is applicable when data stored in a database can be inserted into a probabilistic filter. First the filter can be queried and tested to check if the data exists. If the results are successful then the expensive database operation

can be avoided. Edge filtering is useful in filtering queries for non-existent data received at the edge. Filters do not include the original data like cache.

The Pseudo Code for the Insert Procedure for Cuckoo Filter Is Specified Below

	Pseudo code for Insert procedure : Insert (LE)	
1	F = Fingerprint(IDLE);	
2	I = Hash (F);	
3	J = I ⊕ Hash (F) ;	
4	**IF** (Bucket (I) OR Bucket (J) has an empty entry **THEN**	
5	Add F to this Bucket;	
6	**RETURN** *successful ;*	
7	K = Randomly select I or J ;	*** Relocation of existing element in the cuckoo filter*
8	**FOR** N = 0 ; N < MaxKicks **; STEP** 1 **DO**	
9	Randomly select an element from Bucket (I) ;	
10	SWAP F and the fingerprint stored at in the location of the element	
11	K = I ⊕ Hash (F) ;	
12	**IF** Bucket(I) has an empty entry **THEN**	
13	Add F to this Bucket;	
14	**RETURN** *successful ;*	
15	**RETURN** *failure;*	***Hash Table is full of fingerprints*

The Pseudo Code for the Lookup Procedure for Cuckoo Filter Is Specified Below

	Pseudo code for Lookup procedure : Lookup (LE)	
1	F = Fingerprint(IDLE);	
2	I = Hash (F);	
3	J = I ⊕ Hash (F) ;	
4	**IF** (Bucket (I) OR Bucket (J) has an entry for F **THEN**	*** Check the presence of the fingerprint of LE*
5	**RETURN** *successful ;*	
6	**RETURN** *failure;*	*** if fingerprint not present*

The Insistent Database Log Entry (IDLE) used in the insert and the lookup Procedure is referred in figure 5 which is inserted in the cuckoo filter created at the top layer as specified in figure 4. The details are discussed in the section 6.Cuckoo filter stores only the hash values of the logs that are generated by the system created and implemented in this paper and not the original values. So no entity can access the orignal values once the logs are inserted to the cuckoo filter. The pseudo code specified above is built on similar lines from the research paper of Bin et al [8]. A more detailed description of test conditions for the PDS is specified in figure 1. A true positive test result is the one which identifies the log if it is present. A true negative test result is one which does not detect the log when the log is absent. A false positive test result is one which detects the log even when the log is absent. A false negative test result is one which does not notice the log when the log is existing [27].

		Conditions	
		Present	*Absent*
Tests	*Positive*	True Positive	False Positive
	Negative	False Negative	True Negative

Figure 1: Description of the Different Test Conditions

Threat Model Terminologies

The Table 1 describes the significant terminologies used in the proposed system. They are the different entities involved in the entire process of investigation. These fundamental terminologies are inspired from Zawoad et al. [22]

Table 1: Fundamental Terminologies

Fundamental Term	**Description**
Log	A log can be any of the OS logs , VI logs or the network process log,
Potential Electronic Evidence of Past Logs (PEEPL)	The PEEPL has the evidence of logs to safeguard the integrity of logs
Hash Chain (HC)	HC preserves the consecutive ordering of logs to guard the logs from regrouping or reorganizing.
CSP	A CSP is the proprietor of a public cloud infrastructure, when used by the clients generates the PEEPL, which is publicly available, and allows the processes to gather the logs.
Client	A client is a user. He is a customer to the CSP, who hires services like renting VIs which are a part of the CSP's Infrastructure. A client can be mischievous or truthful.
Forensic Investigator (FI)	The investigator is a proficient and skilful person in the forensics domain who is allotted the responsibility of gathering the required logs from clouds infrastructure in case of some crime or mischievous episode.

Electronic Evidence Examiner (EEE)	An EEE is a representative from the agencies appointed by the Central or State Government as per Indian Cyber Laws (Section 79 A). He is like an authority whose responsibility is to validate the accuracy of the logs using PEEPL and HC.
Intruder	Intruder can be some mischievous person, including employee of a CSP or even the investigator, who wants to disclose clients' actions from the PEEPL or from the stored logs

Proposed Work and Prototype Implementation

A Forensics Braced Cloud (FBC) is designed in the system implemented in this paper as shown in figure 2 which is based on the OCFPM discussed in Section 2. The OCFPM has various stages like preservation, identification (incident and evidence), collection, organization (examination and analysis), conservation and verification. The conservation and verification stages are very significantly needed to handle the issues related to volatile logs. The conservation stage has to be online which is termed as insistent database which is continuously throwing data to the Web server, the evidence publisher as referred in figure 2. Many manipulating entities exist as discussed in the threat model Table 2. They can be a threat to the logs when some crime is committed and thus want to modify the logs. Karen Kent et al have suggested how to integrate forensics techniques into incidence response as these are the guidelines from NIST [20].

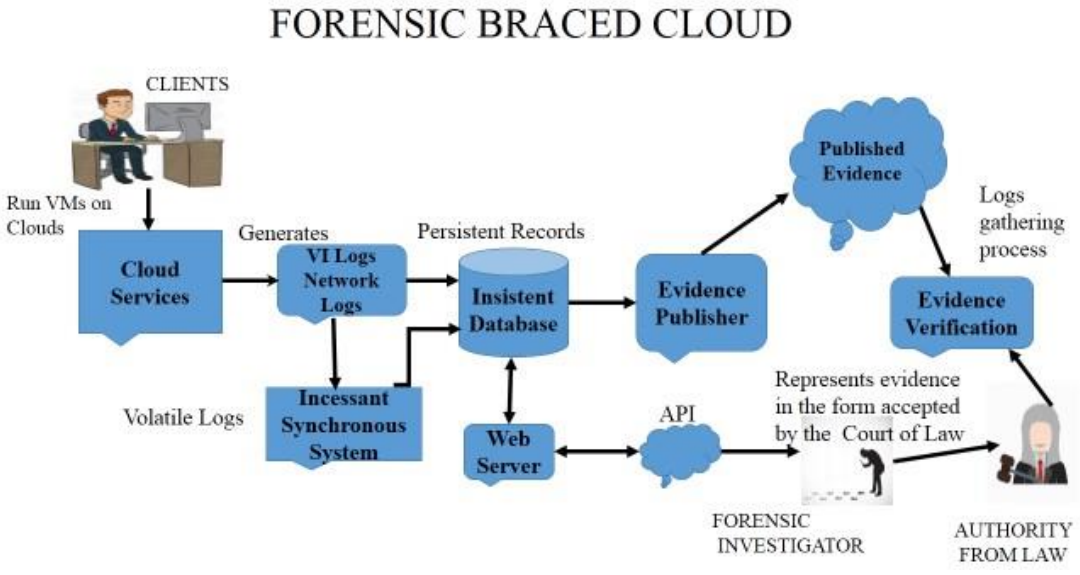

Figure 2**:** Forensics Braced Cloud

Figure 3: Cloud Forensics Process Flow

A client as specified in the threat model table 1 can use most of the cloud services hosted by the CSP. These services when used generate logs. In the implemented model of this paper the Virtual Instance (VI) logs are considered which is referred to as VI in the prototype environment configuration referred in figure 4. A client can hire the VIs for some time, commit some crime and close the VIs assuming that he cannot be traced by anybody as the VIs are shut down. A few such cases are stated in some countries like USA, India etc. as per the reports [34-36]. The experts relate about legal evidence gathering issues, issues of not reporting about the crimes, issues of honoring the warrants of arrest when cross-boundary problems arise. Some countries like China and Russia will never honor the warrants of arrests. Such matters can be handled through Interpol and international talks which can lead to a consensus between the nations. The implemented system stores these logs as a set of evidences which can be useful in the court of law as per their requirement. A CSP cannot amend or delete any of the logs generated because of the security provided by the system. If the CSP tries to delete some logs or modify some logs, it can be easily identified by the implemented system. FI can perform the analysis of the logs when a cybercrime is committed using these VIS. The FI can present the set of evidences and prove its integrity in the court of law. The cybercriminal can thus be prosecuted. The prototype environment configuration is shown in figure 4.

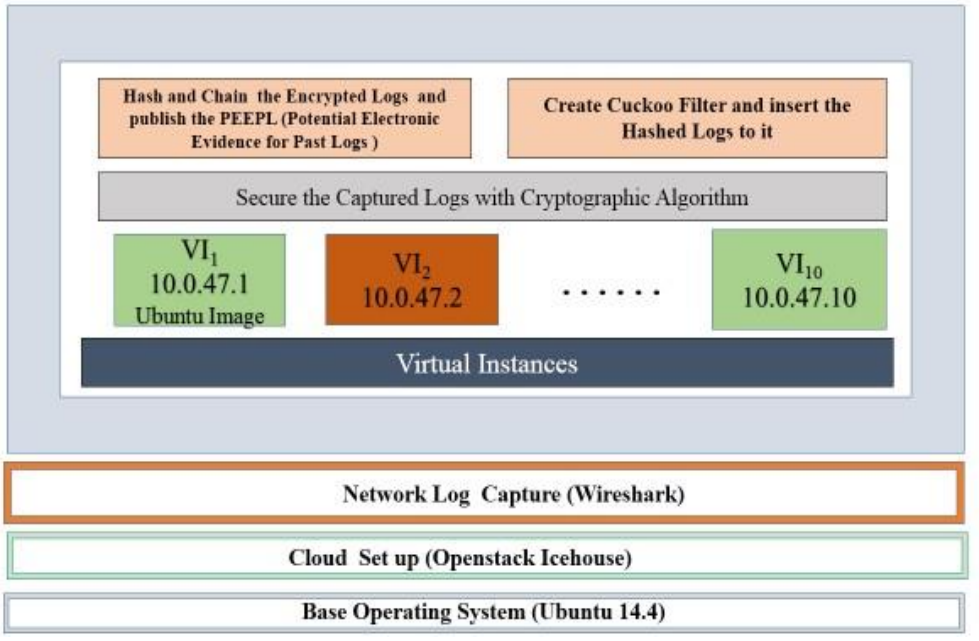

Figure 4: Prototype Environment Configuration

- LE =< SrcIP; DstIP; Ts; SrcPort; Dst Port; UserId >; ** Log Entry
- ELE= { EPKA(DstIP; DstPort; UserId,Ts); SrcIP; } ** Encrypted using ECC Log entry
- HC ={ $H(ELE;HC_{Preceding})$} ** Hash Chain
- IDLE =< ELE;HC > ** Insistent Database Log Entry

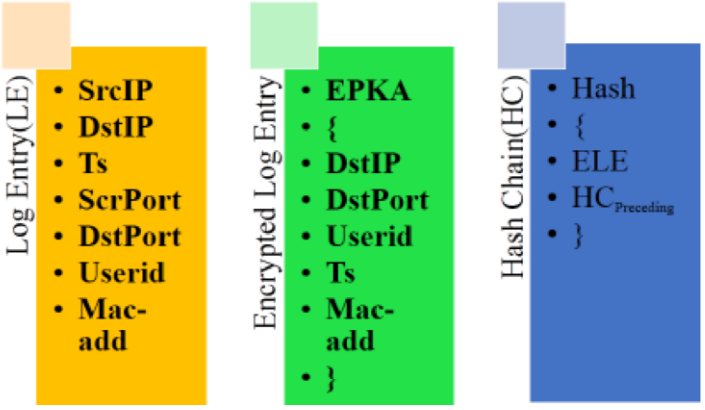

Figure 5: Encryption and Hash Chaining Process for the Logs Captured

Ubuntu 14.4 is used as the base OS in the system implemented here. VIs are created using the OpenStack Icehouse referred to as VI with different IP addresses allotted during creation as shown in figure 4. A VI is allotted “10.0.47.1” IP addresses from the pool of IPs. Similarly other VIs is created in the similar manner.

The users can hire and use these VIs and make request for URLs or the websites as shown below:
"1963","2019-01-18
10:06:12.159773344","10.0.47.2","52.222.187.201","TCP","78","[TCP Dup ACK 1959#1] 55857 → 443 [ACK] Seq=1261 Ack=220916 Win=1444 Len=0 TSval=1149679
TSecr=1237403989 SLE=222364 SRE=223812The above log is a request that is captured in the set up environment using Wireshark installed in the Node Controller of the Openstack cloud which specifies that VI with IP “10.0.47.2” is sending an HTTP request to “52.222.187.201” machine at “2019-01-18 10:06:12.159773344" time . The request-id is “1963” with TCP protocol log shown above. The details of the user who is allotted VI “10.0.47.2” can be captured from the “Nova” mysql database of Openstack. The logger module captures all these elements as shown in figure 5. The RE module is a Regular Expression used to extract the significant elements shown in figure 6. It’s a python module to parse the significant elements.

Wireshark	Logger	RE	Cryptos / HC	PDS	PEEPL	Web Publishing
(1) Log collection/ capturing process	(2) Stores the Logs	(3) Regular expression in Python to extract the significant elements for the Logs captured	(4) Cryptographic procedure applied to the Log Elements to encrypt them and the hash chain is formed	(5) Encrypted and chained elements are added to Cuckoo Filter or Bloom Filter (IDLE)	(6) IDLE signed using private /public keys	(7) PEEPL published on the WEB

Figure 6: Transformation from Log to PEEPL Published On Web through Pds

The implemented system uses a cryptographic algorithm ECC (Elliptic Curve Cryptography) [28]. ECC Algorithm aids are creating quicker, smaller and more efficient public and private keys. The keys generated are asymmetric in nature. As per literature survey, a 256 bit ECC key is more efficient as compared to a 3072 bit RSA key. Such keys with minimum bits offer strong security and fast SSL handshakes. Applying ECDSA (Elliptic curve digital signature algorithm) to the system, the private key of the CSP can be employed to generate a signature of the logs and the other key can be used by the FI for decrypting when needed to verify the evidences. A hash of the log is computed using the cryptographic SHA512 hash function, which supports in the integrity of the log chain referred to as Hash Chaining (HC) in the table 1 and figure 5. The computed hash value can be used to verify the integrity of copies of the original logs without providing any original data. Hashing process is one way, not like encryption process in which decryption is essential. The logs are signed by the CSP at the end of the day at a specified

time before the next day starts. The process of signing of logs and keeping the backup logs can be made mandatory by the Cyber law.

The VI logs recorded in the implemented system would have the following elements: source IP (SrcIP), destination IP (DstIP), time stamp (Ts) the time at which these web sites were requested or accessed, source port (ScrPort), destination port (DstPost). The User-id can be obtained from the Nova database of the OpenStack. The log description has been shown in figure 5. Few items of the logs are encrypted using the ECC algorithm as shown in figure 5 which is referred earlier and are denoted as ELE. A HC as shown in figure 5 is formed using the hashing algorithm which takes the ELE and HC of the previous record. At the initial step, when the first log is taken the HC value can be set to some assumed negative value as there will be no preceding log. A hash chain is formed for all the accesses made by the different clients while using the services of the CSP as shown in figure 5. These logs are stored in the insistent database and also on the web server under control of some other CSP. Such multiple copies can be stored with different CSP's if required. The encrypted log plus the hash code of the previous encrypted log gives a better security level for the logs. As encryption is applied to the log elements in addition to the hash code, which assists in providing the confidentiality and integrity. The PEEPL is signed by the CSP using the secret key generated using ECDSA. The elements used for PEEPL as listed below:

$$Peepl = \{Idle, Tp, Signature_{PV}(Idle, Tp)\}$$

IDLE is shown in figure 5; Tp is the time stamp which specifies the evidence generated time. Signature (IDLE, Tp) is the signature on (IDLE, Tp) by the CSP by employing the private keys. The other key is shared with the FI or the EEE and only they can view the data and no one else, so the privacy and the authenticity is maintained. Multiple copies are kept at various Web servers and the other CSPs if required, forming a trust model among them. The FI when allocated the duty of an analyst can check the hash value of the log using the key allotted in case a crime is detected and the logs on the server belong to the CSP, who's Infrastructure was used to pursue the crime. If the hash code of the forensic log in the backup copy is same with the one the CSP has, then it concludes that it's not compromised till date and the logs and the evidences are said to be authentic. Also the CSP will be alert in such cases and as the backups are available, the CSP cannot deny giving any data in raw form. Thus if a crime is committed by using the hosted services by the client and then closes the VIs after committing the crime , the CSP has the data of the past logs PEEPL even if the VMs are shut off to hide the crime. The framework for such a system and its need is discussed and implemented, but is not optimized [24 – 25]. Now in the proposed system the cuckoo filters and bloom filters are created on the top layer as shown in figure 4 which is explained in detail in the next section. The logs are inserted to the cuckoo filter as soon as they are generated and chained as a part of the chaining process. This is referred to as PDS in figure 5, then published on the web server after the signing process is complete.

Security Analysis

The conspiracy model inspired from Zawoad et al [22] which is slightly altered for the system implemented here as shown in Table 2 has mainly three persons the CSP, user who is a client and the FI. CSP is represented as C if the CPS is trustworthy and ~~C~~ if the CSP is not trustworthy. CSP has the complete hold over for the logs which are stored and which can act as evidence. Thus the CSP cannot be trusted. FI is represented as F if the FI is trustworthy and ~~F~~ if the FI is not. As FI can acquire logs and if the FI conspires with the CSP or the user, the FI can manipulate the logs before showing it to the EEE or the Court. The user can also conspire for manipulating the logs if the said user has committed some crime. The represented as ~~U~~ and U if the user is honest. Any step or person manipulating the logs can be detected by the system designed here, which is providing tamper proof evidence. The implemented system has to follow the basic concealment and integrity properties which ensure security of the logs as discussed below.

Basic Concealment and Integrity Property

The system designed ensures the safety of the logs in the persistent storage that is the insistent database and also disallows any malicious actor from inserting or creating fake PEEPLs. Also the system ensures that the FI is not manipulating the logs. The logs generated by the system here should own the following concealment (Con) and integrity (Int) properties:

Con1: An oppugner (attacker or malicious entity) will not be able to qualify for the logs from the published PEEPLs.
Con2: A mischievous cloud employee will not be able to recover concealed information from the stored logs.

Int1: A CSP will not be able to eliminate a log entry from the storage once the PEEPL is published.
Int2: A CSP cannot alter or reorder the logs from its actual order of generation.
Int3: A CSP cannot insert a fake log after-the-fact.
Int4: A FI will not be able to conceal or remove a log entry at the time of presenting logs to court. Int5: A FI will not be able to alter the chronological order of a log entry at the time of presenting evidences to court.
Int6: A FI will not be able to provide the duplicitous
logs to the court. Int7: CSPs will not be able to deny
previously published PEEPLs.

Table 2 lists the different types of compromises possible and how the compromises can be handling or defended using the essential security properties. The concealment property Con1, Con2 holds true and is justified in the system implemented because store the logs are not stored in the original form but the hash of it. As the hash functions are one way (original data cannot be obtained) and also the logs are encrypted, so no external attacker or any malicious employee can recover any concealed information from the logs. The integrity properties Int1, Int2, Int4 and Int5 support non removal and sequential disordering of the logs by the malicious CSP or dishonest FI. If any of the logs are

removed or replaced after the PEEPL is published, the hash chain process will identify the difference in hash values. The hash values are also not found in the cuckoo filter as it is trained in advance as soon as the logs were generated. The cuckoo filters never give false negatives. This process flow is shown in figure 6. The EEE and the FI can use this method as a verification process for validity of PEEPL and can conclude whether the integrity is compromised. A dishonest FI can take advantage of unavailability of logs for a few hours, but in the system the logs are signed by the CSP when the day changes using the private key and also kept on the Web, in which case such claims will remain an entertained. The counterfeit or the duplicitous logs which cannot be inserted by the CSP or the FI as per the Int3, Int6 properties. The hash of the fake logs will vary and will be absent in the cuckoo filter created in the implemented system. This again can be traced using the process flow shown in figure 7 and figure 8. The Int7 integrity property supports the non-repudiation of PEEPL by the CSPs. As the logs are already published earlier and also signed by the CSP using the private key. The PEEPL can also be unlocked using the public key. The CSP cannot deny the process and the logs.

Table 2: Conspiracy Model

Sr-No	Is Trug 3stworthy			Conditions	Compromise Status	Essential Security Properties Needed
	CSP (C)	Client (User) (U)	FI (F)			
1	Y	Y	Y	CUF	no conspiracy	Not any
2	N	Y	Y	~~C~~UF	discloses clients activity from the logs	Con2
3	Y	N	Y	C~~U~~F	recover clients log after the PEEPL is published	Con1
4	Y	Y	N	CU~~F~~	eliminate, reorganize and insert counterfeit logs	Int4, Int5,Int6
5	Y	N	N	C~~UF~~	eliminate, reorganize, insert counterfeit logs, and recover other cloud clients log	Con1, Int4, Int5,Int6
6	N	Y	N	~~C~~U~~F~~	eliminate, reorganize, insert counterfeit logs, disown published PEEPL and disclose other cloud clients activity	Con1,Int1, Int2, Int3,Int4, Int5,Int6,Int7
7	N	N	Y	~~CUF~~	eliminate, reorganize, insert counterfeit logs, disown published PEEPL and disclose other cloud clients activity and logs	Con1,Con2,Int1, Int2, Int3,Int7
8	N	N	N	~~CUF~~	eliminate, reorganize, insert counterfeit logs, disown published PEEPL and disclose other cloud clients activity and logs	Con1,Con2,Int1, Int2, Int3,Int4, Int5,Int6,Int7

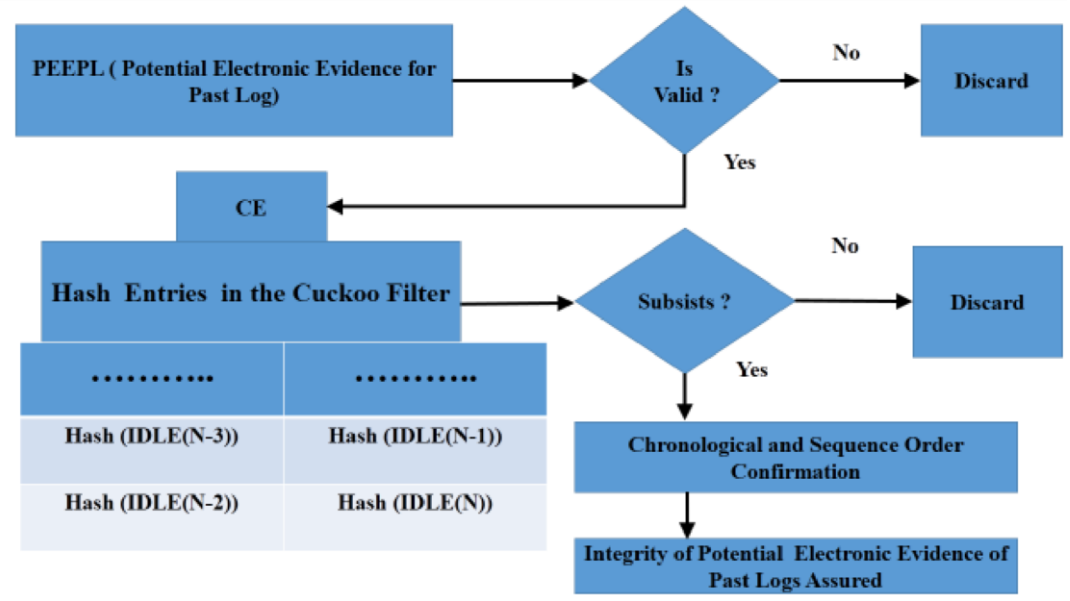

Figure 7: Process Flow Diagram for Log Confirmation and Integrity Assurance

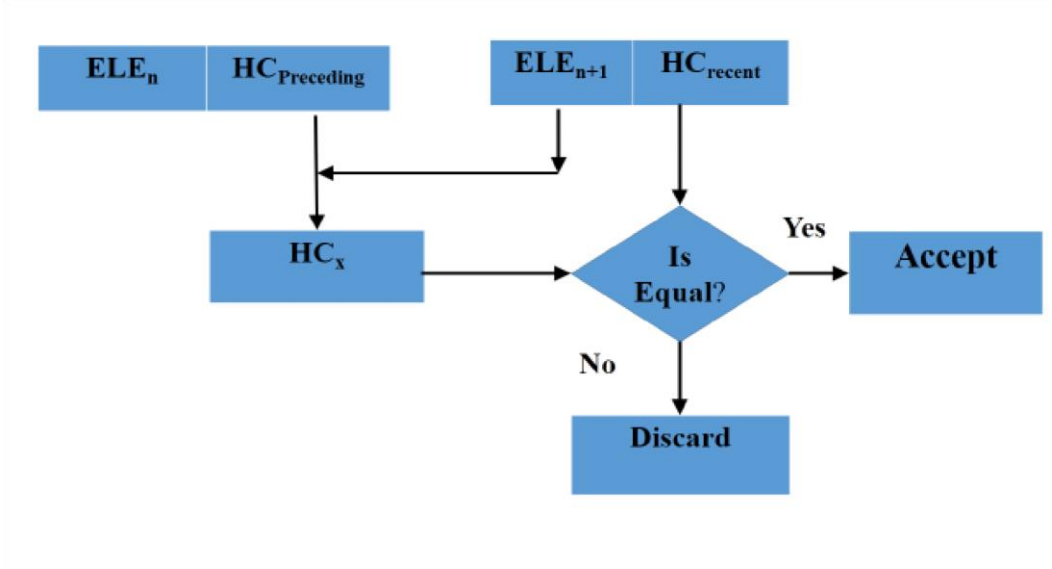

Figure 8: Process Flow Diagram for Sequential Order of Log Confirmation

Results Observed

The prototype environment configuration is shown in the figure 3. A cuckoo filter is created using the python library and logs which are encrypted and hashed, are added to the cuckoo filter. The cuckoo filter in this trained. The implemented system has used cuckoo filter for optimizing the database search operations. A very few libraries exists for cuckoo filter [26 – 28] as of now because of its recent creation. The Lookup (IDE) operation against the cuckoo filter can detect the presence of the hash of the logs that is the PEEPL inserted by the Insert (IDLE)

procedure. The operations Insert (IDLE) and Lookup (IDLE) performed on the cuckoo filter can either return successful or failure. The output behavior or the output test condition of the cuckoo filter can be false positive as shown in figure 1 because of its probabilistic nature which is less than or equal to 3%.

When the FI and the EEE want to check the presence of the order of the hash of the logs in case of a committed crime, they can be given an interface to check the contents of the trained cuckoo filter. They can use the cuckoo filters created in the implemented instead of database and if the order of logs i.e. the hash values are present in the specified manner means the result is true positive. When the crime is committed using the CSPs infrastructure, the integrity of the logs can be proved in the court of law using the evidence for crime committed. This evidence can be traced optimally fast from the CSPs infrastructure where the cuckoo filters are built. For confirming the same, the database can also be referred. Bloom filters are also created in the similar manner and compared with cuckoo filters. The bloom filter and the cuckoo filter are the probabilistic data structures discussed in section 4.

The graph in figure 9 shows that the PEEPL generation time is linear. The graph in figure 10 shows that cuckoo filter is very efficient in integrity verification as compared to database or bloom filter. The lookup operation performed on the cuckoo filter gave a 0(zero) false positive result. During the insertion operation a 0 (zero) false positive result was obtained. During verification process, the FI can perform a set membership test on the cuckoo filter to verify the hash chain process. This is avoiding the database which saves a lot of time. The table 3 shows the observed parameters. The results observed are specific to the executed environment.

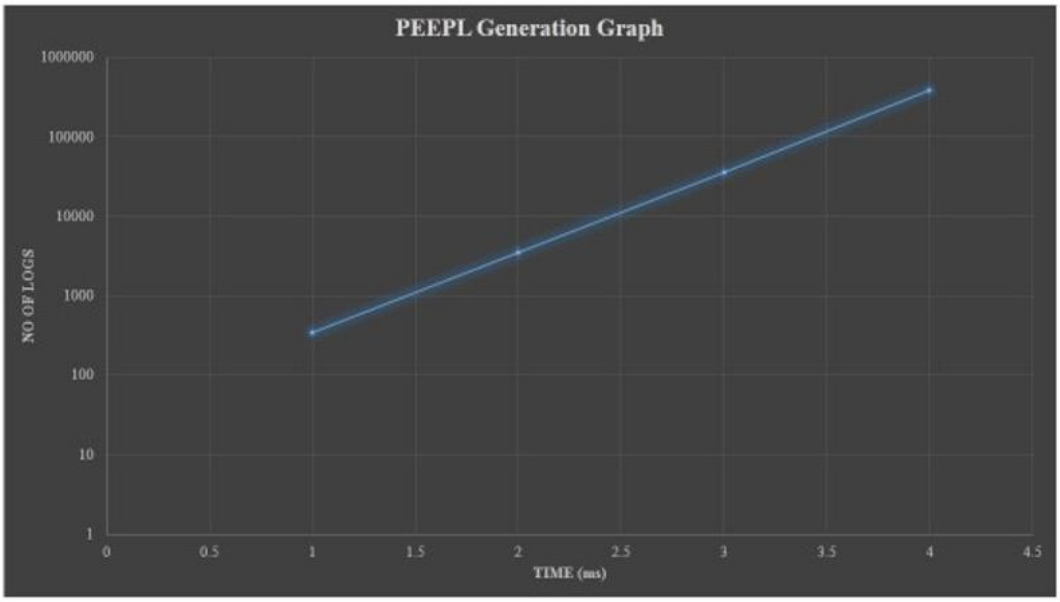

Figure 9: PEEPL Generation Graph

Table 3: Observed Parameters

#Sr-No	Description	Time Noted in Milliseconds (ms)
1	Key pair generation	17.659
2.	Encryption time for a single Log	3.4029
3.	Hashing time	0.0131
4.	Encryption and Hashing	3.4160
5.	Database search time for an existing Log	4.380
6.	Cuckoo Filter Look up time	0.060
7.	Bloom Filter Look up time	0.250

Discussion

The previous experiments conducted by the authors of this paper stores the records in the database [25]. The hash chain formation process is explained in figure 5 remains the same for the current research. The database table for the logs with the column names like ScrIP, DstIP etc. as specified in the figure 5 are created using Mysql database and the logs are stored in the persistent database. Pseudo Code for the earlier system designed and implemented is as below:

	Pseudo for the Log Hash Chaining	
1	*TimeInterval =Tm*	*Time interval as per the policy*
2	*L = Log from the Virtual instance V*	*Logs captured using the Wireshark*
3	*Userid From Nova database*	*Nova database of Openstack*
4	*For each log L do*	*Till end of the day*
5	SrcIP = Search_Pattern(SourceIP,L)	*Few elements are searched from the Log*
6	DstIP = Search_Pattern(DestinationIP,L)	*Regular expressions are written in python for the Search*
7	Ts= Search_Pattern(Timestamp,L)	
8	DstPort = Search_Pattern(Timestamp,L)	
7	Macadd = Search_Pattern (Macadd,L)	
9	EC = Concat(DstIP,DstPort,Userid,Ts,Macadd)	
10	ELE = Encrypt(EC)	*Encrypt using a standard Crypto*
11	HC = Hash(ELE, HC_{prev})	*Generate the Hash using standard Hashing Algorithm, use the hash of the previous log to form the Hash chain*
12	Send the HC to IDLE	
13	*If Timeofbackup == Tm then*	*Regular backups are taken*
14	PEEPL = CSP.Signature(IDLE)	*Signed values are published*
14	Publish the evidences on the EvidencePublisher	*Publish the evidences on the Web*
15	*If EOD then*	*It's the Epoch*
16	Sign the last log using the Private Key of the CSP	
17	Send the L, HC to IDLE	
18	*End if*	
19	*End for*	

	Pseudo for the evidence verification by the FI using the database	
1	*L= Log(SrcIP,DstIP,Ts)*	*FI gets Logs through Investigation*
2	***If L Exists in the Log Database of CSP then***	*Matches with the CSPs details*
3	*FI Publickey; Unkey PEEPL*	*Public key provided by the CSP*
4	***If CSP.PEEPL == EvidencePublisher.PEEPL***	*Test Hash Chain for some samples*
5	Evidence is genuine and integrity maintained	*Hash values are equal ,integrity of logs is intact*
6	Proceed for court of law for trials	
7	***Else***	
8	Evidence is compromised or wrong	*Hash values are not equal*
9	***End if***	
10	**If CSP's Database is Empty or Compromised**	
11	Invoke fine procedure by Law	*Suggested as per Law*
12	***End if***	

	Pseudo for the evidence verification by the FI using cuckoo filter	
1	*L= Log(SrcIP,DstIP,Ts)*	*FI gets Logs through Investigation*
2	*LE = Hash(L)*	
3	***For each log in L***	
4	*LE = Hash(L)*	
5	*T = Lookup(LE)*	*Lookup procedure for the trained Cuckoo Filter is invoked*
6	***If T is true then***	*Test Hash Chain for some samples*
7	*Hash exists; evidence is not tampered; proceed for court of law for trials*	*Hash matches ; Logs are genuine*
8	***else***	*Hash values are equal ,integrity of logs is intact*
9	*Hash does not exists; Evidence is tampered*	*Hash values are not equal*
10	***End if***	
11	**If CSP's Database is Empty or Compromised**	*Hash values are not equal*
12	Invoke fine procedure by Law	*Suggested as per Law*
13	***End if***	

Regular expressions are coded in python to extract the other significant elements from the logs. After the process of chaining and hashing, the logs are added to the database in the order in which the logs are generated. The hash chain is also stored in order to ensure that the proper chain is formed. Searching a hash value in the database is a very costly process. In the earlier system developed by the authors the purpose of hash chaining is served and the integrity can also be maintained, though the operations are very costly. The privacy and the confidentiality is compromised as the table contents and logs can be seen by the CSP's employees [25]. The table contains the logs of the websites accessed and other details of the users of the system which can be revealed easily to the CSP and the employees. The evidences can be easily compromised in the database

thus compromising the confidentiality and integrity of the logs. The evidences can be questionable in the court of law. The system implemented in this research paper optimizes the costly operations as discussed above in the results observed section 8 and also improves on the data privacy and confidentiality. The contribution of the current research is improving the data privacy and proving the integrity of logs using the cuckoo filters and bloom filters. The insert procedure needs to be invoked when the logs are generated to train the cuckoo filter. The FI can use the interface for the cuckoo filter designed using the above pseudo code. Bloom filters are also employed in the similar manner. A slightest change in the logs can be detected by the implemented system. The users of the system cannot deny the logs generated for the cloud services used by them and disown them.

Conclusion

The cuckoo filters and bloom filters are employed for proving the integrity of logs. The implemented system is employing the cuckoo filters for optimizing the database operation which are very costly. For example, querying a database, a set membership test can be done to see if the desired log is even in the database. Cuckoo filters support strongly in proving the integrity of the logs. Cuckoo filters are new and stable libraries for many languages simply don't exist. The results observed in this paper shows that cuckoo filters are very promising in the forensic area as compared to database and bloom filters. To execute a successful and rapid forensics investigation in clouds cuckoo filters work at a faster pace as compared to bloom filter. In the cloud domain there are issues related to volatile logs, making the logs accessible to investigators, and also simultaneously safeguard the confidentiality and integrity of the logs. Such issues are solved in this paper. Also the PDS used are gaining a lot of importance in the domain of large data sets. The proposed system implemented here preserves the confidentiality of cloud clients. The EEE can verify the integrity of the logs using the PEEPL and the hash chain of the logs which are in the trained cuckoo filter. The prototype system uses Ubuntu 14.4 as the base operating system and the system is implemented in python. The earlier system implemented was using Mysql database [25]. The earlier system was slow as shown in the results. This newly built system is fast and can aid the FI in proving the crime at a faster pace and also aids in preserving the integrity and confidentiality of the logs. No user of the system can deny the logs generated for the cloud services used by them and disown them.

Figure 10: Log Integrity Verification Graph

Acknowledgements

The authors are thankful to the anonymous reviewers and the editors for their suggestions. The authors are also thankful to the managers for quick and timely responses. The authors also thank the staff members of LJK trust, Dr. H.B.Bhadka of C.U.Shah University, Dr. Sameer Patel of PDPU, Dr. Manik lal Das from DAIICT for their best co-operation, support and guidance.

References

Almorsy, M., Grundy, J., & Ibrahim, A. S. (2011, July). Collaboration-based cloud computing security management framework. In *2011 IEEE 4th International Conference on Cloud Computing* (pp. 364-371). IEEE.

Almorsy, M., Grundy, J., & Ibrahim, A. S. (2012, June). Tossma: A tenant-oriented saas security management architecture. In *2012 IEEE Fifth International Conference on Cloud Computing* (pp. 981-988). IEEE.

Almorsy, M., Grundy, J., & Müller, I. (2016). An analysis of the cloud computing security problem. *arXiv preprint arXiv:1609.01107*.

Al-Shardan, M. M., & Ziani, D. (2015). Configuration as a service in multi-tenant enterprise resource planning system. *Lecture Notes on Software Engineering*, *3*(2), 95.

Ashalatha, R., & Agarkhed, J. (2016, March). Multi tenancy issues in cloud computing for SaaS environment. In *2016 International Conference on Circuit, Power and Computing Technologies (ICCPCT)* (pp. 1-4). IEEE.

Azeez, A., Perera, S., Gamage, D., Linton, R., Siriwardana, P., Leelaratne, D., ... & Fremantle, P. (2010, July). Multi-tenant SOA middleware for cloud computing. In *2010 IEEE 3rd international conference on cloud computing* (pp. 458-465). IEEE.

Bakshi, K. (2011, March). Considerations for cloud data centers: Framework, architecture and adoption. In *2011 Aerospace Conference* (pp. 1-7). IEEE.

Behl, A., & Behl, K. (2012, October). An analysis of cloud computing security issues. In *2012 world congress on information and communication technologies* (pp. 109-114). IEEE.

Bezemer, C. P., & Zaidman, A. (2010, September). Multi-tenant SaaS applications: maintenance dream or nightmare?. In *Proceedings of the Joint ERCIM Workshop on Software Evolution (EVOL) and International Workshop on Principles of Software Evolution (IWPSE)* (pp. 88-92).

Brookbanks, M. D., Coffey, B. F., Dawson, C. J., Nellutla, T., Patterson, R. C., & Seaman, J. W. (2011). *U.S. Patent Application No. 12/630,079*.

Hajibaba, M., & Gorgin, S. (2014). A review on modern distributed computing paradigms: Cloud computing, jungle computing and fog computing. *Journal of computing and information technology*, *22*(2), 69-84.

Kriouile, H., & Asri, B. E. (2018). A rich-variant architecture for a user-aware multi-tenant SaaS approach. *arXiv preprint arXiv:1812.08253*.

Mahmood, Z. (2011). Cloud computing for enterprise architectures: concepts, principles and approaches. In *Cloud computing for Enterprise architectures* (pp. 3-19). Springer, London.

Malathi, M. (2011, April). Cloud computing concepts. In *2011 3rd International Conference on Electronics Computer Technology* (Vol. 6, pp. 236-239). IEEE.

McGrath, M. P., & Lamourine, M. A. (2015). *U.S. Patent No. 9,058,198*. Washington, DC: U.S. Patent and Trademark Office.

Mishra, A., Mathur, R., Jain, S., & Rathore, J. S. (2013). Cloud computing security. *International Journal on Recent and Innovation Trends in Computing and Communication*, *1*(1), 36-39.

Motahari-Nezhad, H. R., Stephenson, B., & Singhal, S. (2009). Outsourcing business to cloud computing services: Opportunities and challenges. *IEEE Internet Computing*, *10*(4), 1-17.

Odun-Ayo, I., Ananya, M., Agono, F., & Goddy-Worlu, R. (2018, July). Cloud computing architecture: A critical analysis. In *2018 18th International Conference on Computational Science and Applications (ICCSA)* (pp. 1-7). IEEE.

Odun-Ayo, I., Misra, S., Abayomi-Alli, O., & Ajayi, O. (2017, December). Cloud multi-tenancy: Issues and developments. In *Companion Proceedings of the10th International Conference on Utility and Cloud Computing* (pp. 209-214).

Povedano-Molina, J., Lopez-Vega, J. M., Lopez-Soler, J. M., Corradi, A., & Foschini, L. (2013). DARGOS: A highly adaptable and scalable monitoring architecture for multi-tenant Clouds. *Future Generation Computer Systems*, *29*(8), 2041-2056.

Rimal, B. P., & Choi, E. (2012). A service_oriented taxonomical spectrum, cloudy challenges and opportunities of cloud computing. *International Journal of Communication Systems*, *25*(6), 796819.

Rimal, B. P., Choi, E., & Lumb, I. (2009, August). A taxonomy and survey of cloud computing systems. In *2009 Fifth International Joint Conference on INC, IMS and IDC* (pp. 44-51). Ieee.

Shue, D., Freedman, M. J., & Shaikh, A. (2012). Performance isolation and fairness for multi-tenant cloud storage. In *Presented as part of the 10th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 12)* (pp. 349-362).

Takabi, H., Joshi, J. B., & Ahn, G. J. (2010). Security and privacy challenges in cloud computing environments. *IEEE Security & Privacy*, *8*(6), 24-31.

Tsai, W. T., Sun, X., & Balasooriya, J. (2010, April). Service-oriented cloud computing architecture. In *2010 seventh international conference on information technology: new generations* (pp. 684689). IEEE.

Voorsluys, W., Broberg, J., & Buyya, R. (2011). Introduction to cloud computing. *Cloud computing: Principles and paradigms*, 1-44.

Vuyyuru, M., Annapurna, P., Babu, K. G., & Ratnam, A. S. K. (2012). An overview of cloud computing technology. *International Journal of Soft Computing and Engineering*, *2*(3), 244-247.

Willis, D., Dasgupta, A., & Banerjee, S. (2014, September). ParaDrop: a multi-tenant platform to dynamically install third party services on wireless gateways. In *Proceedings of the 9th ACM workshop on Mobility in the evolving internet architecture* (pp. 43-48).

Wu, D., Rosen, D. W., & Schaefer, D. (2014). Cloud-based design and manufacturing: status and promise. In *Cloud-based design and manufacturing (CBDM)* (pp. 1-24). Springer, Cham.

Wu, D., Rosen, D. W., Wang, L., & Schaefer, D. (2015). Cloud-based design and manufacturing: A new paradigm in digital manufacturing and design innovation. *Computer-Aided Design*, *59*, 1-14.

Youssef, A. E. (2012). Exploring cloud computing services and applications. *Journal of Emerging Trends in Computing and Information Sciences*, *3*(6), 838-847.