# CLOUD LOG REASSURING SOUNDNESS AND SECRECY THEME FOR CLOUD FORENSICS

Preethi. D , Raja A

IV th Sem M.Tech , Assistant Professor in Department of Computer Science and and Engineering
C ByreGowda Institute of Technology,
Kolar, India.

***Abstract :*** User activity logs can be a valuable source of information in cloud forensic investigations; hence, ensuring the reliability and integrity of such logs is crucial. Most existing solutions for secure logging are designed for conventional systems rather than the complexity of a cloud environment. In this paper, we propose the Cloud Log Assuring Soundness and Secrecy (CLASS) process as an alternative scheme for the securing of logs in a cloud environment. In CLASS, logs are encrypted using the individual user's public key so that only the user is able to decrypt the content. In order to prevent unauthorized modification of the log, we generate proof of past log (PPL) using Rabin's fingerprint and Bloom filter. Such an approach reduces verification time significantly. Findings from our experiments deploying CLASS in OpenStack demonstrate the utility of CLASS in a real-world context.

*IndexTerms* - Cloud forensics, Cloud log, Cloud log assuring soundness and secrecy, Cloud security, Proof of past log, Sustainable computing.

## I. INTRODUCTION

Cloud storage, security and privacy are fairly established research areas [1-7], which is not surprising considering the widespread adoption of cloud services and the potential for criminal exploitation (e.g. compromising cloud accounts and servers for the stealing of sensitive data). Interestingly though, cloud forensics [8- 10] is a relatively less understood topic. In the event that a cloud service, cloud server, or client device has been compromised or involved in malicious cyber activity (e.g. used to host illegal contents such as radicalization materials, or conduct distributed denial of service (DDoS) attacks) [11, 12], investigators need to be able to conduct forensic analysis in order to "answer the six key questions of an incident – what, why, how, who, when, and where [13].

Due to the inherent nature of cloud technologies, conventional digital forensic procedures and tools need to be updated to retain the same usefulness and applicability in a cloud environment [14]. Unlike a conventional client device, cloud virtual machines (VMs) can be supported by hardware that might be located remotely and thus would not be physically accessible (e.g. out of the jurisdictional territory) to an investigator.

In addition, VMs can be distributed across multiple physical devices in a clustered environment or they can exist within a pool of VMs on the same physical components. Therefore, seizing the machine for forensic analysis is not viable in most investigations. Furthermore, data residing in a VM may be volatile and could be lost once the power is off or the VM terminates. Hence, the cloud service provider (CSP) plays a crucial role in the collection of evidential data (e.g. cloud user's activity log from the log). For example, the CSP writes the activity log (cloud log) for each user. Thus, preventing modification of the logs, maintaining a proper chain of custody and ensuring data privacy is crucial [15]. This research considers "activity log data" as any recorded computer event that corresponds to a specific user. Such data must be maintained confidentially to preserver user privacy and to facilitate potential investigative activities.

In 2016, Zawoad et al. proposed a secure logging service called "SecLaaS" [16] that is designed to collect data from one or more log sources, parse the data and then store the parsed data in persistent storage in order to mitigate the risk associated with data volatility. Prior to the storing of data, it encrypts the log and generates a log chain to achieve confidentiality and integrity respectively. SecLaaS encrypts the log(s) using the investigating agency's public key and stores the encrypted log(s) in a cloud server. This ensures privacy and confidentiality of the cloud user, unless the particular user is subject to an investigation (e.g. via a court order). To facilitate log integrity, SecLaaS generates proof of past log (PPL) with the log chain and publishes it publicly after each predefined epoch. A trust model was also suggested that stores the PPL in other clouds to minimize the risk of a malicious cloud entity altering the log. However, in SecLaaS, it is difficult to ensure or verify that the CSP is writing the correct information to the log, or that any information pertinent to the investigation is not omitted or modified. Specifically, SecLaaS does not provide the user the ability to verify the accuracy of the log (since the log is encrypted with the agency's public key). In other words, SecLaaS has limitations in addressing accountability and transparency enforced, especially from the perspective of the user

## II. LITERATURE SURVEY

Reliable cloud logs play a crucial role in forensic investigations. Log acquisition, maintaining confidentiality, integrity and forward secrecy, validity verification and accessibility by investigators (or another authorized party), are some of the many different dimensions a forensic investigator and researcher should pay attention to. Anwar et al. [20] attempted to address some of these challenges by identifying the possibility of Syslog or snort log to assist in the detection of cloud attacks. The authors conducted cloud forensic investigations based on the logs generated by Eucalyptus, an open-source cloud computing software. Specifically, they generated their own dataset by simulating a DDoS attack on Eucalyptus and identified the attacking machine IP address by analyzing the log. Security, access control, and verification of log were not considered. Patrascu and Patriciu [21] proposed a forensic module to be maintained as part of the cloud's controller module, which is designed to communicate with a different stack of cloud assets (e.g. virtual file system, virtual memory, network stack, and system call interface) in order to collect and store logs. Similarly, security of the collected log (either in transmission or in storage) was not considered.In an already compromised system (e.g. after an adversary has obtained access to the cloud server's secret key), it is too late to cryptographically ensure that the unsecured log data has not been altered. Bellare and Lee introduced the notion of forward integrity [22] or forward secrecy as a system property to mitigate an attacker's capability to contaminate a logging system without detection. Contamination includes insertion of false logs, modification or deletion of existing logs, and reordering of logs. Forward integrity is established using a cryptographically strong one-way hash function (i.e. HMAC) and a secret key that serves as an initial point of a chain of a pseudo-random function (PRF). In other words, each successive log entry has an associated hash key that is dependent upon the previous entry (similar in functionality to a blockchain).Schneier and Kelsey proposed a log management scheme [23] based on forward integrity and provided several real- world example applications. Unlike Bellare and Lee, the forward integrity property in the approach of Schneier and Kelsey is ensured using a secret key which is the initial point of a one-way hash chain and message authentication code rather than PRF. The basis of both schemes relies on the fact that, keeping a small and secret piece of information with each log entry which cannot be generated without a secret key, and this secret key changes with each new log. With this secret information, a log entry can be verified later on for its integrity. However, such schemes [22, 23] require the presence of an online trusted server to maintain the secret key and to verify its integrity. To remove the need for an online trusted server, Holt proposed using public key cryptography [24]. Specifically, instead of using the one-way hash function, the author used a digital signature or identity-based signature and in lieu of private keys residing in the trusted server, the author proposed encrypting with public key cryptography and keeping the encrypted keys with the log entries. A known limitation of public key cryptography is the associated computational overheads; thus, it was proposed to use an elliptic curve cryptosystem. However, these approaches do not consider privacy protection from an honest but curious logger, which is the baseline adversary model typically used in the security literature. There are other attacks and security properties that need to be considered, such as the truncation attack and delayed detection challenge. In a truncation attack, an attacker truncates some log entries without detection, and the delayed detection problem occurs when the contamination of a log continues until someone detects such contamination. To address both truncation attack and delayed detection challenge, Ma and Tsudik proposed the concept of forwarding secure sequential aggregate (FssAgg) [25]. In this approach, two tags (known as FssAgg tags) are associated with each log entry, namely: one is for the semi-trusted log accumulator and other is for the trusted verifier. Using FssAgg, they were able to ensure forward secure stream integrity instead of forwarding security. Also, an FssAgg tag can ratify any log prior to it in a certain epoch; thus, the last FssAgg tag of an epoch can `testify' the entire chain of log entries up to that epoch. This, however, incurs additional computation costs during the verification phase.

## III. THREAT MODEL

In this section, we will describe some definitions required to understand our scheme, the threat model, an attacker's capability, possible attacks [29] , and the standard security properties that a secure cloud logging system must possess. A summary of notations used in this paper is presented in Table 1.

Table 1. summary of notations

| | |
|---|---|
| Log | Log can be network log, process log, registry log, application log or any customized text that meets the requirement of being stored for investigation purpose. |
| Log Chain (LC) | LC is a small piece of information that co- exists with its corresponding log in order to maintain the integrity and to prevent any modification of the log (such as addition, modification, deletion, and reordering). |
| Proof of Past Log (PPL) | PPL is a signature or information about the actual log that will be available publicly for forwarding secrecy [22]. That means if the system is compromised, an attacker cannot change the log without detection. PPL can be used to establish log veracity. |
| Cloud Service Provider (CSP) | CSP is a cloud service provider in which a user can rent and use computing and storage resources. We assume that a CSP is honest but curious [30]. That means it will serve according to contract agreement but has a curiosity about client activity. We design our (CLASS) scheme to include features to prevent a dishonest CSP. |
| User | User is a CSP client. |
| Investigator | An investigator is an individual or entity with legal authority to conduct investigative activities in response to some event. These activities include accessing and assessing the contents of log files supplied by a CSP. It is possible for an investigator to collude with a malicious user or CSP to manipulate the perception of an event. |
| Auditor | An auditor is an individual or entity who is authorized to verify the integrity of log entries, typically through techniques such as PPL. It is assumed that the auditor is always fully trusted. |

Our scheme is designed based on the "trust no one" policy. Any party among the CSP, investigator, and user, should be capable of protecting its own security and privacy against another party or collusion between other parties. Potential challenges to designing forensic enabled cloud logging have been discussed in a number of previous studies [16, 26, 31, 32]. For example, in an insecure cloud logging model, only the CSP can write to a log. An investigator or user can collude with the CSP to modify a log before or after publishing PPL. Thus, if a CSP falsely alters a log, whether in collusion with a malicious user or investigator or not, it can hinder the investigative process and conceal the truth of an event. This could result in an attacker failing to be identified or, more dangerously, attributing the attack to the wrong entity.

Conversely, as the cloud is the host of multiple users, a malicious user can repudiate the log under investigation as his/her own log which can lead the criminal lawsuit to be dismissed. On the other hand, the log contains secretive data of the user and the user's privacy may be vulnerable due to this fact. A malicious investigator can alter the log before presenting to court authorities. Moreover, in collaboration with dishonest CSP or CSP employee(s), the investigator can violate the privacy of the user. Based on the above discussion, possible attacks on secure cloud log are given below:

**Modification of Log**: A dishonest CSP can modify the log before or after publishing its proof (PPL) upon or beyond collusion with the user or investigator. A malicious investigator may alter the log before presenting to court to save a dishonest user or to frame an honest user. Modification of a log can be of many forms, such as insertion of invalid entries, removal of the crucial entries, changing existing entries, reordering log entries to mislead the investigation and to hide malicious activities.

**Privacy Violation**: Leakage of a log file can reveal information that is able to be directly linked to a users' identity or is able to aggregate in such a way as to create such a link. Even with cryptographic security, cloud employees can transfer the log to an entity that has the key to decrypt (i.e. an investigator) and thus privacy violation may take place.

**Repudiation of Ownership of Log**: Cloud servers host many users. This presents the possibility for a malicious cloud user to repudiate that the log files under investigation represent the activity of another user. On the other hand, a CSP can repudiate that it did not write the log under investigation. Likewise in SecLaaS, the CSP writes a log for every user and the user has no visibility regarding log entries. This may raise user suspicions regarding log veracity and credibility.

## IV. THE PROPOSED SCHEME

In this section, we improve on SecLaaS and present CLASS scheme. We are assuming that in a cloud infrastructure, no party is trusted, that means an attack can come from any party: a CSP, user, or investigator. We are also assuming that cryptographic primitives work properly (i.e. if someone encrypts a message, then nobody can decrypt it without knowing the secret key).

### SYSTEM OVERVIEW

A dishonest cloud user can attack a system outside the cloud. They can also attack any application deployed in the same cloud or an attack can be launched against a node controller which controls all the cloud activities. For a virtual machine (VM), CLASS scheme (Fig. 1) takes the log from the node controller (NC), hides its content, and stores it in a database. This allows logs to become available for further investigation despite VM shutdown. Moreover, CLASS publishes its proof so that log integrity can be protected and admissibility ensured.

### SYSTEM DETAILS

Before a detailed examination of the proposed scheme, the following definitions and notations are provided:

- $M_1||M_2$: concatenation between two messages $M_1$ and $M_2$.
- H(m): collision-resistance one-way hash function
  of message m.
- $E_{PK}(m)$: encryption of message m with the public key (PK).
- $Signature_{SK}(m)$: signature of message m with private key SK.
- $E_K(m)$: encryption of message m with symmetric key K.

We presume that the cloud service provider (CSP), law enforcement agency (LEA), and user set up their necessary public/private key pairs and publish public keys. $PK_C$ and $SK_C$ are the public and private keys of the CSP respectively and $PK_A$ and $SK_A$ are the public and private keys of the LEA (or auditor). $PK_U$ and $SK_U$ are the public and private keys of a particular cloud user. $PK_U$ and $SK_U$ are titled together as CC-key because they are used to conceal log content of a particular user. During a user's subscription time, the CSP and user mutually generate a CC-key for the user and the CSP only keeps the public portion of the CC- key. The private key ($SK_U$) of the CC-key is shared among multiple clouds using Rabin's or Blakley's secret sharing scheme.

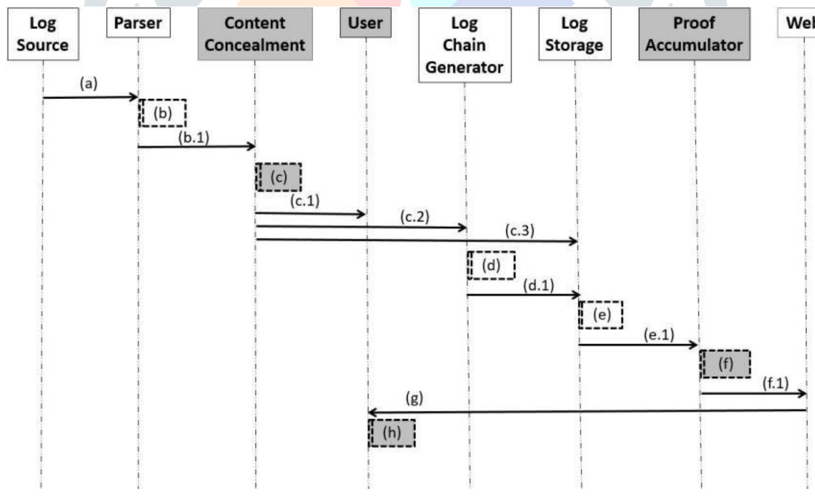

fig. 1. detailed seclaas scheme process

### PRESERVATION OF LOG & ITS PROOF

Fig. 3 depicts details of CLASS from log retrieval to proof of past log (PPL) publication. Modified portions are shaded in grey. For illustration, we have chosen the network log but it can be any log of the system. Details of our scheme are given in the following step-by-step list:

a. Parser collects the log from log source. For example, we collect log from log file stored in a specific directory. When a log file changes (i.e. new lines append) it triggers the parser to check the change and to start processing new log.

Retrieving log from log source, the parser parses the log and gets the necessary information. For the time being there

b. is no standard format of a log and this is another challenge of cloud forensics [26]. The how, when, where, and who, aspects of logs are not yet standardized. Our goal is to keep log content secure given a parser that will provide the system a log message in string format, regardless of content. The format of the log is out of the scope of this work. For our example, we choose originating IP (FromIP), destination IP (ToIP),

user id (UserID) and log time ($T_L$). With this information, the parser generates log entry (LE).

$$LE = <FromIP, ToIP, UserID, T_L>$$

c. Asymmetric encryption of log with individual user's public key $PK_U$ is computed to conceal user's content. This helps to resolve the problem mentioned in section 4.2.1. If we use public key encryption with investigator's public key $PK_A$ and store the log database in CSP's custody then neither cloud employee nor investigator can violate the privacy of user alone. But if a malicious cloud employee provides a portion or full copy of the log database to an investigator then the user's privacy is in jeopardy. Asymmetric encryption with (individual) user's private key addresses this problem, though in turn leads to a potential concern of recovery of the log in the case of an investigation because privacy is provided with user's secret key. This problem can be solved using Shamir or Blackley's secret key sharing scheme and is discussed in the following "secret key sharing" section (section 5.6). After content concealment, the CLASS scheme will generate an encrypted log entry (ELE) and this ELE will be available to the user so that user can cross check if CSP is writing a correct log entry. Because searching through encrypted data is expensive, we keep some data in plain text format for filtering purposes [33, 34, 36].

$$ELE = \langle EP_{Ku}(ToIP \,||\, UserID), FromIP, T_L \rangle$$

d. After that, log chain (LC) is created in order to protect the integrity of the log and prevent potential manipulation. CLASS creates LC using a hash function with current ELE and previous LC:

$$\mathbf{LC = \langle H(ELE \,||\, LC_{Previous}) \rangle}$$

e. At this stage, the payload is ready to be stored in the database. CLASS generates database log entry (DBLE) with ELE and LC and stores it in the log database.

$$DBLE = \langle ELE, LC \rangle$$

For each DBLE, the CLASS scheme requires the generation of proof of past log (PPL) which is then made publicly available. CLASS generates the PPL in a manner that is designed to minimize the usage of memory space. In CLASS, we propose to generate PPL in a batch of the logs of a certain period or a certain amount of logs (e.g. n number of logs). At the end of each epoch, for each DBLE in a batch, CLASS concatenates each of the logs in a chain of logs in chronological order, derives the fingerprint, FP using Rabin's fingerprint [19]. Then the CLASS scheme constructs an accumulator entry (AE) which is bloom filter membership information of the fingerprint FP. For each static IP, CLASS retrieves accumulator entry (AE), epoch time $T_E$, a signature using CSP's private key and concatenates AE, $T_E$, with its signature, generating PPL.

## V. CONCLUSION

In this paper, we proposed a secure logging scheme (CLASS) for cloud computing with features that facilitate the preservation of user privacy and that mitigate the damaging effects of collusion among other parties. CLASS preserves the privacy of cloud users by encrypting cloud logs with a public key of the respective user while also facilitating log retrieval in the event of an investigation. Moreover, it ensures accountability of the cloud server by allowing the user to identify any log modification. This has the additional effect of preventing a user from repudiating entries in his own log once the log has had its PPL established. Our implementation on OpenStack demonstrates the feasibility and practicality of the proposed scheme. The experimental results show an improvement in efficiency thanks to the features of the CLASS scheme, particularly in verification phase. Potential future extensions include the following:

1. Normally logs are low-level data and hard for the common user to understand what exactly those logs signify. Thus, we will explore leveraging big data techniques to facilitate user retrieval and visualization of information from log data. Standardization of log format is also an associated research area.
2. To ease searching, we kept some crucial and sensitive information in plaintext format. This makes them vulnerable to be exposure. Thus, designing secure and efficient searchable encryption would extend this work.
3. There is also the need for an online credibility system designed to develop trust and credibility of a cloud user so that the CSP can enable stricter auditing policies for low-trust users in comparison to high-trust users.
4. Designing and implementing a prototype of the proposed scheme in collaboration with a real- world CSP, with the aim of evaluating its utility (e.g. performance and scalability) in a real-world environment.

## REFERENCES

*[1] X. Liu, R. H. Deng, K.-K. R. Choo, and J. Weng, "An efficient privacy-preserving outsourced calculation toolkit with multiple keys," IEEE Transactions on Information Forensics and Security, vol. 11, pp. 2401-2414, 2016.*

[2] Y. Mansouri, A. N. Toosi, and R. Buyya, "Data storage management in cloud environments: Taxonomy, survey, and future directions," ACM Computing Surveys (CSUR), vol. 50, p. 91, 2017.

[3] M. Tao, J. Zuo, Z. Liu, A. Castiglione, and F. Palmieri, "Multi-layer cloud architectural model and ontology-based security service framework for IoT-based smart homes," Future Generation Computer Systems, vol. 78, pp. 1040-1051, 2018.

[4] Z. Xia, Y. Zhu, X. Sun, Z. Qin, and K. Ren, "Towards privacy-preserving content-based image retrieval in cloud computing," IEEE Transactions on Cloud Computing, pp. 276-286, 2018.

[5] L. Zhou, Y. Zhu, and A. Castiglione, "Efficient k-NN query over encrypted data in cloud with limited key-disclosure and offline data owner," Computers & Security, vol. 69, pp. 84-96, 2017.

[6] Q. Alam, S. U. Malik, A. Akhunzada, K.-K. R. Choo, S. Tabbasum, and M. Alam, "A Cross Tenant Access Control (CTAC) Model for Cloud Computing: Formal Specification and Verification," IEEE Transactions on Information Forensics and Security, vol. 12, pp. 1259-1268, 2017.

[7] L. Li, R. Lu, K.-K. R. Choo, A. Datta, and J. Shao, "Privacy-preserving-outsourced association rule mining on

vertically partitioned databases," IEEE Transactions on Information Forensics and Security, vol. 11, pp. 1847-1861, 2016.

[8] K.-K. R. Choo, M. Herman, M. Iorga, and B. Martini, "Cloud forensics: State-of-the-art and future directions," Digital Investigation, pp. 77- 78, 2016.

[9] C. Esposito, A. Castiglione, F. Pop, and K.-K. R. Choo, "Challenges of Connecting Edge and Cloud Computing: A Security and Forensic Perspective," IEEE Cloud Computing, vol. 4, pp. 13-17, 2017.

[10] Z. Qi, C. Xiang, R. Ma, J. Li, H. Guan, and D. S. Wei, "ForenVisor: A tool for acquiring and preserving reliable data in cloud live forensics,"