International Journal of Engineering Technology Science and Research
IJETSR
www.ijetsr.com
ISSN 2394 – 3386
Volume 6, Issue 7, July 2019

CLASS: CLOUD LOG ASSURING SOUNDNESS AND SECRECY SCHEME FOR CLOUD FORENSICS

BADUGU RAJESWARI 1*, BADUGU SUNEEL KUMAR 2
1*Computer Science and engineering, Nova College of engineering and technology, Jupudi
2 Computer Science and Engineering, Nova College of engineering and technology, Jupudi

***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-worldcontext.*

Index Terms— Cloud forensics, Cloud log, Cloud log assuring soundness and secrecy, Cloud security, Proof of past log, Sustainable computing

I. INTRODUCTION

C LOUD 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.

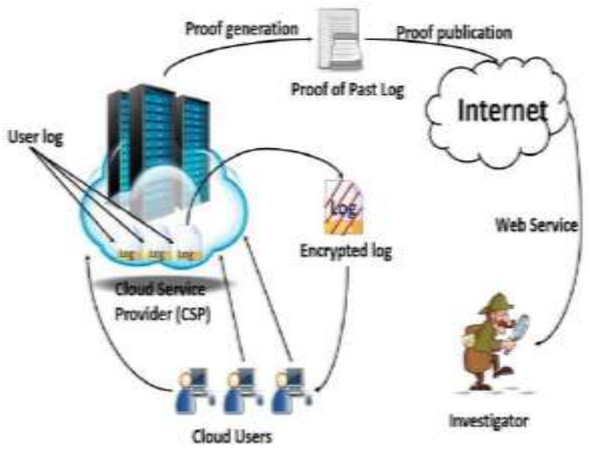


Fig. 1. Overview of CLASS scheme process

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.

Extending SecLaaS, we propose a secure cloud logging scheme, Cloud Log Assuring Soundness and Secrecy (CLASS), designed to ensure CSP accountability (i.e. writing the correct information to the log) and preserve the user's privacy – i.e. our contribution in this paper. Specifically, we include the capability for the user to verify the accuracy of their log. To do this, the log will be encrypted using the user's public key (rather than the agency's public key). To avoid introducing unnecessary delays to the forensic investigation, during user registration with the cloud service, both the CSP and the user will collectively choose a public/private key pair referred to as content concealing key (CC-key) for the user. The corresponding (content concealing) private key will be shared with other CSPs using Shamir's [17] or Blakley's [18] secret sharing schemes. This would allow the private key to be regenerated whenever necessary. We also demonstrate how we can leverage Rabin's fingerprint [19] and bloom filter in PPL generation to establish log veracity. We then implement CLASS in OpenStack and evaluate its performance.

II. RELATED WORK

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 approved party), area unit a number of the numerous totally different dimensions a rhetorical investigator and scientist ought to listen to. Anwar et al. [20] tried to deal with a number of these challenges by distinctive the chance of Syslog or snort log to help within the detection of cloud attacks. The authors conducted cloud rhetorical investigations supported the logs generated by Eucalyptus, associate ASCII text file cloud computing code. Specifically, they generated their own dataset by simulating a DDoS attack on Eucalyptus and known the offensive machine informatics address by analyzing the log. Security, access management, and verification of log weren't thought of. Patrascu and Patriciu [21] planned a rhetorical module to be maintained as a part of the cloud's controller module, that is meant to speak with a unique stack of cloud assets (e.g. virtual filing system, computer memory, network stack, and call interface) so as to gather and store logs. Similarly, security of the collected log (either in transmission or in storage) wasn't thought of.

In associate already compromised system &#40;e.g. when associate person has obtained access to the cloud server's secret key&#41;, it's too late to cryptographically make sure that the unsecured log information has not been altered. Bellare associated Lee introduced the notion of forward integrity [22], [23] or forward secrecy as a system property to mitigate an attacker's capability to contaminate a work system while not detection. Contamination includes insertion of false logs, modification [24], [25] or deletion of existing logs, and rearrangement of logs. Forward integrity is established employing a cryptographically sturdy unidirectional hash operate (i.e. HMAC) associated a secret key that is an initial purpose of a series of a pseudo-random operate (PRF). In alternative words, every sequential log entry has associate associated hash key that's dependent upon the previous entry (similar in practicality to a blockchain).

There area unit different attacks and security properties that require to be thought of, like the truncation attack and delayed detection challenge. During a truncation attack, associate degree aggressor truncates some log entries while not detection, and also the delayed detection drawback happens once the contamination of a log continues till somebody detects such contamination. to handle each truncation attack and delayed detection challenge, Ma and Tsudik planned the thought of forwarding secure consecutive combination (FssAgg) [25]. During this approach, 2 tags (known as FssAgg tags) area unit related to every log entry, namely: one is for the semi-trusted log accumulator and different is for the trustworthy champion. Victimization FssAgg, they were ready to guarantee forward secure stream integrity rather than forwarding security. Also, associate degree FssAgg tag will validate any log before it during a sure epoch; so, the last FssAgg tag of associate degree epoch will `testify' the complete chain of log entries up thereto epoch. This, however, incurs extra computation prices throughout the verification part.

THREAT MODEL &amp; SECURITY PROPERTIES

In this section, we are going to describe some definitions needed to know our theme, the threat model, associate degree attacker's capability, doable attacks [29], and also the normal security properties that a secure cloud work system should possess. An outline of notations utilized in this paper is conferred in Table one.

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

Accumulator style

Zawoad et al. [37] used Bloom filter as a symbol of past information possession, that is fails to account for Bloom filter's inherent potential for false positives. Once employing a Bloom filter technique, there's a trade-off between the quantity of false positives and also the size of the filter. To mitigate this drawback, a science unidirectional accumulator may well be used. However, this needs important procedure overhead. In SecLaaS, they used their own arrangement Bloom Tree that reduced range|the amount|the quantity} of false positive incidents however needs associate inflated number of instances of logs and important procedure resources at verification time. This can be true notwithstanding the amount of entries being verified. Additionally, it still remains liable to false positives (albeit reduced).

CLASS proposes a bloom filter based mostly PPL that computes membership data of (Rabin's) fingerprint [19] of chronologically ordered log chain of associate epoch. It needs one single bloom filter for a whole batch of logs. The fingerprint includes a total reflection of the complete log chain during a specific epoch. for instance, if there ar L logs in associate epoch and log chains of those logs in written account order ar LC1, LC2, LC3, ..., LCL. Fingerprint FP of those log chains is,

FP = FingerPrint(LC1 | | LC2 | | LC3 | | .... | | LCL)

Then a bloom filter with membership information of FP is accumulator entry (AE) for this epoch of log. If the hash function for generating bloom filter information are: hash1, hash2, hash3, hashn.

v1= hash1(FP)

v2= hash2(FP)

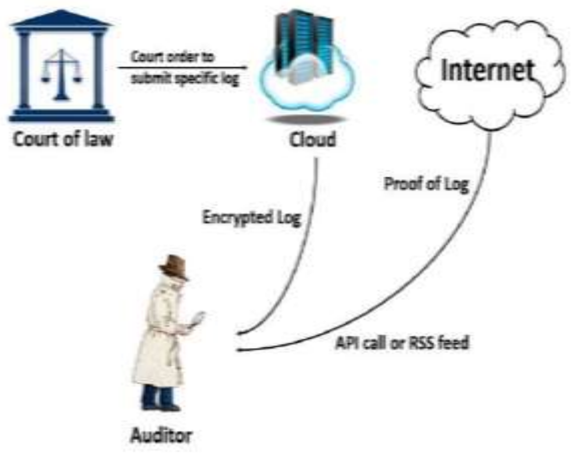
v3= hash3(FP)

... ...

vn= hashn(FP)

A bloom filter with one at v1th, v2th, v3th, ..., vnth positions and zero at remainder of the positions represents accumulator entry, AE. Consequently, PPL for this epoch of log could be a combination of accumulator entry (AE), epoch time (TE) and signature of this data by CSP.

PPL = < AE, TE, SignatureCSP(AE, TE) >



III. SECLAAS

Specification

The SecLaaS theme presents associate economical and sturdy procedure for cloud log management, from collection logs to making their proof of truthfulness in public accessible. At the same time, it preserves their integrity, privacy, and forward secrecy. SecLaaS preserves the encrypted log in storage then generates and publishes its proof so no party will modify it.

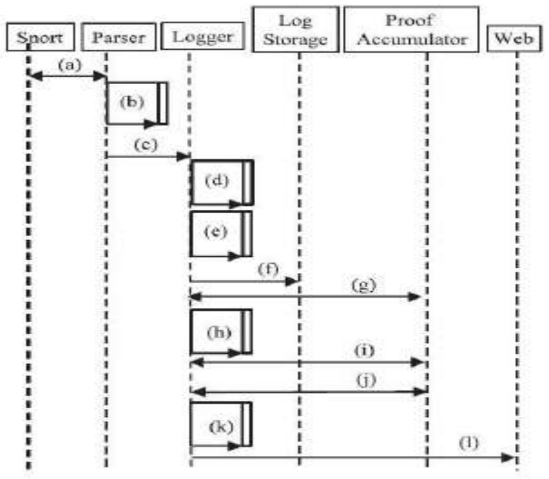


Fig. 2. Detailed SecLaaS scheme process

Now, we present SecLaaS scheme:

a) Parser module communicates with log source to get activity log.
b) The parser parses the log and generates Log Entry, LE. LE can be any useful information required for proper investigation. SecLaaS considers, for sake of demonstration, originating IP address (FromIP), destination IP address (ToIP), time of log (TL), port (Port) and user ID (UserID).
c) LE = <FromIP, ToIP, TL, UserID>
d) Parser sends the LE to logger module for further processing.
e) To preserve the privacy of the user, logger module encrypts some part of LE with law enforcement authority's public key, Pa. Because searching through encrypted data is computationally expensive [33-35], another part of LE is kept unencrypted. Thus, encrypted log entry is generated in this phase.
f) ELE = <EPa(ToIP, Port, UserID), FromID, TL>

IV. EXISTING SYSTEM

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 acloud 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.

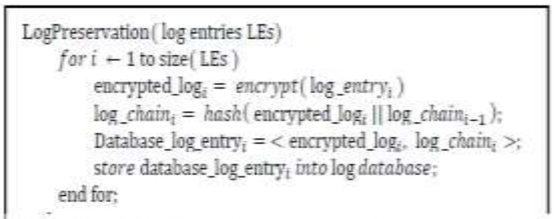
V. PROPOSED SYSTEM

By Extending SecLaaS, we propose a secure cloud logging scheme, Cloud Log Assuring Soundness and Secrecy (CLASS), designed to ensure CSP accountability (i.e.writing the correct information to the log) and preserve the users privacy–i.e. our contribution in this paper. Specifically, we include the capability for the user to verify the accuracy of their log. To do this, the log will be encrypted using the user's public key (rather than the agency's public key). To avoid introducing unnecessary delays to the forensic investigation, during user registration with the cloud service, both the CSPand the user will collectively choose a public/private key pair referred to as content concealing key (CC-key) for the user. The corresponding (content concealing) private key will be shared with other CSPs using Shamir's [17] or Blakley's [18] secret sharing schemes. This would allow the private key to be regenerated whenever necessary. We also demonstrate how we can leverage Rabin's fingerprint [19] and bloom filter in PPL generation to establish log veracity. We then implement CLASS in OpenStack and evaluate its performance.

Algorithm

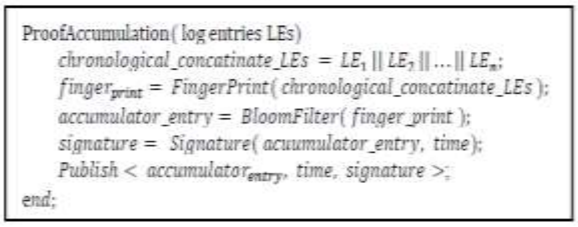
1. CLASS algorithms: can be categorized into two major groups: One for Log Preservation and one for Proof Accumulation. The Log Preservation algorithm can take log entries individually or in a batch and performs processing prior to storage in a log database. This algorithm encrypts for secrecy and generates hash digest for consistency. The Proof Accumulator algorithm performs daily processing of all log entries corresponding to an IP address to prepare and publish proof of past log (PPL).

2. Shamir's secret sharing algorithm: Small information will be extracted from private key following Shamir's secret sharing algorithm and each portion will be shared to one CSP. After getting secret portions of a particular user, the host cloud can reconstruct the private key to decrypt the log of that user using Shamir's secret sharing algorithm.



```
LogPreservation( log entries LEs)
    for $i \leftarrow 1$ to size( LEs )
        $encrypted\_log_i = encrypt( log\_entry_i )$
        $log\_chain_i = hash( encrypted\_log_i \,||\, log\_chain_{i-1} );$
        $Database\_log\_entry_i = < encrypted\_log_i,\ log\_chain_i >;$
        store database_log_entry$_i$ into log database;
    end for;
```

Algorithm 1. LogPreservation pseudocode for processing log entries



```
ProofAccumulation( log entries LEs)
    $chronological\_concatinate\_LEs = LE_1 \,||\, LE_2 \,||\, \ldots \,||\, LE_n;$
    $finger_{print} = FingerPrint( chronological\_concatinate\_LEs );$
    $accumulator\_entry = BloomFilter( finger\_print );$
    $signature = Signature( acuumulator\_entry,\ time);$
    $Publish < accumulator_{entry},\ time,\ signature >;$
end;
```

Algorithm 2. ProofAccumulation pseudocode to generate and publish proof of past log (PPL)

PROPOSED SCHEME: category

In this section, we have a tendency to improve on SecLaaS and gift category theme. We have a tendency to area unit presumptuous that in an exceedingly cloud infrastructure, no party is trusty, which means associate attack will come back from any party: a CSP, user, or investigator. We have a tendency to be presumptuous that scientific discipline primitives work properly (i.e. if somebody encrypts a message, then no one will rewrite it while not knowing the key).

System summary A dishonest cloud user will attack a system outside the cloud. They'll additionally attack Associate in Nursing application deployed within the same cloud or an

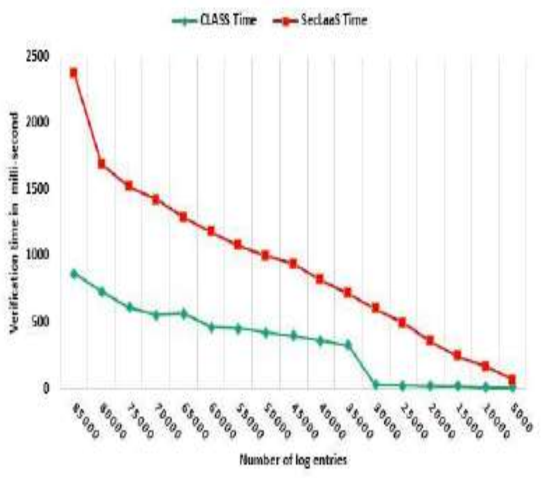
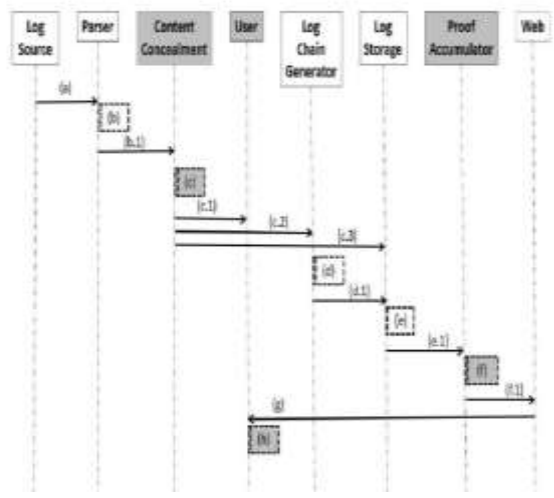


Fig. 7. Verification time comparison

attack will be launched against a node controller that controls all the cloud activities. For a virtual machine (VM), category theme (Fig. 1) takes the log from the node controller (NC), hides its content, and stores it in a very info. This enables logs to become accessible for more investigation despite VM conclusion. Moreover, category publishes its proof in order that log integrity will be protected and acceptability ensured.



Verification Time

In the case of testifying proof of log, we have a tendency to contemplate 5000 to 85000 log entries at every epoch with associate increment of 5000 log entries at every step. In CLASS, we have a tendency to concatenate every log entries of associate epoch and generate the fingerprint. Later, this fingerprint's membership during a bloom filter is calculated and published: for one epoch one fingerprint and one bloom filter. During this case, fingerprint reflects every log entry of associate epoch and bloom filter's accuracy improves as only 1 member resides during a bloom filter. By comparison, SecLaaS generates bloom tree that may be a multi-level bloom filter and inserts every log entry individually into bloom filter. Thus, SecLaaS' verification time is longer than that of CLASS'.

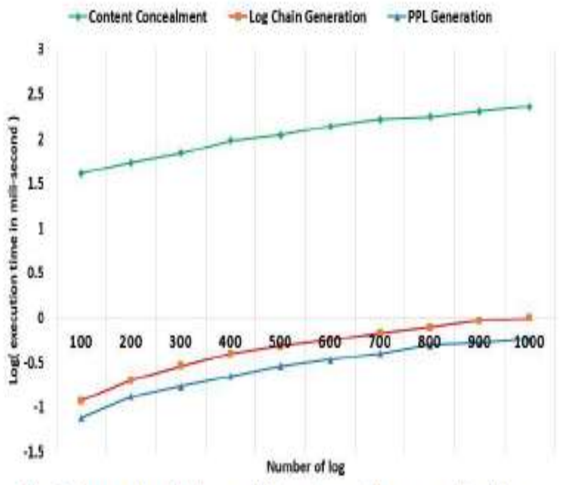


Fig. 8. Comparison between different operations execution time

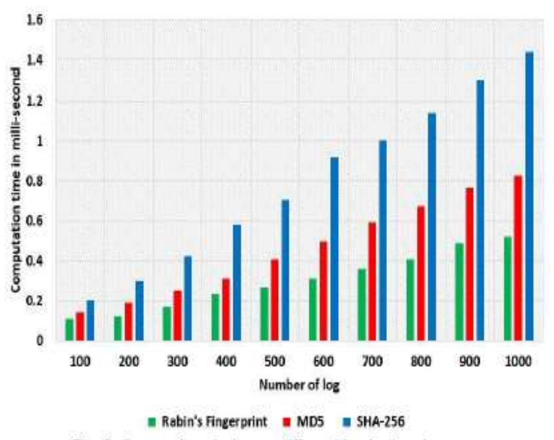


Fig. 9. Comparison between different hash digest

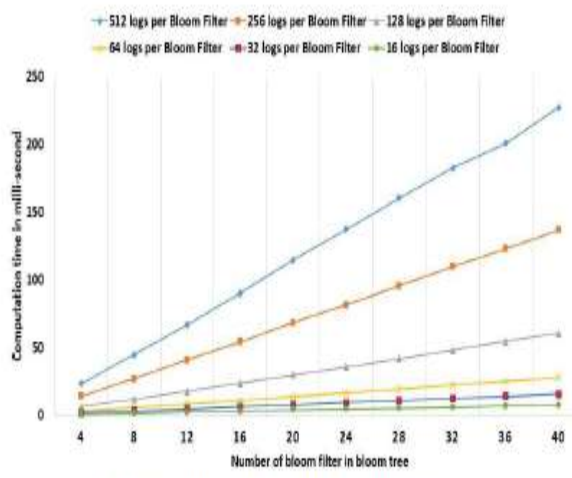


Fig. 10. Different Bloom Tree comparison

VI.CONCLUSION

In this paper, we proposed a secure logging scheme (CLASS) for cloud computing with features that facilitate the preservation of user privacyand that mitigate the damaging effects ofcollusion amongother parties. CLASS preserves the privacy of cloud usersby encrypting cloud logswith 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 auser from repudiating entries in his own log once the log has had its PPL established. Our implementation on OpenStack demonstrates the feasibility andpracticality oftheproposed scheme. The experimental results show an improvement in efficiency thanks to the features of the CLASSscheme, particularlyin verification phase.

VII. RESULT

In this section, new future analysis directions within the context of CLF square measure bestowed. However, CLF remains in its early stage of the analysis to supply ample opportunities for each technical and economic future work to mitigate the challenges associated with its paramount log management.

Every future direction as shown in Figure five can bring the focus of academician, businessman, vendors, and CSPs to analysis out profound solutions for CLF in creating them applicable at intervals cloud computing. Cloud Log Forensics Apis. At present, cloud computing provides completely different Apis to help shoppers to move with cloud resources for various services together with storage and computation. However, CLF lacks standardized Apis to help investigators in accessing cloud log knowledge for analyzing malicious events that occurred at the time of the attack. In Patrascu and Patriciu [2014], cloud forensics API is projected, which is used to collect log knowledge from the VM within the virtualization layer. The cloud forensics API Bridge between the investigator and therefore the monitor VM for a particular quantity of your time to collect completely different logs. The projected cloud forensics API lacks the power to supply log knowledge between completely different VMs, which can be very important for VM-side channel attacks. Therefore, it's necessary to develop distinctive and secure Apis for CLF to supply straightforward and secure interfaces for investigators to research cloud log knowledge at intervals and out of doors the cloud. Conversely, if Apis don't seem to be properly developed, inflicting vulnerabilities, then this will have an effect on all of CLF by harming cloud log knowledge whereas having spurious access to that.

VIII. 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] M. Abd-El-Malek, G. R. Ganger, G. R. Goodson, M. K. Reiter, and J. J. Wylie, "Fault-Scalable Byzantine Fault-Tolerant Services," in *ACM Symposium on Operating Systems Principles (SOSP)*, 2005, pp. 59–74.

[8] M. K. Aguilera, R. Janakiraman, and L. Xu, "Using Erasure Codes Efficiently for Storage in a Distributed System," in *International Conference on Dependable Systems and Networks (DSN)*, 2005, pp. 336–345.

[9] W. Aiello, M. Bellare, G. D. Crescenzo, and R. Venkatesan, "Security amplification by composition: The case of doublyiterated, ideal ciphers," in *Advances in Cryptology (CRYPTO)*, 1998, pp. 390–407.

[10] C. Basescu, C. Cachin, I. Eyal, R. Haas, and M. Vukolic, "Robust Data Sharing with Key-value Stores," in *ACM SIGACTSIGOPS Symposium on Principles of Distributed Computing (PODC)*, 2011, pp. 221–222.

[11] A. Beimel, "Secret-sharing schemes: A survey," in *International Workshop on Coding and Cryptology (IWCC)*, 2011, pp. 11–46.

[12] A. Bessani, M. Correia, B. Quaresma, F. André, and P. Sousa, "DepSky: Dependable and Secure Storage in a Cloud-ofclouds," in *Sixth Conference on Computer Systems (EuroSys)*, 2011, pp. 31–46.

[13] G. R. Blakley and C. Meadows, "Security of ramp schemes," in *Advances in Cryptology (CRYPTO)*, 1984, pp. 242–268.

[14] V. Boyko, "On the Security Properties of OAEP as an All- ornothing Transform," in *Advances in Cryptology (CRYPTO)*, 1999, pp. 503–518.

[15] R. Canetti, C. Dwork, M. Naor, and R. Ostrovsky, "Deniable Encryption," in *Proceedings of CRYPTO*, 1997.

[16] Cavalry, "Encryption Engine Dongle," http://www.cavalrystorage.com/en2010.aspx/

[17] C. Charnes, J. Pieprzyk, and R. Safavi-Naini, "Conditionally secure secret sharing schemes with disenrollment capability," in *ACM Conference on Computer and Communications Security (CCS)*, 1994, pp. 89–95.

[18] R. L. Rivest, "All-or-Nothing Encryption and the Package Transform," in *International Workshop on Fast Software Encryption (FSE)*, 1997, pp. 210–218.

[19] A. Shamir, "How to Share a Secret?" in *Communications of the ACM*, 1979, pp. 612–613.

[20] D. R. Stinson, "Something About All or Nothing (Transforms)," in *Designs, Codes and Cryptography*, 2001, pp. 133– 138.

[21] StorSimple, "Cloud Storage," http://www.storsimple.com/.

[22] J. H. van Lint, *Introduction to Coding Theory*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 1982.

[23] Wikipedia, "Edward Snowden," http://en.wikipedia.org/wiki/Edward_Snowden#Disclosure.

[24] Z. Wu, M. Butkiewicz, D. Perkins, E. Katz-Bassett, and H. V.Madhyastha, "SPANStore: Cost-effective Geo-replicated Storage Spanning Multiple Cloud Services," in *ACM Symposium.on Operating Systems Principles (SOSP)*, 2013, pp. 292–308.

[25] H. Xia and A. A. Chien, "RobuSTore: a Distributed Storage Architecture with Robust and High Performance," in *ACM/IEEE Conference on High Performance Networking and Computing (SC)*, 2007, p. 44.

Authors Profile

Ms. **Badugu Rajeswari** pursuing Mtech 2nd year in Nova College of engineering and technology, in Department of Computer Science and Engineering, jupudi.

Mr. **Badugu. Suneel Kumar** currently working as an Assistant Professor in Department of Computer Science and Engineering with the Qualification Mtech (Ph. D).