www.ijsrnsc.org

Available online at www.ijsrnsc.org

IJSRNSC

Volume-9, Issue-1, February 2021
Research Paper

Int. J. Sc. Res. in Network Security and Communication

E-ISSN:2321-3256

Multi-Level and Mutual Log Integrity Preservation Approach for Cloud Forensics using Public Key Infrastructure

Siva Rama Krishna Tummalapalli[1*], A.S.N. Chakravarthy[2]

[1,2]Dept. of CSE, JNTUK University College of Engineering Vizianagaram, Vizianagaram, India

[*]*Corresponding Author: tsrkrishna@jntukucev.ac.in*

Received: 21/Jan/2021, Accepted: 05/Feb/2021, Published: 28/Feb/2021

***Abstract*—** The increasing growth of cloud computing makes the forensic investigation complex. It is very significant to collect and preserve the admissible evidences of different actions happened in the cloud environment in spite of collusion problem. The forensic investigation of logs poses a great challenge to make sure that the investigated logs are not tampered and consistent. Various integrity preservation methods are developed to secure the trustworthy evidences of cloud, but storing the logs of user actions and to ensure the trustworthiness with respect to forensic investigation still result a challenging issue in cloud. Hence, an effective multi-level and mutual log integrity preservation method is developed to assure the integrity of log evidence using public key infrastructure. The integrity preservation mechanism is modelled by employing the security mechanisms, such as hashing, encryption, decryption, and walsh transform. The communication between the entities is achieved by generating the keys using the hashing and encryption function with the concatenation operator. However, the public key and the random number is used to generate the message that can be exchanged between the cloud entities. The investigation process is done by the investigator for transferring the encrypted log file to the server. The proposed method achieved higher performance by considering the 500 number of cloud users in terms of detection rate, memory, and time with the values of 0.9828, 2185 bytes, and 7287 ms, respectively.

***Keywords*—** Cloud Security, Cloud Forensics, Evidence Integrity, Log Analysis, Public Key

I. INTRODUCTION

The computing industry is revolutionized with cloud computing as the cloud provides unlimited storage and computing power on the pay-per-use service model. However, the information technology (IT) service expenditure of the business model from the capital expenditure (CapEx) to the operational expenditure (OpEx) that gains more computing services in cloud. Moreover, Bessemer Venture Partner (BVP) cloud index traps public companies and reported that cloud will triple to reach US$500 billion in the capitalization of cloud market by 2020 [1,2]. In 2018, BVP state of cloud industry shows the strong prediction of cloud computing and increases the innovative cloud services, such as payment as a service and serverless computing [3]. Moreover, the right scale of 2017 cloud report states that cloud adoption increases, and the companies started to run majority of their business applications in the cloud environment. Despite that, cloud computing is susceptible to hype and confusion that surrounds it [4-6]. The cloud computing is more attractive such that it is the high performing and cost-efficient model. Cloud environments also suffer from various security issues [7,8] especially in the regard of digital forensics [9-13].

Due to the availability of large volume of storage facilities and computation power at low costs, malicious individuals are motivated to launch attacks from virtual machines in the cloud [14] or to record contraband documents [15-18]. The attackers are placing Linux DDoS Trojan and compromise the Amazon EC2 virtual machines (VMs) to launch the distributed denial of service (DDoS) attacks from VM. In case of such security incidents, it is highly essential to execute the digital forensics process for determining the details of the incident in cloud. Most of the implicit statements defined in the forensic analysis, like physical accessing of hardware are not valid in the cloud model. Cloud forensics is a branch of the digital forensics developed by the researchers aimed at studying/designing methods for gathering evidence of security incidents in cloud environments. The activity logs of users in cloud can explore the actions taken by the consumer with cloud infrastructures. Therefore, logs form crucial evidence for investigating security incidents on cloud. Collecting the logs from cloud environments results a complex task due to the limited control of users and investigators over cloud resources. On many cloud platforms, there exist no way for the investigators to extract the logs from terminated virtual machines such that they must depend on Cloud Service Provider (CSP) for collecting the logs of such VMs. The investigators are required to trust CSP, as there is no guarantee whether the logs provided by CSP are tampered or not. Attackers usually attack the logging system also to cover their tracks [19]. The necessity of logs is indisputable in the forensic investigations such that the trustworthiness of the evidence are remains questionable if proper measures are not considered to secure them from tampering. The adversary may host a botnet server, phishing services, or

spam email server in the cloud VM and eliminate the traces of malicious actions by tampering logs. Sometimes, the investigators also may be malicious and alter log file before submitting it as evidence [13,20,21].

Researchers developed a web-based read-only API to make the log acquisition procedure easy and to achieve effective forensics support [22]. However, this method does not preserve the integrity and confidentiality of logs if the CSP is malicious. Most of the existing methods that preserve the integrity and privacy of logs from the external attackers are not modeled for protecting the integrity while the logger itself is malicious [19,23,24]. However, the traditional forward secure logging method relies on the initial secret and hence the external attacker cannot access. To verify the traditional forward secure chain, investigator requires capturing all logs from the beginning of chain even though, the investigator requires the logs about very days or hours. However, the traditional secure logging methods only protect the logs from being altered by the investigator, as investigator cannot access initial secret. If the investigator colludes with CSP, traditional methods fail to detect the modification done by investigator. These issues are solved in the trustworthy and secure manner to achieve success forensic method based on the logs [13,25].

This paper proposes a multi-level and mutual log integrity preservation approach using public key infrastructure for cloud forensics. Rest of the paper is reviews different existing integrity preservation methods, explains the proposed integrity preservation method, and compares the proposed method with existing integrity preservation methods.

II. LITERATURE REVIEW

In this section, some of the existing log integrity preservation approaches are presented along with their benefits and drawbacks, which motivated the researchers to design the proposed multi-level and mutual log integrity preservation method.

Rane, S. et al. [26] introduced a blockchain and Interplanetary File System (IPFS) model to secure the trustworthy evidence in the cloud framework. Here, the blockchain system was built to record the logs of users' actions and assure the trustworthiness and for recovering the logs in the forensic investigation. The blockchain was used to assure the integrity of trustworthy evidence. The threat driven model was used for analyzing the security of critical systems. However, this method failed to make the cloud forensic model more transparent and secure. Lokhande, P., and Mane, V. [12] introduced a Bloom filter-based T-tree (BR-tree) model to construct the proofs of the past logs. This method was derived by the integration of bloom filter and R-tree nodes. It offered more robust cloud forensic model that helped to reduce the chances of post incidence attack. The requirement of logs from various sources, like databases, process and network was undeniable for the execution of successful investigation in forensics. This method had high computational complexity. Singh, K. D. et al. [27] introduced an integrity and confidentiality preservation model for maintaining the confidentiality and integrity of logs. Here, the cloud provider was responsible to offer all the required logs to investigator in the case of terrorism actions. Here, the logs were securely stored in the persistent storage and the logs were verified before sending to forensic investigator. This method offers the auditor for verifying authenticity of logs in cloud. Zawoad, S. et al. [13] introduced a secure logging as a service (SecLaas) for preserving different logs created for the actions of VM to run in cloud and to preserve the integrity and confidentiality of the logs. This method was effectively used to stores the logs for forensics activities. The integrity of logs was verified with the log chain and past logs. It was practically feasible for integrating the solution with cloud infrastructure.

Dalezios, N. et al. [28] developed a cloud auditing data federation (CADF) model to make the forensic investigation. The CADF logging was used in Openstack and become forensically by modifying the Cloudstack platform. This method was expandable and more robust but failed to utilize the information recorded in CADF logs. Pichan, A. et al. [6] developed a forensic logging model for analyzing the perspective of forensic practioners based on the business requirements. It is the widely available open-source platform that has been activated by validating it for evidence guidelines. It defines the confidence and trust in logs to the significant level and easily interprets with the logs. However, it failed to directly enable the investigator in collecting the logs. Ahmed Khan, M. N., and Ullah, S. [29] developed a log aggregation forensic analysis model in cloud environment to minimize the dependency on the acquisition of logs from service provider. Here, the logs were obtained from various sources, such as VM logs, server-side logs, and client machines. The features were extracted from accumulated logs and the correlation engine was triggered with respect to the attributes. This method achieved lower accuracy in identifying the behavior. Patidar, M., and Bansal, P. [30] developed a log-based model to offer the secure solution to maintain the cloud-based applications. It mainly concentrated on the alerting and monitoring process, forensic process, and troubleshooting. It offered the roadmap to cover the major areas in cloud environment. However, this method was not applicable for log management in the case of large-scale organization.

III. PROPOSED LOG INTEGRITY PRESERVATION METHOD

In the proposed method integrity preservation is taken care in different phases: setup phase, log file creation phase, key management phase, certification and encryption phase, and validation phase. Different entities involved in the proposed method are cloud service customer (CSC), cloud server (CS), cloud service provider (CSP), cloud forensic investigator (CFI), certificate authority (CA) and a cloud log storage system. The user creates the identity (ID) and

password of user and communicate them to the server, which receives the user credentials and stored them in the server database for further processing. The server generates the key factor and thereby it generates the shared key of CSP and CSC. However, the key that is shared between the CSP and CSC is generated by the server, whereas the key used to share the data between CFI and CA is generated by CA, and the key that is enabled to transfer the data among CS and CFI is created by the server, respectively. The server creates the log file using the ID of user, timestamp, ID of CSP and the service offered to the user. Accordingly, the CFI creates their own ID and password and transfer them to CA, which receives the credentials from the investigator and generates the key for transferring the data between CFI and CA, respectively. The CFI creates its own ID and password and store them at CA, which receives the investigator credentials and generate the message for the mutual verification process. After receiving the mutually verified message by CA from the investigator, the CA creates the certificate using walsh transform by considering the polynomial factor in addition to the file associated with the header. The message generated by the server is send to CFI for generating the decrypted shared key and the server verifies the certificate the deliver the encrypted log file with the associated user ID to CFI. The investigation process is carried out by the CFI to determine the success or failure of data transmission.

Figure 1 represents different entities of cloud forensic system model, such as CS, CSC, CSP, CFI, CA, and log file storage system. The cloud entities are connected through the cloud network in such a way that each entity can communicate with other entities by means of the network. Here, the CFI is employed to ensure the trustworthiness of the log evidence by performing digital forensic operations. However, digital forensics is the procedure of collecting, identifying, analyzing, presenting, and recording the crime scene information. The log offers the systematic representation of state of object and actions. It is required to store the log and make it available to investigators for secure logging.

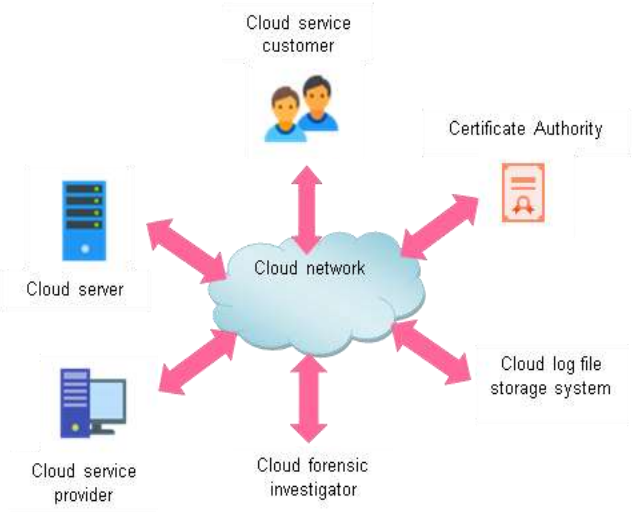

Figure 1. System Model of Cloud Forensics

IV. MULTI-LEVEL AND MUTUAL LOG INTEGRITY PRESERVATION METHOD

The usability of log files for the digital forensics in cloud is explored using the proposed integrity preservation method based on public key infrastructure. The proposed method contains different entities and involves various phases using the security operations, such as encryption, decryption, hashing, and walsh transform. Some of the entities involves in the forensic investigation process are described in the setup phase. In the log file creation phase, the server creates the log file based on the ID of user, timestamp, ID of CSP and the service offered to the user. The third phase is the key management phase, where the shared key between CA and CFI is generated by the CA, whereas the shared key between the server entity and CFI is generated by the server. In the certification and encryption phase, the CA creates the certificate using the polynomial factor, walsh transform, and the file associated with the header. The final phase of the proposed method is the proof of verification phase, where the log file is decrypted, and the process of investigation is carried out by the investigator entity. Table 1 represents different symbols used in the proposed method and their description.

Table 1. Symbols used in proposed multi-level and mutual log integrity preservation method

Symbol	Description
R_{id}	User ID
R_{pwd}	User password
Y	Shared key of CSP and CSC
H	Cryptographic Hash Function
B	Public key
F	Log file
G	Stored log file
X_{id}	CSP ID
E	Encryption
S_{id}	CFI ID
S_{pwd}	CFI password
m	Random number
t	Timestamp
Y_T	Shred key of CFI and CA
Y_S	Shared key of CSC and CFI
Q	Request message
M	Mutual verified message
C	Certificate
A	Encrypted certificate
I	Investigation process
D	Decryption
$\|\|$	Concatenation operation
$\oplus$	Ex-or operation

A. Log file creation

In this phase, the user ID R_{id} and the user password R_{pwd} are generated by the user and communicated to the server, which receives the user credentials and store them as R_{id}^{*} and R_{pwd}^{*}, respectively. After receiving the user credentials, the cloud server verifies whether $R_{id} = R_{id}^{*}$ and $R_{pwd} =$

R^*_{pwd}, and if they matched, the server generates two key factors as y_1 and y_2 to generate the shared key. However, the key factor y_1 generated by the server is represented as,

$$y_1 = \left(H\left(R^*_{id}\right) \| H\left(R^*_{pwd}\right) \oplus E(B)\right) \quad (1)$$

Hashing function is individually applied to the user ID and password, and the hashed result is Ex-ored with the encrypted public key.

$$y_2 = H(t) \oplus E(B) \quad (2)$$

The timestamp is passed to the hashing function and the encryption function is applied to the public key. The Ex-or operation is performed with the hashed result of timestamp and the encrypted public key to generate the key factor y_2.

$$Y = y_1 \oplus y_2 \quad (3)$$

The shared key of CSP and CSC is generated by performing the Ex-or operation using the key factor y_1 and y_2. The key that is shared between the CSC and CSP is referred as the shared key of CSP and CSC, respectively. The key generated by the server is send to service provider and user. When the key shared among the CSP and CSC is matched, the user begins the communication process with that of server.

$$F = \left(R_{id}, t, X_{id}, \alpha\right) \quad (4)$$

The server creates the log file using the parameters of user ID, timestamp, ID of CSP, and the service offered to the user. The log file created by the server is encrypted and the encrypted log file is sent to the user. Figure 2 represents the log file creation phase of proposed multi-level and mutual log integrity preservation method.

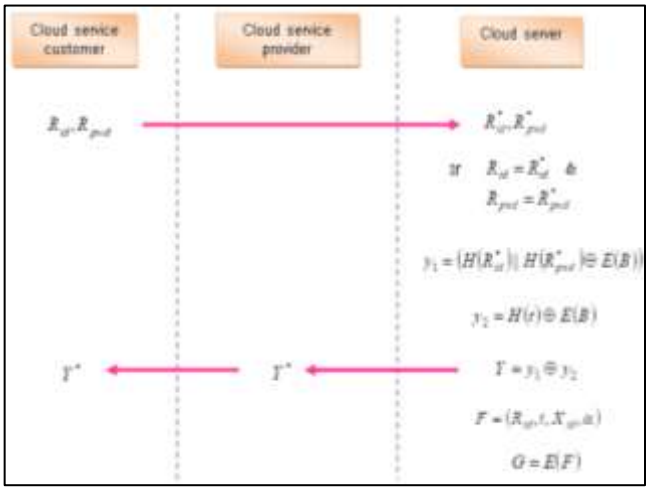

Figure 2. Log file creation phase of proposed multi-level and mutual log integrity preservation method

B. Key Management

After completing the log file creation phase, the key management process begins by generating the ID and password of investigator by CFI. The CFI ID S_{id} and CFI password S_{pwd} generated by the CFI is send to server and CA. The authority checks whether $S_{id} = S^*_{id}$ and $S_{pwd} = S^*_{pwd}$ and if they matched, the CA creates the key factor x_1 and x_2 for generating the shared key of CFI and CA. However, the key factor x_1 is represented as,

$$x_1 = \left(H\left(S^*_{id}\right) \| H\left(S^*_{pwd}\right) \oplus E(B \| m)\right) \quad (5)$$

The ID and the password of CFI are individually applied to the hashing function and the public key is concatenated with the random number such that the concatenated result is applied to the encryption function. The hashed result is Ex-ored with the encrypted result to the generated the key factor x_1.

$$x_2 = H(t) \oplus E(m) \quad (6)$$

The hashed timestamp is Ex-ored with the encrypted random number to generate the key factor x_2.

$$Y_T = x_1 \oplus x_2 \quad (7)$$

The key this is shared between CFI and CSC is generated by the CA using the key factor x_1 and x_2, respectively. Both the key factors x_1 and x_2 are fed to the Ex-or operation to generate the shared key Y_T. Moreover, the key this is shared between the CSC and CFI is represented as,

$$Y_S = H(t) \| E\left(S^*_{id}\right) \quad (8)$$

The hashing function is applied to the timestamp t and the hashed result is concatenated with the encrypted investigator ID stored in the server. The shared key generated by the server is send to the CFI, which receives the key from server and stored it in the investigator database as Y^*_S, respectively. Figure 3 represents the key management phase of proposed multi-level and mutual log integrity preservation method.

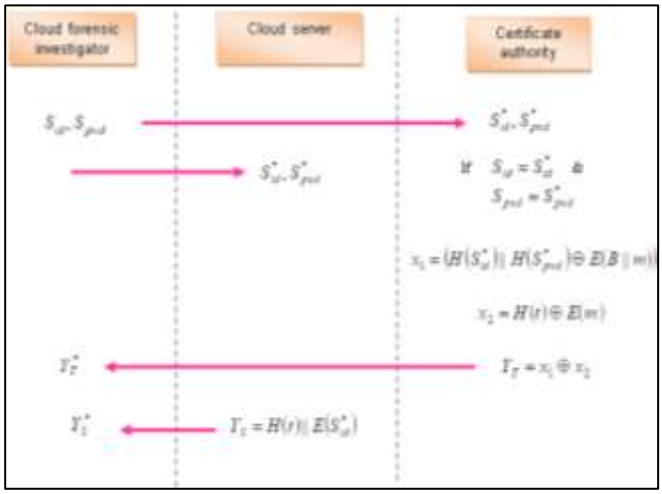

Figure 3. Key management of proposed multi-level and mutual log integrity preservation method

C. Certification and Verification

In this phase, CFI creates the ID S_{id} and the password S_{pwd} along with the ID and password, request message Q and encrypted shared key of CFI and CA are send to the authority. The CA receives the parameters from the investigator and generates a message f_1 as expressed below,

$$f_1 = (Y_T \oplus B \| m) \quad (9)$$

The public key is concatenated with the random number and the result is further ex-or operated with the shared key of CFI and CA. The message generated by the CA is send to CFI, which receives the message and stored it in the CFI as f_1^*. The message f_1^* Ex-ored with the public key and the result is concatenated with the random number. The CFI checks whether $Y_T^r = Y_T^*$, and if it matched, then the CFI generates the mutually verified message *M* and send to the authority along with the ID of CFI. The CA receives the message and store it in the CA as M^* and S_{id}. However, the CA generates the certificate *C* using walsh transform and is represented as,

$$C = L \oplus WT(f_1) \| J \quad (10)$$

The walsh transform is applied to the message f_1 and the resulted factor is concatenated with the file associated with the header. The polynomial factor is used in Ex-or operation with the concatenated result. Accordingly, the polynomial factor is represented as

$$L = 3Y_T^2 + 2Y_T + 1 \quad (11)$$

The certificate is encrypted and the encrypted result is stored as *A* and is send to the CFI, which receives the encrypted certificate and recorded at the investigator as A^*. Figure 4 portrays the certification and verification phase of proposed method.

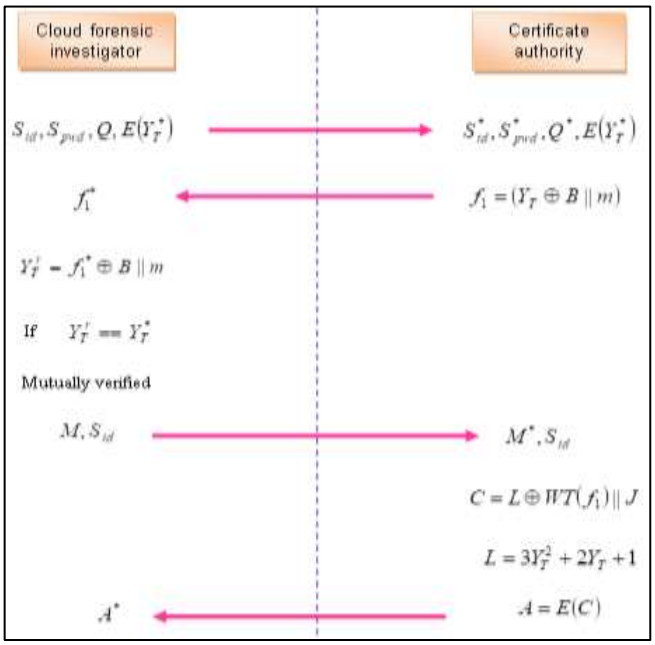

Figure 4. Certification and verification phase of proposed multi-level and mutual log integrity preservation method

D. Proof of Verification

In this phase, CFI ID S_{id}, CFI password S_{pwd}, and the log file request F_{req} are send to the server, which receives these factors and stored in the server database as S_{id}^*, S_{pwd}^*, and F_{req}^*, respectively. After verifying the credentials stored in the server database with the of CFI, the server generates a message a_1 using the below expression as,

$$a_1 = E(Y_S) \oplus m \quad (12)$$

The encryption function is applied to the key that is shared between CS and CFI and the encrypted result Ex-ored with the random number *m*. The server send the message a_1 to CFI, which receives the message and use the message a_1^* to perform Ex-or operation with the random number and applied to the decryption function to generate Y_S^*. The CFI sends the ID of log file along with the encrypted certificate to the server, which receives the factors and stored in the server database as F_{id}^*, and $E(C)^*$. However, the server verifies the certificate and delivers the encrypted log file with respect to the user ID as G_{id}^* to CFI. The success and the failure process is investigated by employing the investigation process to the encrypted user log file ID G_{id} and if it determines as success, then the CFI continue the communication process. Figure 5 portrays the proof of verification phase of proposed multi-level and mutual log integrity preservation method.

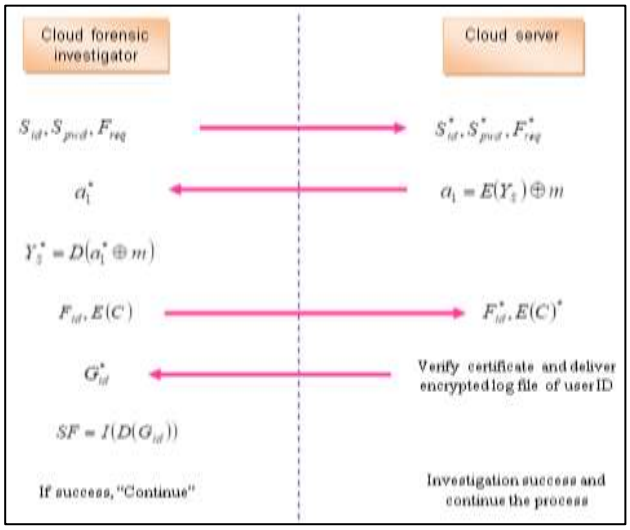

Figure 5. Proof of verification phase of proposed multi-level and mutual log integrity preservation method

V. RESULTS AND DISCUSSION

This section describes the results and comparative analysis of proposed multi-level and mutual log integrity preservation method with existing integrity preservation methods. The implementation of the proposed method is carried out in Cloudsim tool and the performance is evaluated by varying key length with respect to the number of users.

A. Evaluation Metrics

The performance of the proposed multi-level and mutual log integrity preservation model and other existing integrity preservation methods is analyzed by considering the metrics: detection rate, memory, and time.

Detection rate: It is used to measure the quality of detection and it specifies the rate of correctly detected result.

Memory: It is the amount of memory (in bytes) that is consumed to process the integrity preservation mechanism in cloud.

Time: It is also referred as running time that determines the length of time taken to finish the executing process in milliseconds (ms).

B. Comparative Methods

The performance of proposed method is compared with the existing techniques, such as blockchain and Interplanetary File System (IPFS) (Blockchain+IPFS) [26], Bloomfilter based R-tree (BR-tree) accumulator scheme [12], and Log-Based Approach [30], respectively.

1) Analysis with 64-bit key

When implemented with 64-bit key, detection rates of different integration preservation methods are as follows. When considering 200 number of users, the detection rate achieved by the existing blockchain+IPFS, BR-tree, and log-based approach is 0.81, 0.86, and 0.89, whereas the proposed multi-level and mutual log integrity preservation method achieved higher detection rate of 0.93 that shows the percentage of improvement when comparing the proposed with the existing blockchain+IPFS, BR-tree, and log-based approach is 13.2%, 7.9%, and 4.6%, respectively. When increasing the number of users to 500, detection rate obtained by the proposed multi-level and mutual log integrity preservation model is 0.9 such that it reports the performance improvement with that of blockchain+IPFS, BR-tree, and log-based approach is 29%, 8%, and 4%, respectively.

Memory consumption of different integration preservation methods are as follows. For 200 number of users, the memory utilized by the existing blockchain+IPFS, BR-tree, and log-based approach is 5534bytes, 2712bytes, and 1974bytes, while the proposed multi-level and mutual log integrity preservation model utilized less memory of 13241bytes. When increasing the number of users to 500, the memory used by blockchain+IPFS, BR-tree, log-based approach, and proposed multi-level and mutual log integrity preservation model is 6717bytes, 2697bytes, 2684bytes, and 2644bytes.

The analysis of running times of different integration preservation methods when implemented with 64-bit key are as follows. By considering 300 number of users, the time taken by the existing blockchain+IPFS, BR-tree, log-based approach, and proposed multi-level and mutual log integrity preservation model is 6079ms, 5954ms, 5766ms, and 5127ms in such a way that the proposed model shows the percentage of improvement while comparing the developed scheme with the existing blockchain+IPFS, BR-tree, and log-based approach is 12%, 11%, and 9%, respectively. When increasing the number of users to 500, the time taken by the proposed method is 7126ms, whereas the performance gained by the proposed approach with that of blockchain+IPFS, BR-tree, and log-based approach is 29%, 25%, and 20%.

2) Analysis with 128-bit key

When implemented with 128-bit key, detection rates of different integration preservation methods are as follows. When the number of users are considered as 200, detection rate measured by the existing blockchain+IPFS, BR-tree, and log-based approach is 0.844, 0.85, and 0.89, whereas proposed multi-level and mutual log integrity preservation method measured higher detection rate of 0.93 that results the percentage of improvement with that of traditional blockchain+IPFS, BR-tree, and log-based approach is 9.5%, 8.9%, and 4.6%. The proposed method achieved the detection rate of 0.918 for 500 users such that it outcomes the performance enhancement with blockchain+IPFS is 21%, BR-tree is 8.9%, and log-based approach is 6%.

Memory consumption of different integration preservation methods are as follows. For 200 users, the memory used by the traditional blockchain+IPFS is 1806bytes, BR-tree is 1261bytes, log-based approach is 8886bytes, and proposed multi-level and mutual log integrity preservation model is 2974bytes. When increasing the users to 500, memory used by blockchain+IPFS is 2196bytes, BR-tree is 13189bytes, log-based approach is 9684bytes, and proposed multi-level and mutual log integrity preservation model is 2666bytes.

The analysis of running times of different integration preservation methods when implemented with 128-bit key are as follows. For 100 users, the time taken by blockchain+IPFS, BR-tree, log-based approach, and proposed multi-level and mutual log integrity preservation model is 5887ms, 5724ms, 5604ms, and 5063ms such that the performance of improvement reported by the proposed scheme with that of blockchain+IPFS is 14%, BR-tree is 11%, and log-based approach is 9%. When increasing the users to 400, the time taken by blockchain+IPFS is 8482ms, BR-tree is 8359ms, log-based approach is 8173ms, and proposed multi-level and mutual log integrity preservation model is 7018ms that outcomes the percentage of improvement when comparing with existing blockchain+IPFS is 17%, BR-tree is 16%, and log-based approach is 14%, respectively.

3) Analysis with 256-bit key

When implemented with 256-bit key, detection rates of different integration preservation methods are as follows. For 200 users, the detection rate of blockchain+IPFS is 0.84, BR-tree is 0.845, log-based approach is 0.91, and proposed multi-level and mutual log integrity preservation is 0.945, while the performance gained by the developed method when comparing with the traditional blockchain+IPFS is 11%, BR-tree is 10%, and log-based approach is 3%. When number of users are considered as 400, the detection rate measured by the existing blockchain+IPFS, BR-tree, and log-based approach is 0.81, 0.82, and 0.9, while the proposed approach achieved higher detection rate of 0.93 that significantly shows the percentage of improvement with that of blockchain+IPFS, BR-tree, and log-based approach is 12%, 11%, and 3%, respectively.

Memory consumption of different integration preservation methods are as follows. For 200 users, the memory utilized by blockchain+IPFS is 2187bytes, BR-tree is 1256bytes,

log-based approach is 3355bytes, and proposed multi-level and mutual log integrity preservation model is 2313bytes. When considering the users as 500, memory used by blockchain+IPFS is 2172bytes, BR-tree is 1265bytes, log-based approach is 7684bytes, and proposed multi-level and mutual log integrity preservation model is 2185bytes.

The analysis of running times of different integration preservation methods when implemented with 256-bit key are as follows. For 100 users, the time taken by blockchain+IPFS is 7871ms, BR-tree is 7816ms, log-based approach is 5587ms, and proposed multi-level and mutual log integrity preservation model is 5041ms in such a way that the percentage of improvement reported by comparing with blockchain+IPFS is 35%, BR-tree is 35%, and log-based approach is 9%. When increasing the users to 400, the time taken by blockchain+IPFS is 9029ms, BR-tree is 8909ms, log-based approach is 6636ms, and proposed multi-level and mutual log integrity preservation model is 5915ms that reports the performance enhancement as 34%, 33% and 10% with blockchain+IPFS, BR-tree, and log-based approach.

4) Analysis with 512-bit key

When implemented with 512-bit key, detection rates of different integration preservation methods are as follows. When the number of users are considered as 200, detection rate measured by the traditional blockchain+IPFS, BR-tree, log-based approach, and proposed multi-level and mutual log integrity preservation method is 0.8445, 0.85, 0.89, and 0.956 that shows the percentage of improvement while comparing the proposed with that of blockchain+IPFS, BR-tree, and log-based approach is 11%, 11%, and 7%. When increasing the users to 500, detection rate measured by the blockchain+IPFS, BR-tree, log-based approach, and proposed method is 0.71, 0.816, 0.836, and 0.9188 that reports the performance improvement with blockchain+IPFS is 22%, BR-tree is 11%, and log-based approach is 9%.

Memory consumption of different integration preservation methods are as follows. When the users are considered as 300, the memory used by blockchain+IPFS is 2654bytes, BR-tree is 1963bytes, log-based approach is 4684bytes, and proposed multi-level and mutual log integrity preservation model is 2321bytes. For 500 users, memory used by blockchain+IPFS is 1333bytes, BR-tree is 1246bytes, log-based approach is 468bytes, and proposed multi-level and mutual log integrity preservation model is 2834bytes.

The analysis of running times of different integration preservation methods when implemented with 512-bit key are as follows. For 100 users, the time taken by blockchain+IPFS is 2516ms, BR-tree is 2512ms, log-based approach is 5538ms, and proposed multi-level and mutual log integrity preservation model is 5039ms, respectively. When increasing the users to 500, the time taken by blockchain+IPFS is 2538ms, BR-tree is 2522ms, log-based approach is 5711ms, and proposed multi-level and mutual log integrity preservation model is 5020ms, respectively.

C. Defense against Different Security Threats

The proposed method is highly effective against different attacks and the analysis done in terms of proposed approach with that of the existing methods is described in Table 2. The proposed integrity preservation method offers strong authentication and more security when compared with the existing methods, like [26], [12], and [30]. Moreover, the proposed method ensures the mutual and multi-level authentication and resists against password guessing, server proofing, reply attack, main-in-the-middle attack and so on. The existing Blockchain+IPFS failed to resilient against stolen verifier attack, denial of service attacks, and key resilience. The conventional BR-tree does not provide strong user anonymity, and failed to resilient against reconnaissance attack, and key resilience. The conventional methods do not protect against reconnaissance attack, theft attack, whereas the proposed integrity verification method effectively provides better solution to resist against these stacks.

Table 2. Comparison of security against different security threats

Security Threat	Rane, S. *et al.* [26]	Lokhande, P., and Mane, V. [12]	Patidar, M., and Bansal, P. [30]	Proposed multi-level and mutual log integrity preservation method
Provides mutual authentication	Yes	Yes	Yes	Yes
Provides multi-level authentication	Yes	Yes	Yes	Yes
Requires identity-verification table	Yes	Yes	Yes	Yes
Server spoofing attack resistance	Yes	Yes	Yes	Yes
Stolen verifier attack resistance	No	Yes	Yes	Yes
Privileged insider attack resistance	Yes	Yes	Yes	Yes
Password guessing attack resistance	Yes	Yes	Yes	Yes
Provides strong user anonymity	Yes	No	Yes	Yes
Known session-specific temporary information attack resistance	No	Yes	Yes	Yes
Impersonation attack resistance	Yes	Yes	No	Yes
Reply attack resistance	Yes	Yes	Yes	Yes
Man-in-the-middle attack resistance	Yes	Yes	Yes	Yes
Provision for revocation and re-registration	Yes	Yes	Yes	Yes

Free from denial-of-service attack	No	Yes	No	Yes
Profile table-stolen resistance	No	No	Yes	Yes
Key resilience	No	No	Yes	Yes
Reconnaissance attack resistance	No	No	No	Yes
Free from theft attack	No	No	No	Yes

VI. CONCLUSIONS

A multi-level and mutual log integrity preservation method is proposed in this research for assisting forensic analysis of logs in cloud. The proposed method comprises different phases and includes various entities to perform the forensic investigation process. The key is shared between the entities in such a way that the generation of key is done using the hashing and encryption operations. However, the security operations, such as walsh transform, encryption are employed in addition with the concatenation and Ex-or operation to generate the keys than enable to store the log file in server. The creation of log file is made by the server entity using the ID of user, timestamp, ID of CSP, and the service given to the user. The CA creates the certificate using the polynomial factor, walsh transform, and the file associated with the header, respectively. Accordingly, the mutual verification process is accomplished between the forensic investigator and CA based on the key shared between these entities. However, the server verifies the certificate and delivers the encrypted log file with respect to the user ID to CFI. The proposed approach obatined higher performance using the metrics of detection rate, memory, and time with the values of 0.9828, 2185bytes, and 7287ms by considering 500 number of cloud users. The future dimension of research would enhance the performance by considering more number of cloud entities with additional security parameters.

REFERENCES

[1] Reilly D., Wren C. and Berry T., “Cloud computing: Forensic challenges for law enforcement”, IEEE International Conference for Internet Technology and Secured Transactions, **pp. 1-7, November 2010.**

[2] RightScale R., “State of the cloud report”, **2015.**

[3] Tian J. and Jing X., “Cloud data integrity verification scheme for associated tags”, Computers & Security, **pp.101847, 2020.**

[4] Ashok Kumar C., Vimala R., "Load Balancing in Cloud Environment Exploiting Hybridization of Chicken Swarm and Enhanced Raven Roosting Optimization Algorithm", Multimedia Research, **vol.3, no.1, pp. 45-55, 2020.**

[5] Huang P., Fan K., Yang H., Zhang K., Li H. and Yang Y., “A Collaborative Auditing Blockchain for Trustworthy Data Integrity in Cloud Storage System”, IEEE Access, **vol. 8, pp. 94780-94794, 2020.**

[6] Pichan A., Lazarescu M. and Soh S.T., “Towards a practical cloud forensics logging framework”, Journal of Information Security and Applications, **vol. 42, pp. 18-28, 2018.**

[7] Balduzzi M., Zaddach J., Balzarotti D., Kirda E. and Loureiro S., “A security analysis of amazon's elastic compute cloud service”, In Proceedings of the 27th Annual ACM Symposium on Applied Computing, **pp. 1427-1434, March 2012.**

[8] Subashini S. and Kavitha V., “A survey on security issues in service delivery models of cloud computing”, Journal of Network and Computer Applications, **vol. 34, no. 1, pp. 1-11, 2011.**

[9] Zawoad S. and Hasan R., “Towards building proofs of past data possession in cloud forensics”, ASE Science Journal, **vol. 1, no. 4, pp. 195-207, 2012.**

[10] Birk D. and Wegener C., “Technical issues of forensic investigations in cloud computing environments”, In Sixth IEEE International Workshop on Systematic Approaches to Digital Forensic Engineering, **pp. 1-10, May 2011.**

[11] Grispos G., Storer T. and Glisson W.B., “Calm before the storm: The challenges of cloud computing in digital forensics”, International Journal of Digital Crime and Forensics (IJDCF), **vol. 4, no. 2, pp. 28-48, 2012.**

[12] Lokhande P. and Mane V., “Log based privacy preservation in cloud forensic”, 3rd International Conference on Electrical, Electronics, Engineering Trends, Communication, Optimization and Sciences (EEECOS 2016), Tadepalligudem, **2016, pp. 1-6,** doi: 10.1049/cp.2016.1513.

[13] Zawoad S., Dutta A.K. and Hasan R., “Towards building forensics enabled cloud through secure logging-as-a-service”, IEEE Transactions on Dependable and Secure Computing, **vol. 13, no. 2, pp. 148-162, 2015.**

[14] Goodin D., “Amazon cloud hosts nasty banking Trojan”, The Register, **2011.**

[15] Zamani M., Safkhani M., Daneshpour N. and Abbasian A., “A New Searchable Encryption Scheme with Integrity Preservation Property”, Wireless Personal Communications, **pp. 1-24, 2020.**

[16] Jain P., “Decentralize Log File Storage and Integrity Preservation using Blockchain”, International Journal of Computer Science and Information Technologies, **vol. 11, no. 2, pp. 21-30, 2020,**

[17] Jayaraman I. and Panneerselvam A.S., “A novel privacy preserving digital forensic readiness provable data possession technique for health care data in cloud”, Journal of Ambient Intelligence and Humanized Computing, **pp.1-14, 2020.**

[18] Dykstra, Josiah. "Seizing Electronic Evidence from Cloud Computing Environments." In Cloud Technology: Concepts, Methodologies, Tools, and Applications, IGI Global, Hershey, PA, pp. 2033-2062, 2015.

[19] Bellare, M. and Yee, B., “Forward-security in private-key cryptography”, In Cryptographers’ Track at the RSA Conference, Springer, Berlin, Heidelberg, pp. 1-18, April 2003.

[20] Garg, N., Bawa, S. and Kumar, N., “An efficient data integrity auditing protocol for cloud computing”, Future Generation Computer Systems, **2020.**

[21] Devagnanam J, Elango N M, "Optimal Resource Allocation of Cluster using Hybrid Grey Wolf and Cuckoo Search Algorithm in Cloud Computing", Journal of Networking and Communication Systems, **vol.3, no.1, pp. 31-40, 2020.**

[22] Birk, D. and Wegener, C., “Technical issues of forensic investigations in cloud computing environments”, In Sixth IEEE International Workshop on Systematic Approaches to Digital Forensic Engineering, **pp. 1-10, May 2011.**

[23] Holt, J.E., “Logcrypt: forward security and public verification for secure audit logs”, In ACM international conference proceeding series, **vol. 167, pp. 203-211, January 2006.**

[24] Ma, D. and Tsudik, G., “A new approach to secure logging”, ACM Transactions on Storage (TOS), **vol. 5, no. 1, pp. 1-21, 2009.**

[25] Vhatkar Kapil Netaji, Bhole G P, "Optimal Container Resource Allocation Using Hybrid SA-MFO Algorithm in Cloud Architecture", Multimedia Research, **vol.3, no.1, pp. 11-20, 2020.**

[26] Rane, S., Wagh, S. and Dixit, A., “Securing Trustworthy Evidences for Robust Forensic Cloud in Spite of Multi-stakeholder Collusion Problem”, In International Conference on

Hybrid Intelligent Systems, Springer, Cham, **pp. 376-386, December 2019.**

[27] Singh, K.D., Sharma, A., Singh, S., Singh, V. and Rane, S., "Integrity and confidentiality preservation in cloud", In IEEE International conference of Electronics, Communication and Aerospace Technology (ICECA), **vol. 2, pp. 419-424, April 2017.**

[28] Dalezios, N., Shiaeles, S., Kolokotronis, N. and Ghita, B., "Digital forensics cloud log unification: Implementing CADF in Apache CloudStack", Journal of Information Security and Applications, **vol. 54, pp. 102555, 2020.**

[29] Khan, M.N.A. and Ullah, S., "A log aggregation forensic analysis framework for cloud computing environments", Computer Fraud & Security, **no. 7, pp. 11-16, 2017.**

[30] Patidar, M. and Bansal, P., "Log-Based Approach for Security Implementation in Cloud CRM's", In Data, Engineering and Applications, Springer, Singapore, **pp. 33-43, 2019.**

AUTHORS PROFILE

Mr. Siva Rama Krishna Tummalapalli is currently working as Assistant Professor in the Department of Computer Science and Engineering at JNTUK University College of Engineering Vizianagaram, India since 2013. He is a life member of CSI, CRSI, ISCA and ISTE, and a member of ISOC, NCUC and IACCP. He is an AWS Certified Cloud Practitioner, Corda Certified Developer, and Automation Anywhere Certified Advanced RPA Professional. He has 1 patent and published 15 research papers in reputed international journals and conferences. His research interests are Network Security, Privacy Issues, Cloud Security, IoT and Digital Forensics. He has 11 years of teaching experience.

Dr. A. S. N. Chakravarthy is currently working as Professor and Head of Department of Computer Science and Engineering at JNTUK University College of Engineering Vizianagaram, India since 2013. He is a life member of IETE, IEI, CSI, CRSI, ISCA, ISTE and ASDF, senior member of IEEE, and a fellow of ISCA. He authored 3 books, has 2 patents, and published 106 research papers in reputed international journals and conferences. His research interests are Computer Networks, Steganography, Watermarking, Data Security, Password Authentication, Biometrics, Cyber Security, Cloud Privacy, IoT Security and Digital Forensics. He has 18 years of teaching experience and 12 years of research experience.

