Received: 8 January 2021 | Accepted: 23 February 2021

DOI: 10.1002/ett.4272

SPECIAL ISSUE ARTICLE

WILEY

BCALS: Blockchain-based secure log management system for cloud computing

Ahmad Ali[1] | **Abid Khan**[2] | **Mansoor Ahmed**[1,3] | **Gwanggil Jeon**[4]

[1] Department of Computer Science, COMSATS University, Islamabad (CUI), Islamabad, Pakistan

[2] Department of Computer Science, Aberystwyth University, Aberystwyth, UK

[3] Innovative Value Institute, Maynooth University, Maynooth, Ireland

[4] Department of Embedded Systems Engineering, Incheon National University, Incheon, South Korea

Correspondence to:
Gwanggil Jeon, Department of Embedded Systems Engineering, Incheon National University, Incheon 22012, South Korea.
Email:gjeon@inu.ac.kr

Abstract

A computing environment requires a robust and comprehensive process to track and document user activities to uphold confidence in the system. Audit logs are used for this purpose to monitor the actions of administrators and users. However, these logs are vulnerable to multidimensional attacks, including modification of logs, erasability of logs, and privacy of the user. Since administrators have unprecedented access to these logs, they can modify, delete, and even destroy them. Securing these logs against malicious activities is the prime requirement of audit log management. Existing schemes have several limitations, including immutability, computational expensiveness, missing semantics, and are not verifiable. Various schemes have been proposed for this purpose, but a standard method is required to structure heterogeneous logs and their security semantically. To cope with these limitations, in this paper, we propose a Log Management System using blockchain. The proposed system will ensure audit logs' security, which will eventually strengthen users' trust in the computing environment and make it unbreachable even by the administrators. It has been evinced that our model performed better in terms of performance and features already mentioned when compared with existing schemes.

1 | INTRODUCTION

The reliance of the manufacturing industry on cloud-based services has enabled the emergence of the next-generation manufacturing paradigm that has the potential to revolutionize the manufacturing industry. Cloud services will be the most popular choice for manufacturing companies, and most of them are already developing cloud-based manufacturing services. The Internet of Things (IoT) is playing a vital role in every dimension of our day-to-day activities. In fog computing, smart devices, that is, IoTs, are known as Edge Nodes that are connected to a cloud using a gateway that acts as a Micro Data Center. Fog computing has increased cloud services availability because of its localized behavior despite its advantages and disadvantages. However, this holistic vision raises some concerns, like which level of security these systems could provide and how it offers and protects the privacy of its users[1,2]? Tenants of cloud services are concerned about their data and services available in the cloud. A comprehensive security mechanism is required in cloud computing to satisfy the clients and respective stakeholders, as discussed by Reference 3. Functionality and performance of information systems, including their unit devices, can be monitored and explored by analyzing the logs generated by the interconnected devices and central monitoring in cloud computing.[2] Accountable audit logs can provide a better level of trust for service providers as well as service tenants.[4] Trust is an important factor for the utility of computing services; therefore, trust and trust management are very significant in the usage, infrastructure, reliability of services, and the whole information system.

Computer forensics requires tamper-proof logs for a successful investigation and forensics. In trusted computing systems evaluation criteria,[5] security requirements of audit logs are described as "Audit data must be protected from modification and unauthorized destruction to permit detection and after-the-fact investigation of security violations." It is important to note that forensic results of any cloud services depend on input logs' health as highlighted by Reference 6.

A blockchain network is primarily comprised of a set of peer nodes which are fundamental elements of the blockchain network because they host ledgers and smart contracts. Blockchain is an implementation of distributed ledger technology which is a record-keeping technology that supports the storage of immutable transactions among the peers based on blockchains and has been used in cryptocurrencies as well as some other immutable data storage applications. Distributed Ledger Technologies (DLT) have a significant features and supersedes the previous record-keeping technologies of this category. Fundamental characteristics of a blockchain are immutability of records, dis-intermediation, and no central control by a single party, new opportunities for management data and statistics as well as data sharing. Transactions in a digital ledger/blockchain are governed by smart contracts. These are automated digital contracts by which the rules and regulations of the transaction handling and its processing are transformed and embedded in computer software code, to be automatically fulfilled upon acknowledgment of a particular input.

Access control in a DLT can be categorized as Permissioned, Permission-less, and Federated Ledgers. Permission-less ledgers are public ledgers that can be accessible by anyone using the Internet. Bitcoin, Ethereum etc. are common examples of permission-less ledgers. These are based on Proof of Work consensus algorithms which are not permissioned. Anyone can participate without any special permission. Only a limited number of transactions can be performed in a specific time span. Federated ledgers operate under the leadership of a group contrary to public ledgers also known as consortium blockchains. Any person with access to the Internet is not allowed to participate in the process of verifying transactions. These are faster (higher scalability) and provide more transaction privacy. This type of ledger is being used by the banking sector and insurance companies. The consensus process is governed by a preselected set of nodes. Private ledgers are centrally controlled and governed by an organization. MONAX and Multichain are common examples of this approach.

Health-record keeping, parking-lots management, real-estate business record, and identity management are some examples of blockchain implementations.

Local and remote administrators and service providers have full access to respective resources and may pose severe information security threats. Securing information system resources are the key requirements for a trustable environment. Audit logs are used to monitor the performance of the resources and the activities of users (Administrators and users) and troubleshoot the issues in an information system environment. These logs are heterogeneous in proprietary issues and data-structures and are not machine-understandable because they lack semantics. To take the true advantages of these logs, an immutable storage scheme with heterogeneity support, semantically enriched, and inbuilt statistics sharing for stakeholders is really needed to counter the following:-

1. Volatile Log Storage.
2. Malicious power users' threats.
3. Heterogeneity in Logs.
4. Missing Semantics in log entries.

To protect the integrity of logs, a lot of work has been done, and the quest for the best is underway. Secure Logging as a Service (SecLaaS), Blind Aggregate Forward (BAF) and Fast Immutable-Blind Aggregate Forward (Fi-BAF), and Practical and Immutable Signature Bouquets (PISB) proposed in Immutable Authentication, and Integrity Schemes for Outsourced Databases are some examples of such schemes. These are computationally expensive, resource-hungry, and are not verifiable among peers or stakeholders. Immutability, semantics, and distribution of statistics are critical issues in these logging schemes.

Our objective is to develop a secure and semantics enriched audit logs security system where administrators cannot modify the systems' traces available in audit logs. We propose an efficient decentralized, secure audit logs scheme using private blockchain implementation of bitcoin protocol (Multichain) known as BCALS. Furthermore, to get most from the audit logs, we used Elasticsearch, a JSON text/document-based full-text search engine based on "Lucene Library." It provides a distributed, multitenant capability with an HTTP web interface and schema-free JSON documents. A transformation layer has also been used to overcome audit logs' heterogeneity issues and provide semantics enrichment. This conversion is based on a dynamic transformation template, separate for each type of log.

The rest of the paper is organized as follows: Section 2 covers related work and critique on existing schemes. System model, comprehensive security requirements, and attacks and goals in addition to an attacker model concerning the logging scheme are given in Section 3, and the proposed solution is given in Section 4. Evaluation parameters, results, and performance evaluation are given in Section 5. Benefits of BCALS are highlighted in Section 6 and finally, this paper is concluded in Section 7.

2 | RELATED WORK

Log management is quite a mature subject in a computing environment. Various solutions are available for log management. Open-source Security Information and Event Management (SIEM) Solutions like GFI Events Manager, Syslog-ng, Manage-Engine Log Storage and Analyzer, LOGalyze, and Splunk Enterprise, etc., are a few examples of such solutions. All these log management applications are capable enough to store and analyze logs of various devices. System administrators have full control over these log stores, and logs' integrity can not be guaranteed. To maintain the integrity of logs even after a system compromise, researchers have proposed different solutions. Symmetric, as well as asymmetric encryption schemes, have been used to maintain data security, LogCrypt[7] is an asymmetric cryptography scheme for log security in which logs are secured using encryption and are publicly verifiable. There is an additional overhead of encryption, decryption processes, and maintaining keys for these encryption types. Signature aggregation Schemes using encryption like BAF, BAFi, and Fi-BAF(Fast Immutable BAF), etc., have been proposed in References 8-10. Similarly, References 11,12 proposed aggregation based schemes. FssAgg scheme[11] has been proposed for aggregation based secure authentication to protect previously logged in long sessions only. Using Authentication Data Structures as Balloons[13] is also another scheme to store log entries in a famous balloons addition approach which introduced only a data structure for secure authentication. In a cloud computing scenario, Logging as a service is also proposed for cloud users. This scheme also depends on the trustworthiness and integrity of the cloud service providers, where system users have the possibility to manipulate these logs. Different researchers have proposed a scheme for receiving, storing, and analyzing these logs using a central log server, SecLaaS, which was proposed in Reference 14 uses cloud functions to store logs, but miscreant cloud administrators may compromise these. In Reference 15, the authors proposed εSecure Logging as a Service using Reversible Watermarking,ε where logs are stored in the cloud for a longer duration, and content authentication is carried out by reversible watermarking. In this approach, only forensics of contents authentication is possible, whereas other security parameters are missing. Semantics-based logging has been proposed in References 16,17. Data extraction using data mining approaches and methods to deal with heterogeneity has been explained in these semantics-based solutions. In contrast, security, immutability, and other basic security requirements are out of sight. Considering a distributed setting, Accorsi[18] proposed "BBoxε approach to apply trusted computing to ensure authenticity and confidentiality of log entries. In contrast, we do not require an additional entity that can become a single point of failure. SLOPPI,[19] another encryption-based scheme for data integrity and policy compliance, whereas immutability, statistics distribution, and semantics of logs are not available. Anonymizing the log data as proposed in Reference 20 surely helps in data privacy, but the basic requirements of data security are missing. Henze et al proposed a framework for secure communication in IoT.[21] The proposed framework allows for control and management from a central location to protect an IoT network from a cloud services provider with malicious intentions. It logs control messages at redundant locations to be verified through different gateways. Old and duplicate messages are removed continuously to reduce the size. The log messages verification is used to highlight malicious behavior, which helps defend cloud-based IoT from tempering, insertion, withholding, and reordering messages. Practical and Robust Secure Logging[12] is an extension of Fi-BAF and proposed storage of sequential aggregate signatures of log entries.

Some hardware-based secure logging schemes have been proposed by some researchers. In Reference 22, authors introduced storage of logs in a forward integrity concept using write once and read multiple (WORM) scenarios. Tamper Proof Storage of Logs using Trusted Platform Module 2.0 (TPM) is explained in Reference 23. EmLog as a Tamper-Resistant System Logging introduced for constrained devices with Trusted Execution Environments (TEEs).[24] All the above-discussed hardware-based schemes are constrained in terms of resources and processing. Their capacity, cost, and availability are not suitable for larger networks with many computing units. Similarly, other nonhardware-based schemes cannot comprehensively provide immutability, semantics, and distribution of logs statistics. File system-based schemes were discussed in Reference 25 contradictory proof for Secure audit log storing techniques was also proposed. Distributed Immutability of logs is explained in Reference 26 and Privacy logs storage using Blockchain is proposed in Reference 27 where only private data is protected from public auditors.

In this section we have explored existing schemes along with their features and observations. Based on this study, a comparison has been drawn in Tables 1 and 2.

TABLE 1 Comparison—Overview of existing techniques

Sr	Technique	Strength	Observation
1	LogCrypt,[7] 2006	Publicly verifiable logs using PKI	Processing overhead, Key change issue
2	FssAgg,[12] 2007	Sequential Aggregate based for long sessions	Only for live sessions
3	BAF,[8] 2009	Aggregation Based	Processing Overhead
4	BBox-,[18] 2010	PKI and trusted execution based log data security during transmission and at rest	Processing Overhead
5	LogFAS,[28] 2011	Fewer Computations Required- Publicly verifiable	Processing Overhead
6	Semantic Logging,[16] 2011	Semantics based logging system	Logs Security parameters are missing
7	BAF & FI-BAF,[9] 2012	Aggregation Based, Less Computations Required	Cryptography Overhead & distribution of logs
8	SLOPPI,[19] 2013	Central Log Server	No logs security and statistics sharing
9	SecLaaS,[14] 2013	Central Log Server	Data can be deleted
10	Semantically Formalized Logging,[17] 2015	Formally structured Semantics based logging	Basic / extended Security parameters are missing
11	BAFi,[10] 2015	Improved BAF	Cryptography Overhead and No redundancy of logs
12	Secure Logging as a Service Using Reversible Watermarking,[15] 2017	Strong Content authentication mechanism	Real-time log reception and storage is not considered
13	Distributed Configuration, Authorization and Management,[21] 2017	DLT based Configuration Logs storage, Logs Verification and distributed control on configuration logs	Covers only configuration logs storage
14	Practical and Robust Secure Logging,[12] 2017	Sequential Aggregate Signatures	Publicly verification only

TABLE 2 Comparative analysis of existing schemes for security properties

Ser	Technique	Confidentiality	Integrity	Availability	Immutability	Heterogeneity	Semantics	Statistics
1	LogCrypt,[7] 2006	✓	✓	X	✓	X	X	X
2	FssAgg,[11] 2007	X	✓	X	✓	X	X	X
3	BAF,[8] 2009	X	✓	X	✓	X	X	X
4	BBox-,[18] 2010	✓	✓	X	✓	X	X	X
5	LogFAS,[28] 2011	X	✓	X	✓	S	X	X
6	Semantic Logging,[16] 2011	X	X	X	X	✓	✓	X
7	BAF & FI-BAF,[9] 2012	X	✓	X	✓	X	X	X
8	SLOPPI,[19] 2013	X	✓	X	✓	X	X	X
9	SecLaaS,[14] 2013	X	✓	✓	X	X	X	X
10	Semantically Formalized Logging,[17] 2015	X	X	X	X	✓	✓	X
11	BAFi,[10] 2015	X	✓	X	✓	X	X	X
12	Secure Logging as a Service Using Reversible Watermarking,[15] 2017	✓	✓	X	✓	X	X	X
13	Distributed Configuration, Authorization and Management,[21] 2017	✓	✓	✓	✓	X	X	✓
14	Practical and Robust Secure Logging,[12] 2017	X	✓	X	✓	X	X	X

2.1 | Comparison—Overview of existing techniques

In this section, a tabular comparison of existing schemes is given with respect to their strengths and observations.

2.2 | Comparative analysis for security properties

In this section, a tabular comparison of existing schemes is given with respect to the security features being provided by them.

Existing schemes along with their features and observations are discussed in this section and a comparison has been drawn in Tables 1 and 2. Models and goals are covered in following sections.

3 | MODELS AND GOALS

In the previous section, we have explored existing schemes along with their features and observations. Based on this study, a comparison has been drawn in Tables 1 and 2. Models and goals are covered in the following sections.

3.1 | Threats to audit logging schemes

Multiple vulnerability points affect the logging scheme and users' trust. Known Secure logging schemes are not secure enough to be trusted. In Reference 28, the authors highlighted vulnerabilities and possible attacks on LogFAS,[28] BM and AR FssAgg schemes claimed as secure against such vulnerabilities. Log generation, Log collection, Network, Log storage are common building blocks of logging schemes, and all these are susceptible to different vulnerabilities. At the log generation level, an attacker can hinder the log generation process. In the log collection phase, the attacker can attack the system and resources where logs are being collected from various locations and make such systems unavailable, for example, DOS and DDOS attacks. In log transfer/communication on the network, the attacker can disrupt the communication on the network channel and launch εMan in The Middle attackε. Log storage is also vulnerable where storage resources are unavailable by any means when required. In the log analysis phase, the attacker exploits resources on which log analysis is being performed to investigate various vulnerabilities found in logs. Cloud administrators and remote administrators, having full access to respective resources, may pose severe insider threats on which tenants show their concerns. Securing these resources from insiders is the key issue. MITIS framework[29] has been proposed to mitigate insiders' threat and enhance users' trust in the environment by implementing two men rule in a supervisory role. It is worth noticing the importance of log entries and users' behavior in this framework.

Local and remote administrators and service providers have full access to respective resources and may pose severe insider threats. Securing information system resources is an essential requirement for a trustable environment. Logs are used to monitor the performance of the resources and the activities of users (Administrators and other users) and troubleshoot the issues in an information system environment. These logs are heterogeneous in proprietary issues and data structure. Due to the nonavailability of semantics, most of these logs are not machine-understandable. To take real advantages of these logs, an immutable storage scheme with heterogeneity support, semantics enabled, and inbuilt statistics sharing approach for stakeholders is a need of the time to counter the following:-

- Privileged users' threat. Log data collection and analysis is the primary process for producing a piece of digital evidence against any malicious activity. In general, users have limited access to log entries, whereas administrators have full access to these entries, which pose severe insider threats.[29]
- Volatile Log Storage Applications. Log management applications allow clearance of logs, which allow to clear log entries and destroy digital evidence.
- Heterogeneity in Log Entries. Similarly, log entries are proprietary and asymmetric because their structure varies from application to application and device to device.[30]
- Missing Semantics in Log Entries. Immutable and semantics-aware log storage is not available.[31]

3.2 Attacker model

Securing audit logs from a miscreant is an important factor in a log management system. Logs forward security is the protection of logs after a system compromise from any alteration. An attacker may opt for any of the following after her successful attack.

1. Tampering logs to invalidate logs integrity,[32] that is, Any of Modification Attack and Reorder attack.
2. Erasing digital fingerprints.[21]
3. May launch delayed attack to disturb log management system,[21] that is, Withhold Attack or delay attack.
4. May generate fake log entries to overload/disturb log management system and cause a denial of service, that is, Insertion Attack.
5. Log data may be leaked out by the attacker, that is, Privacy Attack[21]

3.3 Assumptions

Since achieving secure and scalable data integrity in logs is complex, we have made few BCALS design assumptions. Some of the assumptions are already considered in the current Internet infrastructure. This facilitates us to limit the study scope, and at the same time and ensure that BCALS is practical and fail-safe.

1. The stakeholders are inclined to use a trusted system and cooperate but have limited trust.
2. The trust and responsibility of managing the logs belong to all users.
3. One organization should be able to administer the network.
4. Anti-counterfeiting. Each entity uses the Elliptic Curve Digital Signature Algorithm (ECDSA), the same as used in Bitcoin protocol, ensuring that its signature is not vulnerable in subexponential time. Thus, we assume that the forging of a signature without having access to the right secret key is not possible by any adversary.
5. Multichain (Bitcoin Protocol). Multichain used as it is a private blockchain using Bitcoin protocol and provides strong security and reliability.
6. Resource-constraints. The user is highly conscious about cyberattacks and insider threat and is ready to provide additional resources required for computational requirements and storage.

3.4 Security requirements of an event logging system

In computer forensics, the most important information security factor is εNon-Repudiation,ε which can be managed by recording users' sensitive activity. Along with some handling issues like (real-time transfer, storage, accessibility, and alert generation based on some specific event), the following are critical issues in log storage and analysis schemes. A logging scheme is considered to be a comprehensive resource for recording critical events. These security requirements are tabulated in Table 3. These requirements can be categorized into basic security requirements and extended security

TABLE 3 Key requirements of security

Sr. #	Parameters	References
SP-1	Confidentiality	SP-1 to SP-5 are highlighted by all of the following. NIST Guidelines,[34] BBOX,[18] SecLaaS,[14] Immutable authentication and integrity schemes,[35] Data-centric accountability[36]
SP-2	Integrity	
SP-3	Availability	
SP-4	Authenticity	
SP-5	Immutability	
SP-6	Heterogeneity Support	Heterogeneity and dynamicity of clouds[30]
SP-7	Semantics	Evidence-based context-aware log data[31])
SP-8	Statistics Sharing	Trust Management[4]

requirements. Confidentiality, Integrity, Availability, Nonrepudiation, and Privacy are considered basic security requirements of a secure and trusted logging scheme. Confidentiality means the prevention of unauthorized access. Integrity is required to safeguard data from being altered or even deleted. Availability means the assurance and the guarantee of data being available when required in the form it was saved. Nonrepudiation is the property required to provide the proofs having sufficient data of the occurrence of an activity. Privacy treats personal matters, and that should not be leaked or shared with others.[33] In extended security requirements, logging schemes must provide correctness, integrity with forwarding security, immutability, insider threat mitigation, and statistics sharing. Correctness deals with the authenticity of the information, whereas Integrity with forward Security makes logs data protected in case of any miscreants' activity. If a system gets compromised, its previously stored log data should not be altered by the miscreants or even by the system administrators by any means. Immutability covers all aspects of data consistency; that is, it makes logs data unchangeable after its storage. Insider Threat mitigation deals with insiders with mal-intention because they have unprecedented access to the system resources. A service provider's trustworthiness is closely related to the reality of activities and sharing statistics of such activities, which contributes toward stakeholders' trust. A logging scheme should have such methods to share statistics of activities to establish its trustworthiness.

3.5 | Goals

BCALS is designed to provide trust in a service provisioning environment, it is designed to provide the following goals.

1. **Auditability**. The auditing process should be available to the peers, where each peer can review the auditing process and confirm the inputs.
2. **Immutability**. Each and every administrative activity should be logged using Blockchain to provide immutability.
3. **Collusion protection**. A log management scheme should provide a mechanism where no collusion attack is possible while storing data in a distributed environment.
4. **Efficiency**. The number of peers affects the efficiency in Blockchains. The system should be capable enough to handle workload and efficiency issues amicably.
5. **Decentralization**. To avoid the threat of a single point of failure, Audit log data must be available at redundant locations under strict security control.
6. **Semantics**. To get the most out of these logs, these entries should be tagged and semi-structured so that the attributes of a blog entry can be linked and understood by machines.
7. **Analysis Support**. Logs are required to available in such a structure that they can easily be incorporated in link open data (LoD) applications or graph databases.

4 | BCALS-BLOCKCHAIN-BASED SECURE LOG MANAGEMENT SYSTEM

We carried out an empirical study to find the best solution, and the same was followed to conclude the subject. A central log storage server is configured on the gateway of each fog unit. It is configured to securely collect logs from smart devices such as edge devices, segregate the log collection over a secure communication channel, and transfer them into their respective databases. Since activity, operational and data logs are considered useful at fog level and do not require across the fog level and can be secured by log replication at some other storage. Only administrative activity logs are required across the fog level for forensics, so these are stored in a Blockchain. For this purpose, a private permissive Blockchain of logs against each IoT unit is created and shared among all Fog Units (Fog A to Fog N) Block-chain technology is famous for its immutable data and integrity. It does not allow backdate modifications. This framework additionally proposes a monitoring service on the cloud, which looks after the integrity of Blockchain and will generate an alert in case of any change in the log blocks. Semantic web technologies are being used to resolve data structure symmetry issues. Storing these logs using some ontological structure will surely enable the logging system to manage heterogeneous log data.

This section describes the abstract of the framework where a fog environment is focused with an "N" number of fog units, and each fog unit contains an "n" number of IoT/Edge devices. IoT devices in fog units generate useful data as logs. These logs are segregated into three types as ((a) Administrative and Security Logs; (b) Operational Logs; (c) Activity Logs, or Device Specific Log). It is a management policy to define which type of logs must be stored securely and immutably. Segregation and further processing of these logs are carried out by a log server installed and configured at fog gateway, which is connected with peer fog units. Blockchain is a well-known distributed ledger technology being used in digital currency with few predefined considerations. Here we assume that each Fog Unit performs as a Peer Node of Blockchain, and Administrative or Security Logs are required to be stored securely (Immutably).

4.1 | Proposed framework

Secure and efficient log management must satisfy the properties of Correctness, Forward security, and protection against corruption of audit logs. Immutable as well as semantics-aware storage of logs is required and a real need of the time. BCALS framework is proposed to manage and store immutable logs in the fog environment for accountability and trustworthiness. Finally, a deterrence-based stakeholder's trusted system will be achieved by introducing a Permissive Block-chain to store semantically aware logs. This Block-chain is distributed among the stakeholders; however, some additional processing will be involved in Block-chain implementation. This framework further strengthens forensics by introducing semantics in the logs and introducing the concept of immutable storage of logs to ensure the recording of activities. Components and working of BCALS is shown in Figure 1.

4.2 | Strength of the proposed model against the attacker model

The proposed framework "BCALS" provides the required security parameters along with Immutability and Semantics. The multifold strength of the proposed framework is provided as under.

1. A secured gateway-based log management will protect against evidence erase-ability and delayed attacks.
2. Log entries will be classified and filtered at the gateway level for saving required logs only.
3. These logs are transformed into a semantic-aware data structure to provide heterogeneity support.
4. Audit logs are secured from tampering using distributed ledger technology.
5. Permissive distributed ledger is used to enhance security using permissions-based blockchain.

BCALS is focussed to provide trust in a service provisioning environment, it is designed to provide the following goals.

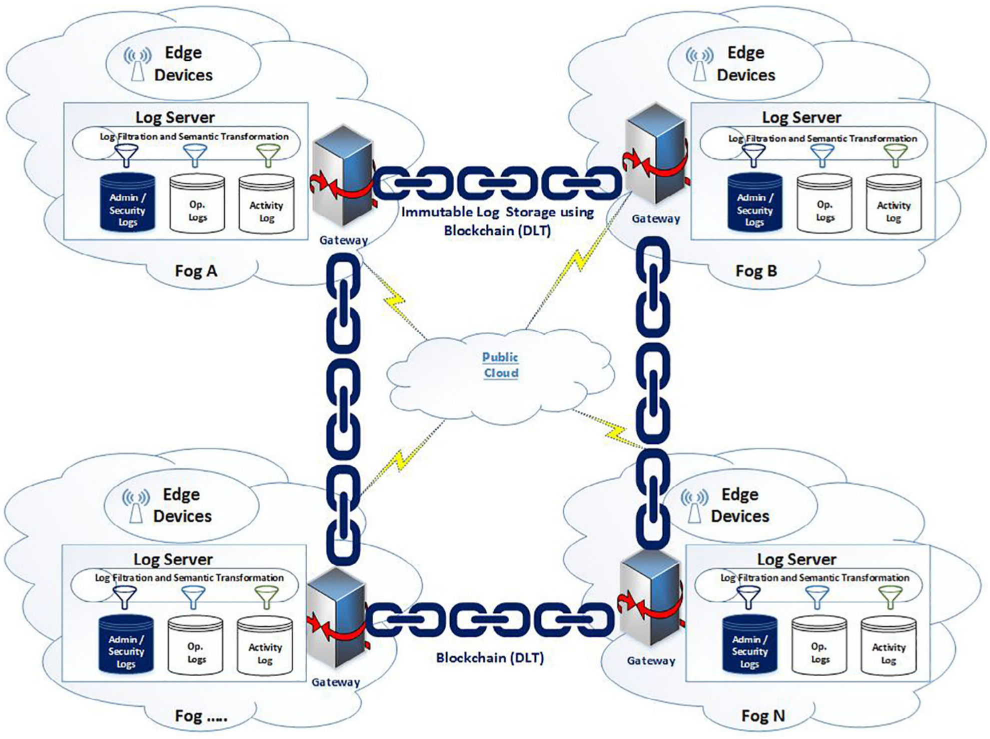


FIGURE 1 BCALS-Proposed framework

1. **Auditability**. The audit data will be available to the peers, where each peer can review the required data.
2. **Immutability**. Each and every administrative is being logged using Blockchain to provide immutability.
3. **Collusion Protection**. BCALS uses private Blockchain, and only a specified peer controls the block generation, so no chance of collusion is possible.
4. **Efficiency**. A number of peers affect the efficiency in Blockchains; therefore, BCALS blockchain is shared among service-provider and tenants only. Different tenants may have different Blockchains for their respective services.
5. **Decentralization**. To avoid the single point of failure, BCALS can be extended with a trusted third party as a peer.
6. **Semantics**. To get the most out of the administrative logs, BCALS uses JSON transformations as tagged semi structured documents.
7. **Analysis Support**. Logs are stored as JSON documents in Elasticsearch and can provide better insight into logs using any graph database or linked open data(LoD).

4.3 | Detailed functional architecture of BCALS

Our proposed framework is performing some additional functions as compared to the generics. Core functions of the proposed framework are explained as under. In our proposed framework, environment monitoring is actively carried out by the gateway system. By this active monitoring, availability and connectivity of the components are ensured. The flow chart of BCALS is presented in Figure 2.

Logs generation. Event-based logs are generated at the gateway by the surveillance agent and the agents installed on the individual components.

Logs collection and storage. A central collection of these logs is configured at the gateway and administered using MITIS[29]/VisTAS framework to counter the insider threat.

Logs filtration. Multiple devices are connected to make a computing environment, and logs of multiple types are collected from these devices. To benefit from a true Blockchain-based distributed technology, it is necessary to make log data size as short as possible. Here we have recommended the classification of logs and separate administrative or security logs from other less important logs.

Logs transformation and semantics inculcation. After the classification of administrative or security logs, these logs are transformed into machine-understandable ontological data or open linked data and make them semantically aware.

Semantically aware logs storage and distribution.

At this stage, logs have been transformed into semantically aware logs and are available for storage in a permissive distributed ledger environment. Fog units behave like blockchain peers. These logs are stored in the peer and distributed across the Blockchain.

Alert generation for stakeholders. Blockchain-based distributed ledger technology has a built-in immutable and permissive architecture; therefore, an alert is generated in case of any modification in the chain. All the stakeholders are informed, which provides real-time monitoring.

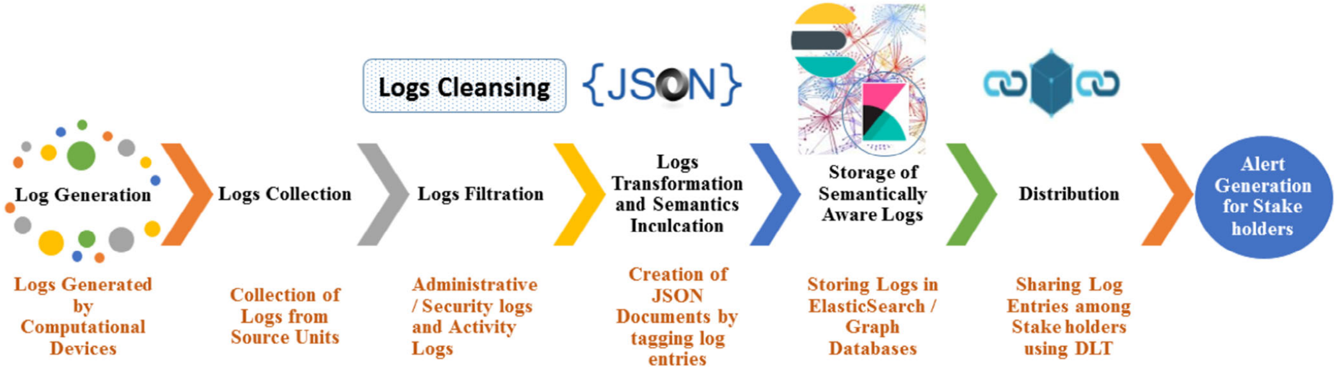

Logs Cleansing
{JSON}
Log Generation
Logs Collection
Logs Filtration
Logs Transformation and Semantics Inculcation
Storage of Semantically Aware Logs
Distribution
Alert Generation for Stake holders
Logs Generated by Computational Devices
Collection of Logs from Source Units
Administrative / Security logs and Activity Logs
Creation of JSON Documents by tagging log entries
Storing Logs in ElasticSearch / Graph Databases
Sharing Log Entries among Stake holders using DLT

FIGURE 2 Flow chart of BCALS

4.4 | Algorithm—BCALS implementation

BCALS framework can be understood and implemented by the following algorithms.

Algorithm 1 explains the overall environment and functions of BCALS. Algorithm 2 explains fog units initialization required to be monitored by the BCALS. Algorithm 3 covers the establishment of a gateway for monitoring the units, and JSON parsing and semantics is shown in Algorithm 4.

Algorithm 1 describes the overall implementation of the proposed scheme BCALS. In the first step, when an environment is activated and interconnected, a fog computing environment is initialized, and connections with all the edge devices are established. Sequel to this, the gateway monitoring system is also activated. All the event logs are captured with the highest privileges, and this monitoring can not be suspended in any case because peers or stakeholders also control this. Until the environment is shut down or any of the termination criteria is not met, all the received log events are processed to achieve maximum. Initially, a log event is collected, parsed, and stored. Every log entry is tagged with respect to its structure, and JSON documents are populated and stored. Further, it is checked whether it is an administrative or a security event or not. If it is found as an administrative or a security event, it is published to the blockchain, distributed among the stakeholders; otherwise, it is processed locally at the gateway for the local fog environment.

Algorithm 1. BCALS implementation

```
inputFogEnvironment $F_o$, GatewayMonitoringSystem $G_o$,
Security_AdministrativeEvents $SE_o$, LogDataForTheEvent $(e_o)$ $LD_o$,
SemanticsAwareLogData $SALD_o$, PermissiveBlockchainForFog$(F_o)$ $BC_o$
Procedure
  $F_o$ = InitializeFog(); // Fog computing environment
  $G_o$ = GatewayMonitoringSystem($F_o$); // Gateway System
  while termination criteria is not met do
      e = g(f) //Event e occurred in Fog environment($F_o$) monitored by gateway ($G_o$)
      $LD$ = LogData(e) // get log data for the specific event (e)
      $SALD$ = GenerateSematicAwareLogData($LD$) // Semantic Transformation
      Process ($LD$) // process & store log data at Gateway System
      // If (e is an administrative/security event) then // Log Classification
      if e is in $SE_o$ then
          addToBlockchain( $BC_o$, $SALD$) // Add SALD to Blockchain BC
          DistributeAndMonitor($BC_o$)
      else
          LocalProcess(e) //Save Log Event to the Local Log Store
      end if
  end while
End Procedure
```

Algorithm 2. InitializeFog() returns an active Fog Pointer F_o

```
Input
FogEnvironment $F_o$, GatewayMonitoringSystem $G_o$;
Procedure
  $F_o$ = PowerOnFogUnits(); // Start Fog Units
  EstablishConnectionsWithFogUnits( $F_o$ ); // Gateway System
  UpdateFogPeerNetwork( $F_o$ );
  return $F_o$
End Procedure
```

Algorithm 3. GateWayMonitoring() returns connection information of the Edge Devices

```
Input
FogEnvironment $F_o$, GatewayMonitoringSystem $G_o$;
Procedure
while the fog environment is active do
    CheckPoweredOnFogUnits(); // Check for the availability of Fog Units
    CheckEstablishedConnectionsWithFogUnits(); // Gateway System
    UpdateFogPeerNetwork();
    Collect_Fog_events_and_forward_to_BCALS();
    if an_error_occurred then
        Alert all the stakeholders
    end if
end while
return Fog_Connections_Status
End Procedure
```

Algorithm 4. GenerateSemanticsAwareLogData(EventLogData e, LogStructure s) returns SemanticsAwareLogData LogEntry

```
Input
EventLogData e;
Procedure
LogEntry logEntry = new LogEntry(e,s);
logEntry.EventId = e.eventID;
logEntry.Severity = e.TraceEventTypeInformation;
logEntry.Message = String.Format(e);
logEntry.RuleName = getcontext.RuleName();,
logWriter.Write(logEntry);
End Procedure
```

The fog initialization process has multiple activities as listed in Algorithm 2. The first activity is powering on edge devices or fog units. When the powering on the process is completed, their interconnectivity and intrafog connectivity with respect to a fog unit are established and published on the network.

Gateway monitoring is also an important function carried out by a highly privileged process at the gateway. This process guarantees the availability of all the required services and connections among the fog units. The details of this function are explained in Algorithm 3.

Semantic aware log data generation is the process of tagging all the values available in a log entry. Log entries are of different types means these are heterogeneous with respect to their structure or elements. Algorithm 4 describes its functionality and how a log entry is transformed into semantic rich JSON documents with respect to their structure. These JSON documents are available for any type of LoD processing and relationship finding.

5 | EXPERIMENTAL EVALUATION

The proposed framework uses gateway-based log collection, semantics-based transformation, and DLT for immutable logging systems to mitigate the insider threat. Log filtration and semantics transformation are carried out at the gateway to avoid unnecessary processing and communication of data, and semantics transformation contributes toward log analysis and forensics. BCALS framework has been implemented in a data center of a software house. The real effectiveness and precision of the proposed system were found very effective and evaluated in the following sections.

Along with the provision of security requirements, performance in terms of log storage and retrieval, storage capacity, and response time are also considered for evaluating this framework. As a benchmark, the proposed model has been compared with existing schemes proposed in References 14,35,36. In addition to that, security analysis in terms of Confidentiality, Integrity, Availability, and Authenticity has been compared with References 14 and 35 and Performance evaluation in terms of Reference 14. Furthermore, validation using formal methods is also possible, which is being carried out. In this section, we explore the practicality of BCALS. Along with the provision of security requirements, performance in terms of log storage and retrieval, storage capacity, and response time will be considered to evaluate this framework. The proposed framework uses gateway based log collection, Semantics-based transformation, and DLT for immutable logging system to mitigate the insider threat. Log filtration will be carried out at the gateway to avoid unnecessary data processing. Communication and semantics transformation will contribute to log analysis and forensics.

5.1 | Experimental setup

For a Proof of Concept (PoC), We have deployed BCALS using two virtual machines, one as BCALS Host and the second as an additional separate Elasticsearch Host. These virtual machines are hosted on CentOS 8, having a XEON processor and 16 GB RAM using Oracle VirtualBox Version 6.1.4. Both the virtual machines are symmetrical in virtual resources. The resources allocation to the virtual machines is as under.

Hardware Resources configured for the implementation and testing of the proposed BCALS include a Dual Core CPU with 100 % Execution Capacity, 4 Gigabytes for Random Access Memory, 20 Gigabytes HDD for installation and storage, and a bridged network adopter.

Software Resources used to implement and test BCALS in a virtual environment using Ubuntu 16.04 operating system for the above-discussed hardware resources. Multichain 2.0.0 is used for blockchain implementation, which is based on The ECDSA. We used Elasticsearch with Kibana in a Firefox 73.0.1 browser interface for storage and query of JSON documents.

5.2 | Implementation

There are a lot of logs being generated by the systems. In this research, we used authentication logs of a Ubuntu operating system. These logs are parsed and stored in multichain streams having uniquely identifiable stream keys for immutability. Furthermore, these logs are transformed into JSON documents and stored in Elasticsearch to find semantics among the user activities. Hence, implementation of BCALS is divided into the following four subprocedures as (1) Log Parsing and Heterogeneity Support, (2) JSON Creation of Logs, (3) Publishing to the Blockchain, and (4) Sending parsed data to Elasticsearch. These activities are discussed in the following sections.

5.2.1 | Log parsing and heterogeneity support

Since different log files have different structures; therefore, to explore each log entry, BCALS adds support for parsing log entries with respect to that structure and prepares data for semantics enrichment. Here the structure for authentication type log entry of Ubuntu Operating system is given in Equation (1).

$$\mathrm{LE} = (\mathrm{TS}, \mathrm{MN}, \mathrm{Act}, \mathrm{Text}), \tag{1}$$

whereas the variables in this equation are:-

TS =TimeStamp

MN =Machine or Computer Name

Act = Activity performed

Text =Detailed Message in log Entry

In this way log files are parsed and each log entry is viewed with reference to given structure.

5.2.2 | JSON creation of logs

After parsing log entries into the respective structure, JSON documents are prepared using the given structure so that it can be further published to Blockchain and Elasticsearch. Administrative logs are published to Blockchain and Elasticsearch both, whereas all other types of logs are published to Elasticsearch only.

$$JSONDoc = JSON.Convert(Struct, LE). \tag{2}$$

5.2.3 | Publishing to the blockchain

Blockchain connection is required once before publishing data to the blockchain is established as given in Equation (3).

$$\text{BCALS} = \text{Multichain(USER, PWD, HOST, PORT, BCName)} \tag{3}$$

where the variables in this equation are:-

USER = permitted user id
PWD = Password to connect Blockchain
PORT = Computer Port No on which blockchain host is listening
BCName = Name of the Blockchain to which it is being connected.

After having a successful connection, Log entry is published to the Blockchain as following.

$$\text{Result} = \text{BCALS.publish(STREAM, Key, Data)}, \tag{4}$$

where the variables in this equation are:-

STREAM = Stream of Logs in which Log entry is to be appended.
Key = Unique identification of Log Entry.
Data = Log Entry as JSON Doc to be published in BCALS Blockchain.
RESULT= Acknowledgment received after publishing to the BC.

5.2.4 | Sending parsed data to Elasticsearch

We have worked on two variants of BCALS, in first scenario, Elasticsearch is available on the same machine, whereas in second scenario, another independent machine is configured for Elasticsearch for load balancing.

Similar to Blockchain host, we need to connect to Elasticsearch host once before sending data to it as following as Equation (5) with minimum host ip and port number:-

$$\text{ES} = \text{Elasticsearch(HOST, PORT)}. \tag{5}$$

After establishing a successful connection, JSON documents are indexed in Elasticsearch for fast correlation of activities as following Equation (6):-

$$\text{RESULT} = \text{ES.index}(index_name, doc_type, \text{JSONDOC}) \tag{6}$$

5.3 | Results

After successful implementation of BCALS and to evaluate the performance, we figured out different scenarios. As a benchmark, BCALS is compared with PISB and SecLaaS scheme.

Authentication logs dataset of a Ubuntu system has been used in this scenario. Computational overhead in terms of time to process the logs is considered for comparison; that is why time is the only parameter for this comparison to process these logs for a given number of entries under a similar environment. Plots for evaluation of BCALS has been drawn in the following section. These forward signature-based schemes calculate hashes of the data blocks and append them accordingly where whole data integrity is not considered. It is very pertinent to note that BCALS took less time than SecLaaS but slightly more time than PISB because PISB is providing immutability for block signature only; in contrast, BCALS delivers much more than that.

5.4 | Evaluation of performance

Initially, we deployed BCALS for a limited number of log entries, and experiments were carried out for 2000, 4000, 6000, 8000, 10 000 and 12 000 log entries, respectively. The performance comparison is shown in the following plot given in Figure 3. The detailed plots are also prepared to evaluate time for each step are as follows: (i) Text Parsing Time arsing (ii) JSON Creation (iii) Adding to Blockchain, (iv) Publishing to Elastic Search (On Same Machine)

5.4.1 | Preliminary processing - Parsing text and semantic enrichment (JSON Creation)

In this preliminary processing phase, log entries are parsed with respect to the given structure. After parsing the log entries with respect to the given structure, these entries are enriched with semantics by tagging respective information. JSON documents are created to store in the Blockchain and further to the Elasticsearch. The plot showing text parsing, JSON creation of log entries, and this process's total time can be observed at preliminary processing Figure 4.

5.4.2 | BC-ES Processing - Adding data to the Blockchain and Elasticsearch

After JSON documents are created, these are required to store in the Blockchain for immutability and to publish to the Elasticsearch for linking and relationship finding in the activities. The total computational time required for uploading these log entries to the Blockchain streams and publishing them to the Elasticsearch is shown in Figure 5. The overall

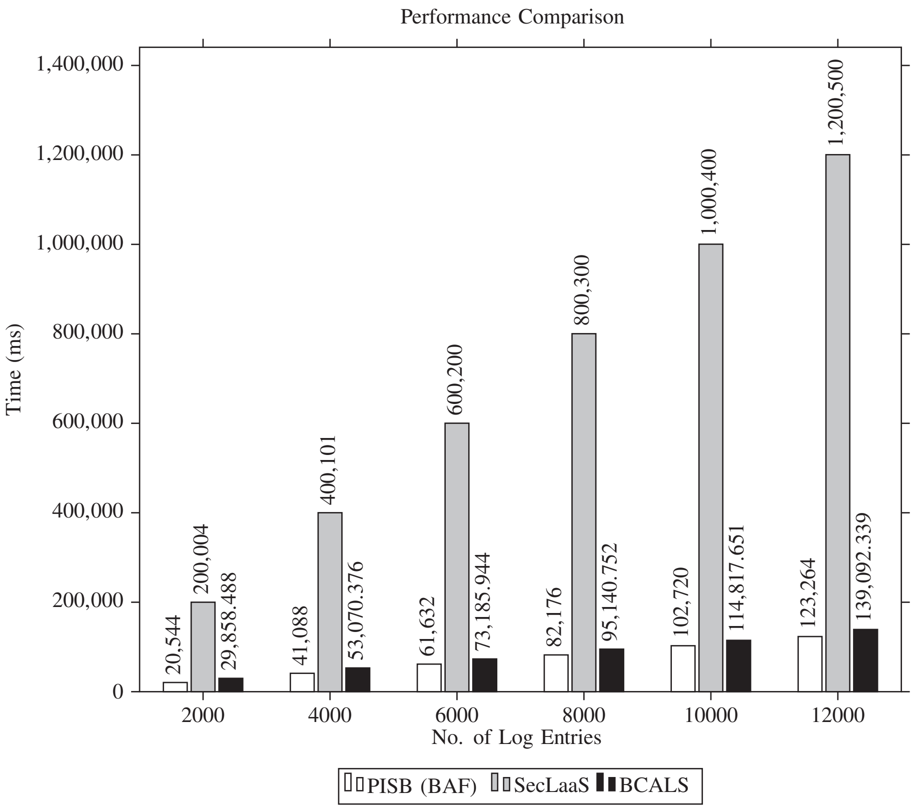


FIGURE 3 Performance comparison with other schemes

FIGURE 4 Preliminary processing-basic test

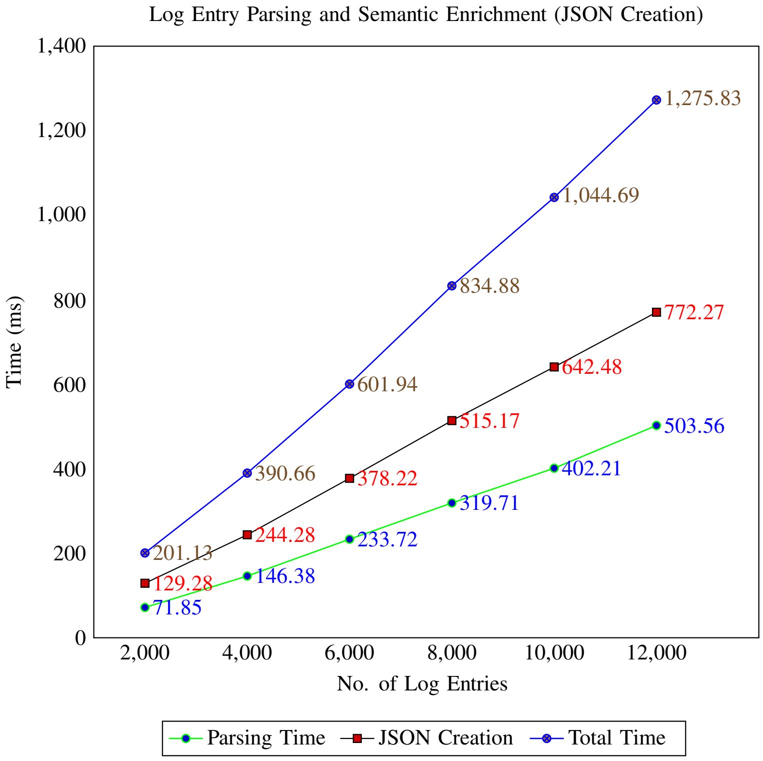
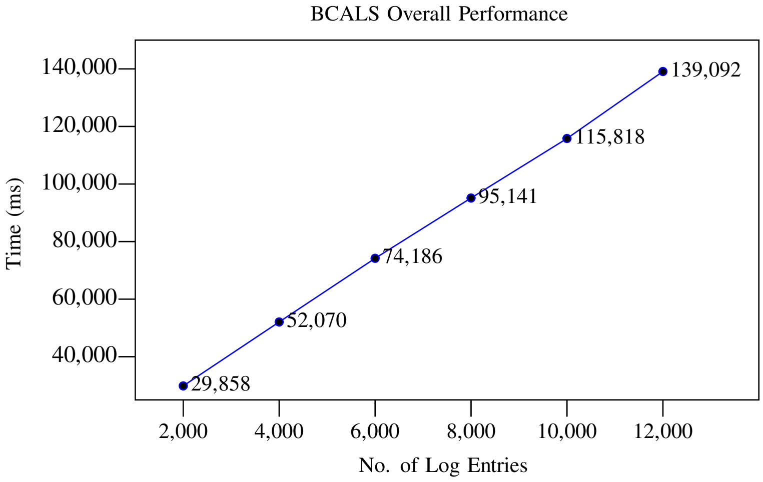


FIGURE 5 Overall performance of BCALS



computational cost in terms of time required for processing these log entries is found very economical as depicted in Figure 5.

5.5 | Evaluation of performance - Stress tests

In order to explore the practicality and suitability, extensive load tests have been carried out. Experiments were carried out for 20 000, 40 000, 60 000, 80 000, 100 000, and 120 000 log entries. Phase wise time required for these experiments is as under. In order to explore the practicality and suitability, extensive load tests have been carried out, and the plots are shown for (a) Text Parsing, (b) JSON Creation, (c) Adding to Blockchain, (d) Publishing to Elastic Search (On the Same Machine), (e) Publishing to Elastic Search (On Separate Machine), and (f) Additional Storage Impact for Elasticsearch.

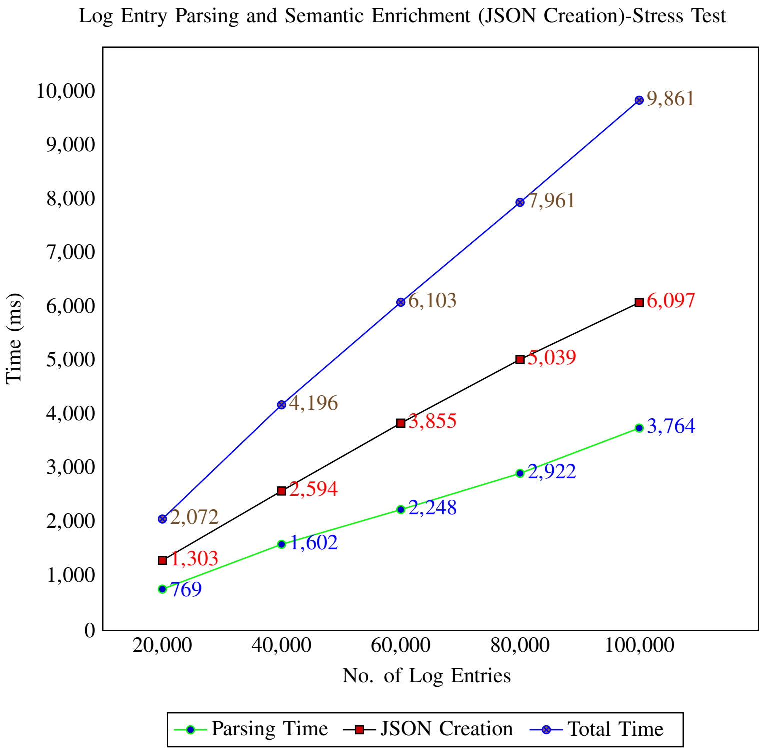


FIGURE 6 Preliminary processing and semantic tagging—Stress test

5.5.1 | Preliminary processing

The computational time required for preliminary processing, like parsing the log entries, the semantic enrichment of different log entries under a stress test of continuous load, is found as follows. The results of these tests are given in Figure 6.

5.5.2 | Overall performance - BCALS stress test

The amazing performance of BCALS has been observed during the stress tests for its initial steps. Nevertheless, the time for storing and indexing the JSON documents took longer time. The results of these test are given in Figure 7.

These JSON documents are stored in the blockchain and Elasticsearch also. Though this is an overhead and additional storage capacity is required but to get semantics and linkage of events among these logs, this is mandatory. Here, it is observed that Elasticsearch is taking a long time as compared to its predecessor processes like text parsing and JSON creation. Since these JSON documents can be published on a separate independent machine for load sharing, BCALS was tested with a separate Elasticsearch machine also. It took more time in this case, and a network latency was observed here. The computational time required for processing these log entries is given in Figure 8.

5.5.3 | Elasticsearch storage overhead

Since log entries are stored in the Blockchain and the Elasticsearch, this causes an additional storage overhead, shown in the following graph as Figure 9.

6 | SECURITY ANALYSIS OF BCALS

Overview, implementation, and results of BCALS have been discussed in previous sections. The prime requirements of BCALS are information security requirements along-with immutability, real-time statistics sharing, coalition, and

FIGURE 7 Publishing Documents to Elasticsearch - Overall performance

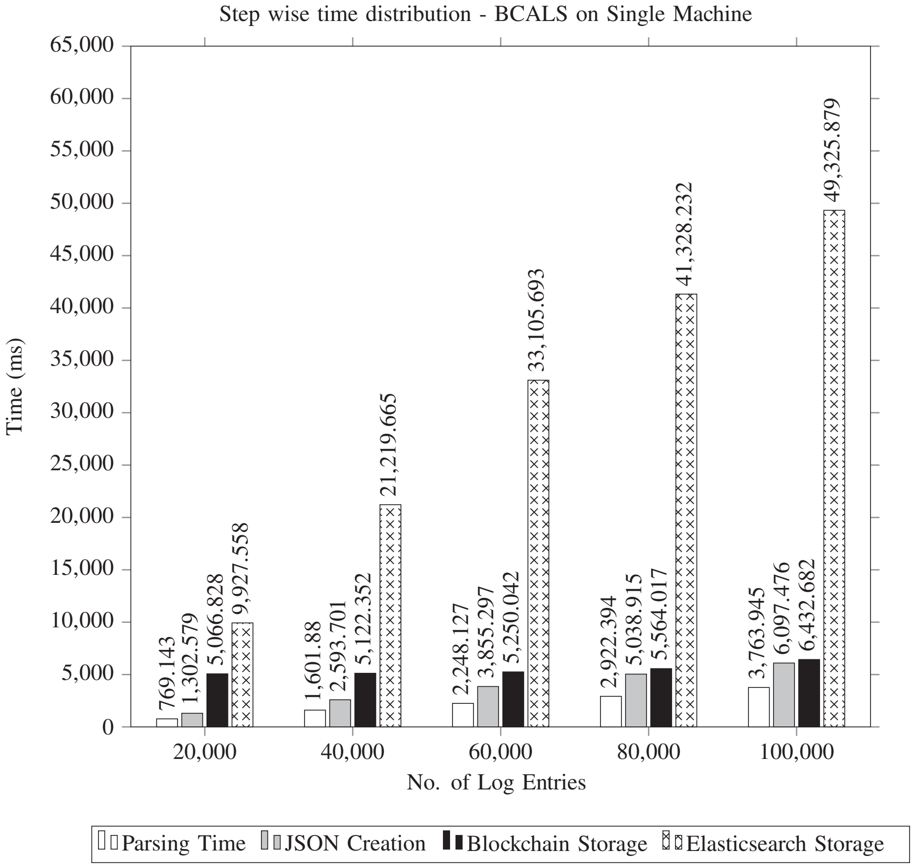
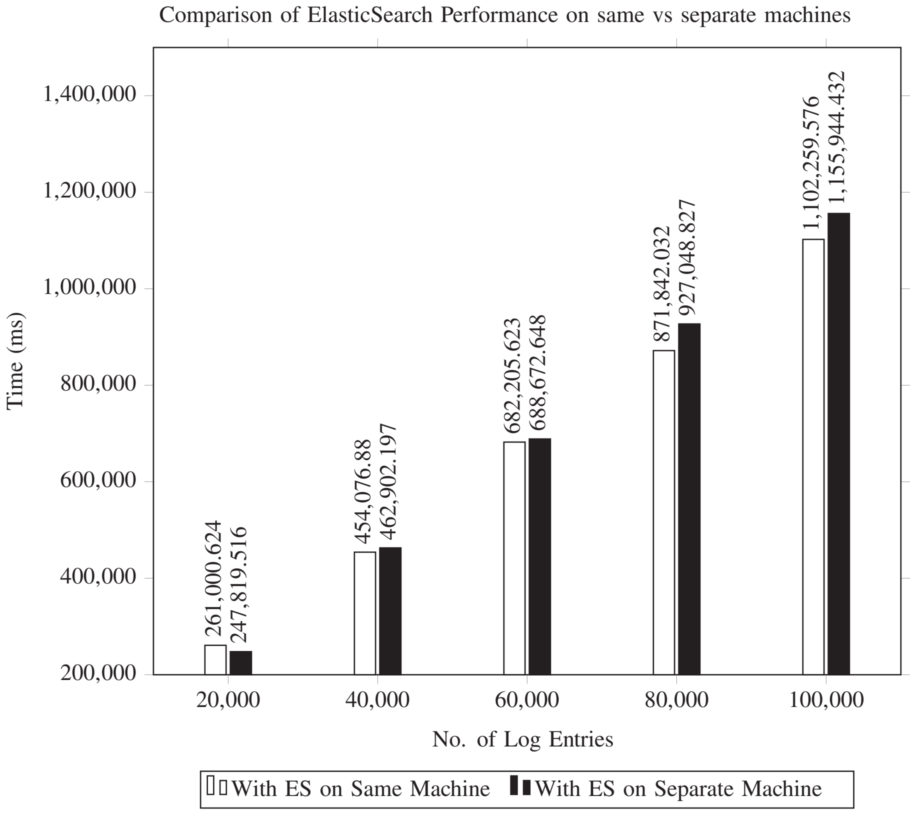


FIGURE 8 Publishing Documents to Elasticsearch—Overall performance



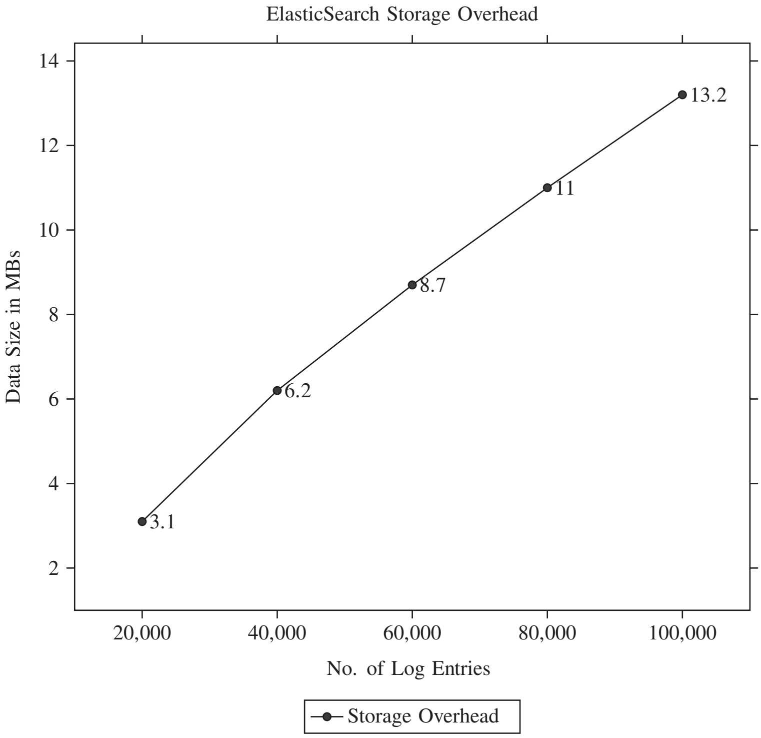


FIGURE 9 Elasticsearch Storage Overhead—Stress test of BCALS

analysis of user activities in a computing environment in a semantics enriched format where logs are in different formats. Blockchains are proven immutable distributed ledgers. Instead of using a public or consortium blockchain, multichain is used, a private blockchain implementation of Bitcoin core protocol with Elliptical Curve Digital Signature Algorithm (ECDSA). In the implementation of BCALS, we have benefited the computing environment by the in-built integrity and immutability features of blockchains. The critique on the BCALS is covered as follows:-

- **Confidentiality.** To provide data confidentiality, blockchain has been implemented with public key infrastructure using the RSA algorithm. Since this is an implementation of private nature blockchains, so data is available to permitted peers only.
- **Integrity.** Data integrity in blockchain has been provided by hashing algorithms, an inbuilt feature of any distributed ledger.
- **Availability.** Redundancy is provided to have the availability of data in any circumstance or situation. In BCALS implementation, peers are redundant nodes for data availability.
- **Immutability.** Data immutability in blockchains is provided by their respective hashing (SHA256) and consensus algorithms. Multichain provides an additional feature of εData Streams.ε These streams can be organized for different types of logs or organized for perpetual and periodic structures.
- **Statistics sharing.** Here, a permissioned distributed ledger is configured, in which peer nodes have real-time access to the data published in the blockchain.
- **Logs heterogeneity and semantics enrichment.** A log's schematic structure has been defined for each type of logs. JSON documents are populated to make them machine-understandable and further published to Elasticsearch for investigating relationships among different activities.

7 | CONCLUSION AND FUTURE WORK

In this paper, we have developed an immutable log management system using distributed ledgers. Along with the immutability, we have used Elasticsearch to store JSON documents: semantics enriched and machine-understandable. This system has been implemented in a software house and found it very useful to monitor the insiders and share statistics with the team leaders. BCALS can be implemented in a homogeneous environment (with satisfactory trust level b/w peers) in general and can also be utilized in a heterogeneous environment (with compromised trust level b/w peers) specifically. We reckon that the application of BCALS in a scalable environment can be planned in the future to prove the efficacy of the framework. Furthermore, Lo semantics of BCALS can also be ingested for an understandable and context aware search mechanism of logs.

Possible future extensions could therefore involve advancing the operation of bots through machine learning techniques to identify malicious users and devices at the fog level through log analysis. Similarly, by combining the real-world scenarios and experimentation, the design and implementation of a real-world BCALS prototype could be a promising path. To cover the log data of several services, this would not only guarantee but also describe the different log formats.

ACKNOWLEDGEMENTS

This research has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No. 801522, by Science Foundation Ireland and co-funded by the European Regional Development Fund through the ADAPT Centre for Digital Content Technology grant number 13/RC/2106_P2.

DATA AVAILABILITY STATEMENT

Data sharing not applicable to this article as no datasets were generated or analyzed during the current study.

REFERENCES

1. Khan MA, Salah K. IoT security: review, blockchain solutions, and open challenges. *Futur Gener Comput Syst*. 2018;82:395–411.
2. Stojmenovic I, Wen S. The fog computing paradigm: scenarios and security issues. Paper presented at: Proceedings of the 2014 Federated Conference on Computer Science and Information Systems. Warsaw University of Technology, Pl. Politechniki 1, 00-661 Warsaw, Poland; 2014:1-8; IEEE. http://dx.doi.org/10.15439/2014F503.
3. Seth B, Dalal S, Jaglan V, Le DN, Mohan S, Srivastava G. Integrating encryption techniques for secure data storage in the cloud. *Trans Emerg Telecommun Technol*. 2020;e4108. https://doi.org/10.1002/ett.4108.
4. Ali A, Ahmed M, Khan A, Ilyas M, Razzaq MS. A trust management system model for cloud. Paper presented at: Proceedings of the 2017 International Symposium on Networks, Computers and Communications (ISNCC) 2017. Marrakech, Morocco; May 16, 2017:1-6; IEEE.
5. Qiu L, Zhang Y, Wang F, Kyung M, Mahajan HR. *Trusted Computer System Evaluation Criteria*. National Computer Security Center, Citeseer Princeton, NJ: 1985:1-129.
6. Ye F, Zheng Y, Fu X, Luo B, Du X, Guizani M. TamForen: a tamper-proof cloud forensic framework. *Trans Emerg Telecommun Technol*. 2020;e4178. https://doi.org/10.1002/ett.4178.
7. Holt JE. Logcrypt: forward security and public verification for secure audit logs. Paper presented at: Proceedings of the ACM International Conference Proceeding Series. Hobart, Tasmania, Australia; 2006:203-211; Australian Computer Society, Inc.
8. Yavuz AA, Ning P. Baf: an efficient publicly verifiable secure audit logging scheme for distributed systems. Paper presented at: Proceedings of the 2009 Annual Computer Security Applications Conference. Sheraton Waikiki Hotel, Honolulu, Hawaii; 2009:219-228.
9. Yavuz AA, Ning P, Reiter MK. BAF and FI-BAF: efficient and publicly verifiable cryptographic schemes for secure logging in resource-constrained systems. *ACM Trans Inf Syst Secur*. 2012;15(2):9.
10. Kampanakis P, Yavuz AA. BAFi: a practical cryptographic secure audit logging scheme for digital forensics. *Secur Commun Netw*. 2015;8(17):3180-3190.
11. Ma D, Tsudik G. Forward-secure sequential aggregate authentication. Paper presented at: Proceedings of the 2007 IEEE Symposium on Security and Privacy (SP'07) 2007 May 20. Berkeley, CA; 2007:86-91; IEEE.
12. Hartung G, Kaidel B, Koch A, Koch J, Hartmann D. Practical and robust secure logging from fault-tolerant sequential aggregate signatures. Paper presented at: Proceedings of the International Conference on Provable Security 2017 October 23, 2017:87–106; Springer, Cham, Switzerland.
13. Pulls T, Peeters R. Balloon: a forward-secure append-only persistent authenticated data structure. Paper presented at: Proceedings of the European Symposium on Research in Computer Security 2015 September 21, 2015:622-641; Springer, Cham, Switzerland.
14. Zawoad S, Dutta AK, Hasan R. SecLaaS: secure logging-as-a-service for cloud forensics. Proceedings of the 8th ACM SIGSAC Symposium on Information, Computer and Communications Security 2013 May 8, 2013:219-230; ACM, New York, NY.
15. Khan A, Yaqoob A, Sarwar K, Tahir M, Ahmed M. Secure logging as a service using reversible watermarking. *Proc Comput Sci*. 2017;110:336-343.
16. Forcher B, Agne S, Dengel A, Gillmann M, Roth-Berghofer T. Semantic logging: towards explanation-aware das. Paper presented at: Proceedings of the 2011 International Conference on Document Analysis and Recognition 2011 September 18. Beijing, China; 2011:1140-1144; IEEE.
17. Shafiq MO. *Semantically Formalized Logging and Advanced Analytics for Enhanced Monitoring and Management of Large-scale Applications* [PhD thesis]. University of Calgary; 2015. https://prism.ucalgary.ca/handle/11023/2225.
18. Accorsi R. BBox: a distributed secure log architecture. Paper presented at: Proceedings of the European Public Key Infrastructure Workshop 2010:109-124; Springer, Berlin, Heidelberg.

19. Von Eye F, Schmitz D, Hommel W. *SLOPPI-A Framework for Secure Logging with Privacy Protection and Integrity*. Princeton, NJ: Citeseer; 2013:14-19.
20. Rajalakshmi JR, Rathinraj M, Braveen M. Anonymizing log management process for secure logging in the cloud. Paper presented at: Proceedings of the 2014 International Conference on Circuits, Power and Computing Technologies [ICCPCT-2014]; March 20. Nagercoil, India; 2014:1559-1564; IEEE.
21. Henze M, Wolters B, Matzutt R, Zimmermann T, Wehrle K. Distributed configuration, authorization and management in the cloud-based internet of things. *Proceedings of the 2017 IEEE Trustcom/BigDataSE/ICESS*. Sydney, Australia: IEEE; 2017:185-192.
22. Jaquette GA, Jesionowski LG, Kulakowski JE, McDowell JA. Low cost tamper-resistant method for write-once read many (WORM) storage. US Patent 6,272,086; 2001. https://patents.google.com/patent/US6272086B1/en.
23. Sinha A, Jia L, England P, Lorch JR. Continuous tamper-proof logging using tpm 2.0. Paper presented at: Proceedings of the International Conference on Trust and Trustworthy Computing 2014 June 30, 2014:19-36; Springer, Cham, Switzerland.
24. Shepherd C, Akram RN, Markantonakis K. EmLog: tamper-resistant system logging for constrained devices with TEEs; 2017. arXiv preprint arXiv:1712.03943.
25. Ko RK, Jagadpramana P, Lee BS. Flogger: a file-centric logger for monitoring file access and transfers within cloud computing environments. Paper presented at: Proceedings of the 2011IEEE 10th International Conference on Trust, Security and Privacy in Computing and Communications 2011 November 16, 2011:765-771.
26. Cucurull J, Puiggali J. Distributed immutabilization of secure logs. Paper presented at: Proceedings of the International Workshop on Security and Trust Management 2016 September 26. Heraklion, Crete, Greece; 2016:122-137; Springer, Cham, Switzerland.
27. Sutton A, Samavi R. Blockchain enabled privacy audit logs. Paper presented at: Proceedings of the International Semantic Web Conference; 2017:645-660; Springer, Cham, Switzerland.
28. Yavuz AA, Ning P, Reiter MK. Efficient, compromise resilient and append-only cryptographic schemes for secure audit logging. Paper presented at: Proceedings of the International Conference on Financial Cryptography and Data Security; 2012:148-163; Springer, Berlin, Heidelberg.
29. Ali A, Ahmed M, Ilyas M, Küng J. MITIS-an insider threats mitigation framework for information systems. Paper presented at: Proceedings of the 3rd ACM Symposium on Cloud Computing; 2017:407-415; ACM, New York, NY.
30. Reiss C, Tumanov A, Ganger GR, Katz RH, Kozuch MA. Heterogeneity and dynamicity of clouds at scale: Google trace analysis. Paper presented at: Proceedings of the 3rd ACM Symposium on Cloud Computing; 2012:7; ACM, New York, NY.
31. Sato T, Himura Y, Yasuda Y. *Evidence-Based Context-Aware Log Data Management for Integrated Monitoring System*. Kanazawa, Japan: IEEE; 2016:1-4.
32. Bellare M, Yee B. *Forward Integrity for Secure Audit Logs. Technical Report*, Computer Science and Engineering Department, University of California at San Diego; San Diego, CA: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.111.6973. 1997.
33. Khan S, Gani A, Wahab AWA, et al. Cloud log forensics: foundations, state of the art, and future directions. *ACM Comput Surv*. 2016;49(1):7.
34. Kent K, Souppaya M. Guide to computer security log management. *NIST Special Publ*. 2006;92:1-72.
35. Yavuz AA. Immutable authentication and integrity schemes for outsourced databases. *IEEE Trans Depend Secure Comput*. 2018;15(1):69-82.
36. Jin H, Zhou K, Luo Y. A framework with data-centric accountability and auditability for cloud storage. *J Supercomput*. 2018;74(11):5903-5926.

How to cite this article: Ali A, Khan A, Ahmed M, Jeon G. BCALS: Blockchain-based secure log management system for cloud computing. *Trans Emerging Tel Tech*. 2022;33:e4272. https://doi.org/10.1002/ett.4272