Logging Integrity with Blockchain Structures

Marco Rosa[1], João Paulo Barraca[1][0000-0002-5029-6191] and Nelson Pacheco Rocha[1][0000-0002-3801-7249]

[1] Universidade de Aveiro, Campus Universitário de Santiago, Aveiro, Portugal

{marcofrosa, jpbarraca, npr}@ua.pt

Abstract. In developed countries, it is frequent for family members do not have the time, knowledge, or live in a close distance of their senior loved ones, so that many institutions offer their services to provide a good quality of life of older adults. To enable distributed local support, there is the need of digital platforms to allow the exchange of information. These platforms need to create trustful environments and to guarantee the integrity of the information exchanged. In this paper, it is presented a solution for a Logging Service that was developed for the SOCIAL platform, based on FHIR, which aims to solve the interoperability and data integrity of the platform user's activity logs.

Keywords: Logging, Integrity, Auditing, Blockchain, FHIR.

1 Introduction

Health care provision, once focused on the management of acute conditions, is increasingly shaped by the epidemiological transition towards Non-Communicable Diseases (NCD), which are related to the main causes of mortality [1]. Also, there is evidence related to disease burden and loss of economic output associated with NCD, mainly cardiovascular diseases, oncological diseases, chronic respiratory diseases, diabetes and neurodegenerative diseases [2].

Most of these NCD last for long periods of time, may have a chronic nature and progress slowly. Consequently, the fragmented nature of the health care systems, focused on the management of acute conditions, is unable to provide universal, equitable, high-quality and financially sustainable care [3]. This is an opportunity to move from organization-centered care to a paradigm focused on the needs of the care receivers [4]. Integrated care approaches are required, not solely focused on medical purposes, but also on a range of essential activities for the maintenance of the individual's quality of life, which increases the complexity of the care services, as a consequence of shared procedures between different organizations and a trend towards community care, including, in addition to health care provision, by medical and nursing staff, social care provision, by social care staff, or home assistance services [5].

The platform of services SOCIAL [6] aims to surpass the so-called health and social care divide [7]. It prides information services to support integrated care and assistance of community-dwelling older adults in scenarios of care provided by a distributed environment of cooperating entities, which is typical for (at least) many European regions.

In this respect, trust and the promotion of trust assume a paramount importance. To promote trustful relationships between all the stakeholders of the SOCIAL platform a major issue is to ensure the integrity of the activity logs that should be traceable by auditing processes. The study reported by the present paper focused on the information audit trail and the mechanisms to promote its integrity in distributed environments.

In the following sections, the paper will introduce the related work that is relevant to understand the reasoning process behind the proposed Logging Service, a description of the SOCIAL platform and its logging requirements, the presentation of the proposed solution, an evaluation of its features and a conclusion.

2 Related Work

Because of the distributed nature and low connection between care providers in the community care context, the policies, rules and processes are frequently difficult to be fully implement or adopted. However, this doesn't alleviate the need for formal access control and the creation of audit trails to be validated by auditors.

An auditor can be someone from the inside or the outside of the auditing target and an audit should be divided into several stages, including planning or gathering of data and its analysis. Analyzing logs allows an auditor to see and evaluate the activities of the users for security purposes (e.g. to verify if someone tried to log in several times in short period of times or attempted to perform non-authorized information accesses). In the end, the results should indicate that everything is going according to the plan or lead to modifications to improve the efficiency of the security mechanisms.

When developing a Logging Service (i.e. a service to record and retrieve the activities of the different users), the simplest procedure is to store the logging messages in raw format inside a file. However, by only doing that, if there are different entities contributing with their logs (each one with a different structure), the log file will be disorganized, unnecessarily complex, and would make the tracking and analyzing process harder to execute. At the same time, obeying protection regulations (e.g. EU Directive 2016/679 on the protection of individuals) becomes more difficult.

At the application level, there are popular solutions, some adapted to the use case of electronic health records [8], which facilitate the production of effective logs by exploiting careful, sometimes automated, log placement [9]. These solutions make it possible to specify the service or component producing the log entry, the event time,

and a message, while associating well known severity levels. The destination is frequently a log file, but some solutions adopt the use of document-based databases [10]. However, in either case, the stored information remains vulnerable to tampering.

Over time, there have been several applications that improve the generation and management of logging records. Retrace [11] stores all the log data on a centralized location and provides a range of tools (e.g. error tracking, performance analysis or detailed log searching). It does not solve the tampering problem, since all logging data is stored in the same place, and there is not a way to verify if someone modified an entry. Alternative log management tools suffer the same problems, such as LogRhytm [12] or Splunk [13], because their main concern is to store data and retrieve entry quickly and efficiently, without analyzing possible integrity problems.

A centralized logging service, located in an area of restricted access, is useful for the case of log integrity, as it can handle different system logs, from remote systems, exposing a read only access to the stream. While this solution, that is typical of many deployments, provides some assurance by means of barriers, it provides no cryptographic support for integrity.

There are some proposed solutions for logging services with integrity control, for different type of use cases. For example, approaches to guarantee integrity of service's logs include: adding markers to the log entries, such as sequence numbers and epoch, to detect reordering attacks [14]; adapting a Merkle-tree by using chameleon hash functions instead of regular hash functions [15]; using a key pair (secret and public) where the secret key is used to generate a signature for the log requests and the public key for auditors to validate the signature and integrity of the log data [16]; adapting the blokchain technology into a 2-layer blockchain architecture (one layer for fast operations and other for heavy operations) where a block's hash can prove the logs integrity [17]; using a domain-specific language (DSL) that can allow to structure a log and to select the security parameters that can guarantee its integrity [18]; or using a homomorphic encryption algorithm, where only the one that sends the encrypted logs knows how to decrypt them and understand their contents [19].

In recent years, the development of blockchain technology, mostly famous through its many digital currencies, provided the means to create distributed ledgers, that can validate the chronological consistent of a series of actions. Its inherent capability of validating the temporal order of events also makes the solution adequate for integrity control of log entries.

As its name indicates, blockchain is constructed by several data blocks that are chained, using a process named hash chain. Inside each block, there are several attributes, namely the data (or payload) itself, a timestamp, a hash (a unique identifier or a digital digest) and the hash from the previous block. These hashes are the key attributes to confirm the data integrity, because each block is dependent on the previous (and/or next) and if someone want to tamper one block it would need to

tamper every single existent block. Because the chain exists in multiple systems, this would make the intrusion very easy to detect. The blockchain also differs from other data storage implementations by not having intermediaries able to handle all the data related to activity logs. Instead, there is a distributed network, in which every participant has access to the relevant data (shared ledger) and contributes to their validation, as each new block added frequently requires the validation of all previous chain. That means that every single block must go through various nodes before being inserted into the shared ledger or exist in multiple replicas [20].

By integrating the blockchain structure with the logging feature of a system, it can be guaranteed that events are not manipulated by an attacker, as the entities manipulating the chain will keep a distributed consensus, built on a trust relationship between the different parties that are involved with the logging. By requesting their validation, thus obtaining a proof that the integrity was verified by a specific entity at a certain time instant. This will avoid certain parties to deny their involvement in the validation process or to tamper the log entries because there is consensus about the log history. It should be considered that timestamping and validation can rely on metadata, as the timestamping of the content fingerprint, effectively timestamps the content itself. This bring the added benefit of not disclosing sensitive information (the log entry) to third parties.

Guaranteeing the integrity of logs is even more relevant in systems that handle health-related information. The blockchain technology has been applied to guarantee the integrity of electronic health records [21, 22]. There are also studies which main goal is integrity by adapting the blockchain technology, but they not consider the integration with interoperability standards such as Fast Healthcare Interoperability Resources (FHIR) [21] or are mainly focused on health care (and not social care) systems [22].

3 Proposed Solution

The SOCIAL platform [6] addresses the care provided in the community, namely supporting informal and formal caregivers who do not have access to structured information services regarding their care receivers, such as the social care and assistance providers, or informal care providers.

Several web and mobile applications are being developed to fulfil the needs of the stakeholders of the SOCIAL platform, including common applications (e.g. registration and self-registration on the platform, insertion and editing of registered data such as demographic data, or notifications and alerts of relevant information, such as tasks to be performed or enrolled activities), care receiver's applications (e.g. requests for assistance, individual care plans, consultation and enrolment in programs being proposed or integration of information of personal health devices), caregiver applications (e.g. management of care plans and tasks) or care management

applications (e.g. administrative management of care receivers and care providers or allocation of human and material resources).

The FHIR was thought to be used to guarantee the internal and external interoperability since it is regarded as a next generation health information interoperability framework and it presents high flexibility, since it is compatible with web standards such as the REST architectural style or the JSON serialization format. Another supporting component of the SOCIAL platform is the Logging Service. To develop a Logging Service that can be integrated with the FHIR standard and able to manage logs from different applications and services, there are some requirements that need to be fulfilled:

Logging Service isolation - a vital requirement is that the execution of the Logging Service should suffer from minimal interference, which imposes that, from the SOCIAL platform point of view, it should be an independent service, accepting log entries from the remaining applications and services. The storage medium should also be isolated, both for security reasons and for exploiting the adequacy of a specific storage backend.

Seamless integration with FHIR services - the Logging Service should be a web service, so that other services can communicate with it and the support for logging doesn't introduce additional dependencies. Since the Logging Service should exchange information, the use of the REST architecture is recommended, due to its performance, scalability and uniform interface. Moreover, integrating the Logging Service with the FHIR framework will allow the processing of data without misinterpretations, and to communicate them in a format that is understood by different parties.

Multidimensional reconstruction of the event timeline - data from activity registries must be contextualized so that auditing processes will be able to reconstruct the timeline of any participating entity. This implies the need for timestamps in all log entries, but also implies the identification of the different entities. In the scope of the development of the SOCIAL platform, this results in the definition of several attributes (e.g. time of creation, time of insertion, service identity or user identity).

Resiliency to manipulation - log entries should be stored in a way that it should not be possible to manipulate them. Also, it should not be possible to alter the timeline by simply deleting entries from the database.

Resiliency to arbitrary losses - if any of the log entries are lost during an attack or other type of issues. These data should be available elsewhere and ready to be quickly retrieved. Proof that the new data are not manipulated is also required.

Non-repudiation - it should be possible to request a validation from the entities that provide their log entries to the Logging Service, so that there can be several integrity proofs accumulated, for future verification.

Privacy and data confidentiality - log entries should contain minimal data required to assert the execution of an action by a user or service, but personal information

should not be directly exposed in the entries. The use of temporary identifiers, and other anonymization techniques is mandatory. At the same time, third parties should be able to validate the timeline, without having access to the content of the log. Full reconstruction and validation of the log content should be possible only to auditors.

In order to integrate a secure logging solution, relying on blockchain technology, to satisfy the defined requirements, the Logging Service architecture is based on a RESTful API (Fig. 1), allowing the communication of the different SOCIAL applications via HTTP POSTs. This service is complemented by a timestamping service (timer) that sends the events triggering the block's insertion process. A cryptographic secure blockchain was adopted, which stores the events and the blocks (of the blockchain) in a special purpose database of the SOCIAL platform. Moreover, audit processes will access the stored events and blocks to determine the relevant data. For that three separate processes were considered: log insertion, log consensus and log audit.

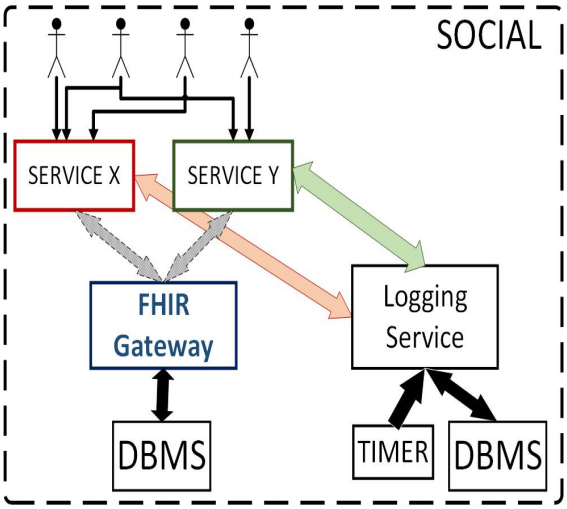

Fig. 1. Logging Service architecture.

Log insertion is triggered when a service event occurs in the SOCIAL platform. The data describing the event is sent to the Logging Service, as an HTTP POST with the event inserted in its request body, as a JSON Object. The Logging Service processes its contents and categorizes the different elements involved in the event (e.g. date, user or application), so that it can construct the appropriate document to be persisted. These requests must also have a JSON Web Token (JWT), which is used to verify signatures (using an algorithm and a secret key previously established) and to validate requested events. The log integrity is validated locally upon block insertion, which happens at fixed intervals, as stated by the timer. Inserting a block requires the validation of the previous blocks in the chain (either by that service or offloaded to

horizontally scalable instances), and the registration of the log entries that are pending to be inserted. This creates interdependencies between log entries from multiple sources, limiting single source spoofing by an adversary. Blocks can have no payload, effectively sealing the chain, proving that no activity was logged during that time interval. If there are data, a cryptographic hash is calculated (e.g. SHA-256), sorted by insertion order, as well as the hash of the previous block and other identification fields. The block is then inserted into a document-based storage, with multiple indexes, using the well-known ElasticSearch indexer. Since the SOCIAL platform main purpose is to coordinate supporting care services, there should be multiple streams in the blockchain (i.e. for each platform application or service and for each user). A global stream can also exist if an absolute timeline reference is desired, which is beneficial for the auditing process, in order to analyze the timeline of events of the users to verify their behaviors or to analyze the timeline of events of an application or service to verify what it is doing. The actual log data is not replicated, and this multi-stream approach is kept by linking digital hashes, and the block payload only contains indirect references. Privacy is inherent to the design as the one-way property of cryptographic digest functions prevents reversing the digest process.

Log consensus is achieved by sending data from the blocks inserted between two wider timeframes, to application or services belonging to the same ecosystem, which should reply with proofs of integrity. Each proof contains a digital signature, which may follow the standard X.509 timestamping practices. It can consist of full logs, allowing the effective replication of the entire event log and block chain in external services. However, only minimal indirect validation is possible, by synchronizing the block chain without actual log data, which is a tradeoff between performance and resilience to failures as well as audit effectiveness.

Log audit is done by direct observation of the event stream stored in the database. Existing tools such as Kibana can be used and the integrity of the data observed can be assured. The integrity of the chain can be verified anytime to check for any inconsistencies. Since every block of the chain has a clear identification, with fingerprints of the log data, as well as timestamps of the first and last event during that time period, the hash of all the events occurred between two checkpoints can be recalculated to check if it matches the one stored in the block. The next and previous hash fields on every block can also be used to verify if the block itself has not been tampered.

4 Evaluation

Besides the actual field validations, which are taking place as part of the planned development of the SOCIAL platform, to validate the proposed solution a simpler scenario was created with the following characteristics: 100 different users accessing four concurrent applications (i.e. an application to support city council's professionals,

an application to support social care providers, an application to support assistance providers, and other application to support care receivers). The Logging Service goal is to handle all the events from all the four applications, while also guaranteeing the integrity of both the application's or service's and user's timelines.

For 60 minutes, it was planned that each application should continuously send events (up to 50), each one linked to a specific user, before stopping for a random period of time (i.e. 45 to 90 seconds for the application to support city council's professionals, 25 to 45 seconds for the application to support social care providers, 10 to 20 seconds for the application to support assistance providers and 40 to 60 seconds for the application to support care receiver's). These events were JSON Objects (from a standard FHIR integration), with different fields to identify the events, such as their users, time of occurrence and the respective applications or services. This was done to evaluate how the Logging Service can handle random situations, whether that be handling multiple events in the same time instant or none during certain time periods (inserting timestamping blocks).

The timer component triggers every 90 seconds, so that the Logging Service verifies if there were new events for each application or service and store the respective highest and lowest timestamps of that set of events. After the timer receives a completion response (showing that the Logging Service finished this stage), it waits a predefined amount of time (20 seconds in this case) and trigger the insertion by the Logging Service, so it creates, and inserts blocks into the respective blockchains (applications, services and users), updating the previous block values (the "nextBlockHash" value). After inserting every block, the Logging Service responds with a status message. After every successful block insertion process, the Logging Service also requests from all parties to validate the new blocks inserted, by sending their hash values to be signed. The signing process being used is the trusted timestamping, where each party concatenates the hash value with the current timestamp and signs it with its private key, and then sends that result, alongside with the timestamp used for the signing to the Logging Service, so it can be stored (Fig. 2).

Table 1 depicts the number of logs and blocks stored by the Logging Service for each application, and if the integrity of the application's timelines was verified. The user's blockchains (that varied between 16 and 19 blocks for each user) was also verified for their integrity. Moreover, it was assured the integrity of the activity logs of the 100 users that were considered for the test.

When an application accesses the auditing feature of the Logging Service, it should receive the events associated to a pair of user and application between two instants. From the Logging Service perspective, whenever it receives an audit request, first, the respective blockchain, with the timestamps between the two time periods requested is retrieved. Afterward, the integrity of those blocks is verified and, finally, if the integrity is validated, the requested events are sent to the platform. During the auditing tests (for application, service or user logs), the Logging Service always sent

the requested events (in these tests, the auditor had privileges to access any event of the platform), which is possible if there were events between those two time periods and if their integrity was verified. The returned events, for every auditing test, are sorted by their timestamp, and it is possible to analyze the respective timeline of events and to carry out an audit with all the necessary data. Since the events follow the FHIR standard, the auditor could filter the events, by any of the fields available in them.

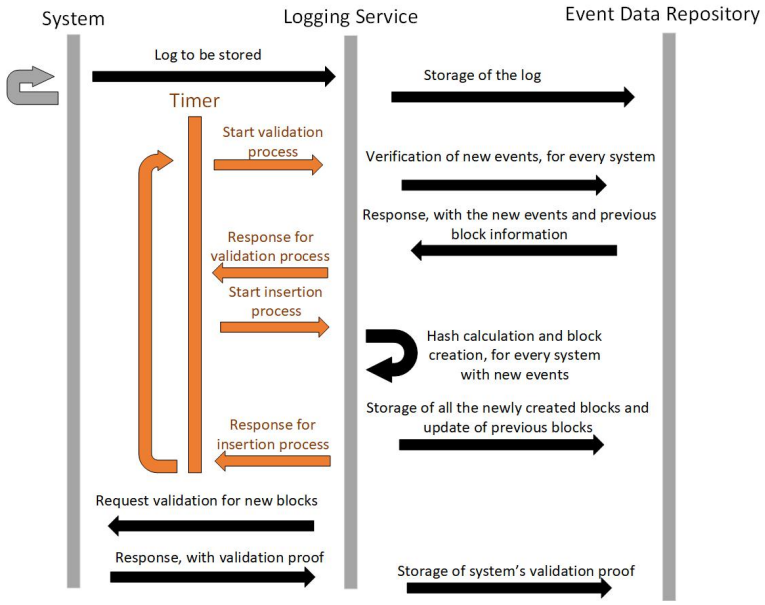

Fig. 2. Logging Service workflow.

Table 1. Service events.

Service	Number of logs (total)	Number of blocks	Integrity verified
City council professionals	921	19	true
Social care providers	1974	19	true
Assistance providers	3751	19	true
Care receivers	1015	19	true

Table 2 shows, the time elapsed for seven different block insertion processes (this also includes the request of the events from the data storage to persist the new blocks and update the previous ones), the total number of events (from all of the different applications) used to calculate the hash and number of blocks created for the user's

and application's blockchains. The first four results were obtained using the parameters established before, while the fifth and sixth result were obtained by increasing the number of events sent by the applications, between the respective two insertion processes, to verify if there were changes when the number of events used to calculate the hash deeply affected the elapsed time. We can see, for our scenario (four applications and 100 users), that the time the Logging Service took to retrieve events, create and update the respective blocks and insert them in the data storage, depends on the number of events, and it can take from 33 seconds to 45 seconds. Therefore, the platform needs to take in account the number of expected events that are going to be exchanged and respective number of users and applications, to determine the interval of time to create and insert new blocks into the blockchain.

Table 2. Block insertion elapsed times.

Number of events	Number of user's blocks	Number of application's blocks	Time elapsed (seconds)
143	81	4	37
417	97	4	33
315	97	4	38
310	97	4	45
2170	100	4	33
3977	100	4	39

The validation proofs (i.e. the trusted timestamping results) were also verified, by using the public key and the hash and timestamp which seals the signed values. Every validation was consistent, which guarantees the block's integrity. Moreover, the signed values cannot be repudiated by the entities that signed them, since the signatures are performed with private keys.

5 Conclusion

Considering the evaluation results, it is possible to conclude that even with large quantities of events in short periods of time, the Logging Service is able to create blocks and manage the respective blockchains for each user and application pair. Consequently, the Logging Service is able to guarantee the integrity of the generated timeline events. Therefore, with the proposed solution, tampering attempts can be efficiently detected in any timeline of events, which means that it is possible to create a trusted environment, since data confidentiality and log integrity are preserved.

The FHIR integration is also beneficial in promoting interoperability. Moreover, the same approach can be used for many other FHIR based instantiations, as it is

based on readily available technology and has configurable levels of external validation and replication.

Acknowledgements

This work was supported by Sistema de Incentivos à Investigação e Desenvolvimento Tecnológico (SI I&DT) of the Programa Portugal 2020, through Programa Operacional Competitividade e Internacionalização and/or Programa Operacional do Centro do FEDER - Fundo Europeu de Desenvolvimento Regional, under Social Cooperation for Integrated Assisted Living (SOCIAL), project number 017861.

References

1. G. Chiarini, G., Ray, P., Akter, S., Masella, C., Ganz, A.: MHealth technologies for chronic diseases and elders: A systematic review. IEEE J. Sel. Areas Commun. 2(9), 6-18 (2013).
2. Abegunde, D.O., Mathers, C.D. Adam, T., Ortegon, M., Strong, K.: The burden and costs of chronic diseases in low-income and middle-income countries. Lancet, 370 (9603), 1929-1938 (2007).
3. WHO global strategy on people-centred and integrated health services. WHO (2015).
4. Blobel, B.: Co-production of health enabled by next generation personal health systems. Stud Health Technol Inform, 177, 52-58 (2012).
5. Hägglund M., Scandurra I., Koch S.: Studying intersection points - An analysis of information needs for shared homecare of elderly patients. J Inf Technol Healthc., 7(1), 23-42 (2009).
6. Sousa, M., Arieira, L., Queirós, A., Martins, A.I., Rocha, N.P., Augusto, F., Duarte, F., Neves, T., Damasceno, A.: SOCIAL platform. In Advances in Intelligent Systems and Computing, 746, 1162-1168 (2018).
7. Rigby, M.: Integrating health and social care informatics to enable holistic health care. Stud Health Technol Inform 177, 41-51 (2012).
8. King, J., Williams, L.: Log your CRUD: design principles for software logging mechanisms. In Proceedings of the 2014 Symposium and Bootcamp on the Science of Security (HotSoS '14), pp.5-15, ACM, Raleigh, North Caroline, USA (2014).
9. Zhao, X., Rodrigues, K., Luo, Y., Stumm, M., Yuan, D., Zhou, Y: The Game of Twenty Questions: Do You Know Where to Log?. In Proceedings of the 16th Workshop on Hot Topics in Operating Systems (HotOS '17), pp. 125-131, ACM, Whistler, Canada (2017).
10. Yang, K.,: Aggregated Containerized Logging Solution with Fluentd, Elasticsearch and Kibana. International Journal of Computer Applications, 150.3 (2016).
11. Stackify, Retrace. [Online]. Available: https://stackify.com/retrace/ (visited on 2018/10/22).
12. Sekharan, S.S., Kandasamy, K.: Profiling SIEM tools and correlation engines for security analytics. In 2017 International Conference on Wireless Communications,

Signal Processing and Networking (WiSPNET), pp. 717-721, IEEE, Chennai, India (2017).
13. Okumura, M., Fujimura, S.: Constructing a Log Collecting System using using Splunk and its Application for Service Support. In Proceedings of the 2016 ACM on SIGUCCS Annual Conference (SIGUCCS '16), pp. 103-106, ACM, Denver, Colorado, USA (2016).
14. Hartung, G.: Secure Audit Logs with Verifiable Excerpts. In Topics in Cryptology - CT-RSA, pp. 183–199. Springer International Publishing (2016).
15. Ning, F. et al.: Efficient tamper-evident logging of distributed systems via concurrent authenticated tree. In 36th IEEE International Performance Computing and Communications Conference, pp. 1-9, IEEE, San Diego, California, USA (2017).
16. Lin, C.-Y. et al.: Secure logging framework integrating with cloud database. 2015 International Carnahan Conference on Security Technology, pp. 13-17, IEEE, Taipei, Taiwan (2015).
17. Aniello, L., Baldoni, R., Gaetani, E., Lombardi, F., Margheri, A., Sassone, V.: A prototype evaluation of a tamper-resistant high performance blockchain-based transaction log for a distributed database. In 13th IEEE European Dependable Computing Conference, pp. (pp. 151-154, IEEE, Geneva, Switzerland (2017).
18. Zawoad, S. et al.: FAL: A Forensics Aware Language for Secure Logging. In Proceedings of the 2013 Federated Conference on Computer Science and Information Systems, pp. 1579-1586, Warsaw, Poland (2013).
19. Rajalakshmi, J. et al.: Anonymizing Log Management Process for Secure Logging in the Cloud. 2014 International Conference on Circuit, Power and Computing Technologies, pp. 1559-1564,IEEE, Kanyakumari, India (2014).
20. Narayanan, A., Bonneau, J., Felten, E., Miller, A., Goldfeder, S.: Bitcoin and cryptocurrency technologies: A comprehensive introduction. Princeton University Press, (2016).
21. Magyar, G.: Blockchain: Solving the privacy and research availability tradeoff for EHR data: A new disruptive technology in health data management. IEEE 30th Neumann Colloquium (NC), pp.135-140, IEEE, Budapest, Hungary (2017).
22. Zhang, P. et al.: FHIRChain: Applying Blockchain to Securely and Scalably Share Clinical Data. Computational and Structural Biotechnology Journal, 16, 267-278 (2018).