TOWARDS SECURE, DECENTRALIZED, AND AUTOMATIC AUDITS WITH BLOCKCHAIN

Research paper

Rafael Belchior, Instituto Superior Técnico, Universidade de Lisboa, Lisboa, Portugal, INESC-ID, Lisboa, Portugal

rafael.belchior@tecnico.ulisboa.pt

André Vasconcelos, Instituto Superior Técnico, Universidade de Lisboa, Lisboa, Portugal, INESC-ID, Lisboa, Portugal

andre.vasconcelos @tecnico.ulisboa.pt

Miguel Correia, Instituto Superior Técnico, Universidade de Lisboa, Lisboa, Portugal, INESC-ID, Lisboa, Portugal

miguel.p.correia@tecnico.ulisboa.pt

Abstract

Organizations are testing the potential of blockchain technology in different areas, bringing implications to record-keeping and the associated business processes, namely audits. Permissioned, private blockchain frameworks can contribute to the processes behind audits by (i) alleviating auditor's work, (ii) hindering fraud and collusion between organizations and auditors, (iii) promote synergies between organizations and their stakeholders, and (iv) protecting access to sensitive information.

This paper studies the use of blockchain technology as an enhancer of secure, distributed, and more automatized audits. The proposed solution uses a permissioned, consortium, blockchain to store audit logs. The blockchain is then leveraged to perform distributed and automatic audits over the recorded logs, aiming to reduce costs associated with audits. In particular, the solution is based on a 4-layer architecture: data acquisition, data transfer, data audit, and data analysis & storage layers. The evaluation shows that the proposed solution is secure and can save and analyse up to around 30 audit log entries per second.

1 Introduction

Nowadays, organizations have *information systems* (IS) supporting many of their business processes. Such information systems produce data from *user activity*, such as accesses to resources. Such accesses are managed by *access control mechanisms* [Ferraiolo et al., 2001], [Pourmajidi et al., 2018] and recorded in *logs*, called *audit logs* in case they are used for auditing purposes.

Logs are used by auditors, cyber-security and forensics experts, and DevOps personnel. Auditors check the organization compliance to financial, legal, and security standards, among others [Bible et al., 2017]. Cyber-security experts normally start by analysing logs when a breach occurs. DevOps personnel uses logs to debug information systems. Logs are important; therefore, organizations have the responsibility of assuring their integrity. Otherwise, malicious entities might, for example, delete their traces from the logs [Chen et al., 2013], [Gaetani et al., 2017].

This paper focuses on the first case mentioned: *audits stored in audit logs, used by auditors*. Current audit log management processes pose some challenges: (i) audit logs are typically saved on conventional databases, thus reflect a centralized client-server model of communication, constituting a single point of failure [Ahmad et al., 2019]; (ii) they may be vulnerable to attacks where adversaries can tamper data, without being detected, possibility deleting their traces (e.g., by modifying or deleting the logs) [Schneier and Kelsey, 1999]; (iii) the analysis of the logs is made *post factum*, sometimes taking a long time for organizations to realize there was a problematic event [Martini and Choo, 2012]; and (iv) there can be distinct stakeholders with different roles, different levels of trust, and different access rights to data, as modern information systems require more dynamic and decentralized access control models [Martins and Guerreiro, 2019].

Trust issues do not disappear when the data used in audits comes from centralized databases. Centralisation poses grave risks of fraud and corruption [Abreu et al., 2018]. In such a scenario, the fraud risk is high because administrators can typically add or update database entries. Furthermore, administrators can access user's data while deleting their traces. The possibility of auditors to collude with administrators exists, if their actions are not recorded in a secure, distributed way. Thus, decentralisation is desirable on a scenario with several stakeholders involved, with different incentives.

Blockchain, the emergent technology introduced in 2008 by Nakamoto [Nakamoto, 2008], is slowly disrupting traditional business processes due to its decentralized nature. Following Bitcoin, Ethereum was created to extend the technology to more general use cases than financial [Buterin, 2014], [Wood, 2014], harnessing the power of smart contracts for powering decentralized apps (DApps). The concept of smart contracts allows translating contractual clauses into code, allowing transactions to be issued in terms of such a contract. Contracts written in a high-level language, such as Solidity, are run on the Ethereum virtual machine (EVM), powering DApps.

Smart contracts enable counterparts to automate tasks that were usually performed by a centralized third party, allowing several use-cases such as data management, data privacy and access control [Casino et al., 2019], [Di Francesco Maesa et al., 2018]. On the contrary of centralized systems, the blockchain's nodes execute distributed code and agree on the state of the shared ledger, ensuring that participants that do not trust the others can trust the content of the ledger. The process that nodes follow in order to agree on the state of the ledger is denominated consensus. This way, untrusted nodes can build a trusted network.

Blockchain is becoming increasingly relevant and more adopted [Xu et al., 2016]. Such trust distribution acquires particular relevance in *multi-stakeholder scenarios*, where participants have different incentives. In particular, blockchain's inherent characteristics are suitable for mediating audits, as audits require data integrity, traceability. Furthermore, by mediating access to audit logs, the blockchain provides guarantees to stakeholders, who can verify if the involved parties are using the system for illegal purposes or to gain an unfair advantage. To successfully mediate audits, the blockchain needs an access control mechanism.

Access control constrains the use of a set of resources based on a specific context, in which a subject requested such permission [Pourmajidi et al., 2018]. Access control starts with authentication, where the user's identity is validated by providing *proof of identity*. The most common proofs of identity are shared-keys, one-time passwords, biometric credentials, and digital certificates. When a subject requests

access to a resource, for instance, held in an information system, the authorization process happens, where the system allows or denies access to the resource. Finally, after authorisation, accountability takes place, where the system records the authorisation request performed by the user and the decision that came from that request [Wilikens et al., 2002]. The accountability phase typically records digital traces that users leave when utilising information systems. Systems record those traces in *log files*, or *audit logs* if they are to be used in audits. Auditors can later analyse such log files, for example, to assure compliance from the involved parties.

To ensure the cybersecurity of *audits* in the described scenario, audit data can be stored in a decentralized information system, with an access control model that records access control requests. Auditors must also be accountable concerning the audit logs they access and operate (view, transfer, operations on logs) [Bible et al., 2017], [Abreu et al., 2018]. This way, the risk of fraud or corruption may be lowered, as the blockchain provides the necessary infrastructure to secure audit logs integrity and access. In particular, the blockchain's intrinsic characteristics, such as traceability, reliability, immutability, and decentralization [Xu et al., 2016], [Nakamoto, 2008], [Wood, 2014], are desirable characteristics to secure and access sensitive information. In terms of the three CIA security properties, decentralization ensures *integrity* and *availability* by replicating the data, whereas *confidentiality* is orthogonal (can be enforced using encryption).

In the case of public administration, the governments can be held accountable, and that is a step forward in fighting corruption. In particular, blockchain allows to identify a participant, through asymmetric cryptography, and to map subject's accesses to resources, in a non-repudiable way. This mapping corresponds to the accountability needed for auditors. Auditors can therefore manually verify transactions (when an alert is emitted, for instance) on the blockchain, excluding the need of a centralized authority to give them data [Abreu et al., 2018].

This paper proposes moving towards the resolution of the aforementioned problems of conventional audits, by using the blockchain technology as the infrastructure for more automatic, distributed and decentralized audits. The solution is built on top of a *permissioned consortium blockchain*, Hyperledger Fabric (Fabric) [Androulaki et al., 2018]. It receives log entries from a set of oracles (sources of data external to the blockchain), processes them at the Log Manager component, and sends them to the blockchain. The Audit Log Manager analyses stored logs in real-time, emitting alerts to auditors when it finds an unusual transaction. Such alerts can reduce the time of forensic audits [Bible et al., 2017]. This component can be used by authorised auditors to read logs from the blockchain, leveraging different blockchain access control techniques [Rouhani et al., 2019]. This paper presents a preliminary experimental evaluation using Hyperledger Caliper[1] a project that enables blockchain load testing.

This paper does not design the blockchain for the proposed logging system from scratch, but instead builds on top of JusticeChain[2], a system developed by the authors [Belchior et al., 2019]. JusticeChain stores logs of a critical Portuguese government application, but does not consider the *auditing* aspect. Since auditing requires the analysis of well-secured logs, the authors leveraged JusticeChain. Auditing is, therefore, the problem considered in the present paper. In particular, this research upgrades JusticeChain with auditing functionality, and the formal definition of the four-layered-architecture.

To extend JusticeChain, this research is supported in the Design Science Research Methodology (DSRM), which aims to provide a method to create innovative, purposeful solutions that solve a given problem [Baskerville et al., 2018]. In particular, this research identifies the problem, defines the objectives for a solution, develops the system, evaluates the system, and communicates the results. Such an approach allows extracting design principles from the design choices, namely the four-layer architecture, and the trade-offs discussed in Section 6.

This paper has thus two-fold contributions: economical and societal. Economic benefits are related with organizations lowering financial investments on audits (both money and time). Societal benefits from the trust distribution that the blockchain offers, opening doors to new synergies between organizations and its stakeholders.

[1] https://hyperledger.org/projects/caliper

[2] https://github.com/RafaelAPB/JusticeChain

The remainder of the paper is structured as follows: the threat model section presents threats to data integrity and the approach to address them. Next, we present JusticeChain along with this paper major contribution: towards automatic, secure, and decentralized audits. Evaluation methodology describes the evaluation developed, and the results from the theoretical and experimental evaluation. After that, related work concerning access control and audit logs is presented. Finally, the conclusions and future work of the paper are described.

2 Data Integrity Threats

In order to mitigate data integrity threats, it is important to perform an analysis of current challenges and risks [Martins and Guerreiro, 2019]. We provide a general threat model with regard to data integrity in information systems and data integrity threats on the blockchain. This section presents the threats to which logs are exposed. The threat model will focus on three factors: i) internal access attacks; ii) remote attacks; and iii) Hyperledger Fabric blockchain attacks, since it is the basis of our implementation.

We assume that there are two types of adversaries. A *third-party adversary*, Adversary A_t, can access the trusted computing base of a computer and remotely penetrate the system, by exploiting software vulnerabilities or hijacking root user credentials. Stemming from this, A_t might tamper audit data to compromise auditing and forensic procedures or tamper system data to cause damage to the organization. On the other hand, an *inside adversary*, Adversary A_o, can be an employee with root privileges. Such a person can access the databases and can be motivated to tamper data for personal benefit. Such adversaries can tamper data to hide their traces or tamper data to aid third-parties illegally.

There can be two main types of attacks to data integrity: physical access attacks and remote vulnerability attacks [Ahmad et al., 2019]. In a *physical access attack*, Adversary A_t or Adversary A_o have access to the critical system components. The attacker generates transactions to change the current values of the objects held in a database. As objects are being updated, the database generates an audit log, tracking the changes made by the attacker. The attacker then focuses on deleting the evidence by deleting the generated audit logs or changing its content. The attacker can manipulate the auditing process by modifying the history maintained by the audit log. As a consequence, the obfuscation of illegal activities and impersonating someone's actions can occur.

In *remote vulnerability attacks*, the attacker exploits default vulnerabilities on systems, such as software malfunctions and security vulnerabilities (e.g., SQL injection). Although such attacks are common [Biswas et al., 2018],, corporate organizations have their systems and databases secure against conventional attacks. The research is focused on the physical access attacks. Five threats compose the threat model:

Threat 1 (T1) – Log tampering from an external element: An attacker violates the integrity of the applicational logs, by editing them. T1 is an external adversary gaining access to the logs. Having access to the logs, attackers can edit the logs at their will, i.e., deleting arbitrary entries. This attack has a higher severity on information systems which do not replicate logs as the attacker can permanently delete information.

Threat 2 (T2) – Database tampering from an internal adversary: An attacker from one of the stakeholders violates the integrity of the applicational logs, by editing them directly. T2 is similar to T1, with a higher degree of severity. If adversaries are insiders, they have direct access to the protected resource. An internal adversary might know peculiar ways to obfuscate such activities.

Threat 3 (T3) – Log tampering by the system administrator: The administrator of the system, with the highest permissions, violates the integrity of the applicational logs, by editing them. There is the possibility of obfuscating activity traces by deleting evidence on other systems (e.g., RADIUS logs). T3 is similar to T2, with higher severity. Administrators have access to all resources and can, theoretically, delete all traces. The usage of a blockchain can prevent the threats as mentioned earlier. Nonetheless, such a method has its threats to data integrity.

Threat 4 (T4) – A participant edits logs that are protected by the blockchain. T4 is not severe, if using a permissioned, consortium blockchain, like Fabric, because transactions have to be endorsed before committed. Even if any participant on the network tries to modify the applicational logs maliciously on its

ledger, they cannot change other peers' ledger state, as honest endorsers would not endorse such transactions.

Threat 5 (T5) – The majority of participants conspire and modify the logs. T5 evolves from T4, where the minority of nodes try to tamper logs. Members from the network can collude to alter the logs' integrity, to trick an external auditor. If all participants on a network want to change its state, it is theoretically possible. The participants can follow the protocol and rewrite the world state, submitting new transactions from the point on in which they want to change the state. Another way to mitigate this threat is to periodically sent the last hash block of the permissioned blockchain, as a transaction, to a public blockchain [Putz et al., 2019], at the expense of paying fees.

3 JusticeChain and Audit Layer

This section introduces *JusticeChain* [Belchior et al., 2019], and extends it with an automatic audit analysis layer.

JusticeChain aims to improve the log resilience by recording applicational logs from a critical information system from the Portuguese Justice System, whereby different stakeholders coexist, and securing them on the blockchain; the latter decentralises the storage of such logs, resulting in higher availability and robustness. Therefore, it allows authorised auditors to analyse the usage of the system with integrity guarantees.

While JusticeChain addresses the business concerns about storing and managing logs, we focus on their decentralized analysis, since the auditing process is decentralized and transparent for all participants on the network. Our solution is built on top of JusticeChain and therefore shares the same system and data models. JusticeChain securely stores audit logs on the blockchain component, focusing on data integrity and distribution. JusticeChain's blockchain client creates transactions based on logs coming from information systems, via authorized loggers. The same component allows audit data retrieval, to authorized auditors.

In JusticeChain, there are three different types of participants who take part in the ecosystem:

- *Logger:* receives logs from an oracle, and signs them. Logs are sent to the blockchain via a transaction created at the Log Manager. The Logger provides traceability and non-repudiability of the logs.
- *Auditor:* Auditors manually inspect audit logs or alerts emitted by the evaluation of such logs. Auditors can only read audit logs or alerts when allowed by other auditors. They have a set of permissions, allowing for fine-grain permissions for auditing purposes.
- *Administrator:* manages the blockchain configurations, and the set of participants. A network administrator can represent an information system, and the interests of their stakeholders, or be independent, acting in behalf of the interests of the consortium.

The asset to be protected is the audit log. An audit log corresponds to a log entry generated by an information system. It has a unique identifier, making it possible to unambiguously identify each log entry. A log entry has a blockchain timestamp, corresponding to the timestamp of the transaction on the blockchain, and an entry creation timestamp. The entry creation timestamp is created when the entry log is generated in an external system from the blockchain.

The blockchain client comprises two fundamental entities: the Audit Manager and the Log Manager. These two components collaborate to serve as the blockchain client, allowing two types of participants to access the blockchain for different purposes. The JusticeChain client exposes application programming interfaces (APIs), which are used by the Audit Frontend and JusticeChain Oracle to access the blockchain.

The JusticeChain blockchain client is composed by:

- *JusticeChain Client:* is a collaboration between two components - Log Manager and Audit Manager. The JusticeChain Client communicates with the blockchain network via a set of REST APIs provided by Hyperledger Composer.
- *JusticeChain Oracle:* retrieves applicational logs from a trusted outward log repository. The oracle can create a buffer of logs and send them to the Log Manager from time to time or send them in

real-time. The optimal scenario happens when logs are sent in real-time, as it diminishes the load from the blockchain and information can be exposed to auditors in real-time.

- *JusticeChain Log Manager:* connected to one or more oracles, creates and submits a transaction to the blockchain, on a Logger's behalf.
- *JusticeChain Audit Manager:* sends transactions to the blockchain on behalf of the corresponding Auditor or Admin who needs to audit logs. Those transactions are queries on the blockchain ledger, allowing the JusticeChain client to retrieve the desired applicational logs.
- *Audit Frontend:* is a user interface that allows the stakeholders (Auditors or Network Admins) to retrieve the applicational logs, via the Audit Log Manager. Allows obtaining the logs in a certain time frame, allowing auditability. It also shows warnings emitted by the smart auditing contracts.
- *Log Repository:* corresponds to the repository that stores logs (i.e. database).
- *Hyperledger Composer* (REST) *Server:* is generated from a business network archive, and exposes an API that the JusticeChain client can use to access the blockchain. Hyperledger Composer Server accesses Hyperledger Fabric using its API.

The blockchain client components' architecture is represented in Figure 1, using the Archimate enterprise architecture modelling language [TO Group, 2016].

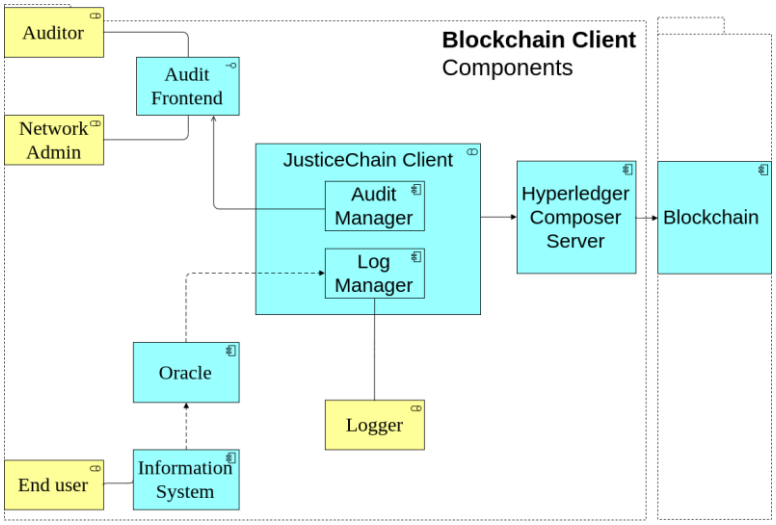

Figure 1: JusticeChain's blockchain client components

4 Auditing Approach

Using JusticeChain's context, one can assume that the system can record every possible action a subject can do over a resource, in any context and thus there is no information loss. The solution also assumes that consensus between participants is always obtained, and malicious nodes on the network are not the majority and thus cannot stall consensus.

Additional steps are now considered: log pre-processing and automatic analysis. Log pre-processing includes log compaction and minimization. Compaction and minimization include transforming log entries. Compaction transforms the content of attributes into a more lightweight version (e.g., for the attribute name, instead of representing the full name, the first and last names are provided). Minimization takes a similar set of attributes and removes redundant attributes. For instance, if there are two attributes, identifier of a court and name of such court, the name can be removed, as it can be inferred off-chain. Log pre-processing is case-specific. Automatic log analysis is leveraged by code executed in the Blockchain nodes (called chaincode in Fabric).

In particular, the solution introduces the audit contract (AC) that observes and reports denied access control requests, and suspicious accesses, encoded dynamically by specific rules. By leveraging the AC, one can implement auditing business logic rules that apply to every transaction. For instance, any suspicious action from a user can trigger an alert, which is shown to the competent authorities. The

blockchain records all important events, on-chain, thus integrity is guaranteed, allowing for quicker decisions from the authorities.

Although blockchains are unlikely to replace auditor judgment, they can reduce preparation times and leverage semi-automatic analysis help to reduce the time-window between the beginning of an audit and its end. In particular, the real time transaction gathering allied to automatically alert auditors when an unusual transaction happens closes the aforementioned gap.

The issuing of transactions can lead to chaincode execution which can produce Events. Events are notifications emitted when a specific condition is met, which applications may listen and react to.

The relevant events JusticeChain emits are:

- New log: this event is emitted when a audit log entry is created.
- New Alert: this event is emitted when the evaluation of the audit smart contract finds a nonconformity. Alerts contain the audit log id that generated them, the id of the Logger that generated it, a message explaining the reason and a timestamp.
- Audit Started: this event is emitted when an Auditor has permission to access audit logs. For an Auditor to be able to initiate an Audit, he or she must collect at least votes from Auditors related with his or her information system.
- Audit Ended: this event is emitted when an Auditor market the audit as completed.
- Permission Revoked: event emitted when an Auditor denies audit permissions to another Auditor.
- Permission Given: event emitted when an Auditor grants audit permission to another Auditor.
- The proposed framework contains four layers that encompass both the blockchain and blockchain client components.

The *Data Acquisition* layer retrieves data from information systems, and sends it to the Data Transfer layer, more specifically to the Log Manager component from the JusticeChain Client.

The *Data Transfer* layer logically connects information systems and end-users to the data stored in the blockchain, and contains the Audit Manager and the Log Manager. The Log Manager receives audit logs from external oracles are pre-processed (standardized, and truncated), and are signed with a Logger's private key. More specifically, in Fabric, a participant on the network represents an entity and has a digital identity, stored in a wallet. In our case, the Logger is a network participant that contains a digital identity, which includes an X.509 certificate, emitted by a tool called configtx (in a production environment, credentials for the Logger should be emitted by a certificate authority). The Logger, therefore, controls a public-private key pair. Hence, the Logger signs each log with its private key. Such information is timestamped and turned into a transaction. The client then issues the transaction to Hyperledger Composer API, that, in its turn, converts it to a Fabric transaction. The audit manager issues queries to access stored data, on behalf of auditors.

The *Data Audit* layer provides a user interface for network admins to manage the network and for auditors to access audit data. Audit data comprises the processed logs and the alerts emitted by a decentralized analysis performed by the blockchain, following specific audit rules. Audit rules are sets of rules made by information systems specialists, business analysts, and other stakeholders, following the design research methodology. Such audit rules encompass the specific conditions of a particular environment. In particular, audit rules define when a log is considered suspicious or needs manual parsing. We set arbitrary rules (e.g., if a user's email that has a specific email domain contained accesses processes categorized as "very sensitive," emit an alert). We do not disclose such rules for privacy motives.
Accesses to this audit data are mediated from the blockchain, which contains and enforces access control rules (for instance, an auditor from information system A can only see logs from information system A, when authorized by a quorum of other auditors from information system A).

Finally, the *Data Analysis & Storage* layer analyses and stores audit data. Processed audit logs are analysed in a decentralized way by all endorsing nodes belonging to the network, yielding events if incidents are found. Such events constitute alerts, and are stored permanently on the blockchain. After analysis, the correspondent audit log is stored on the blockchain. Figure 2 denotes the client server architecture of the solution (the arrows indicate data transfer).

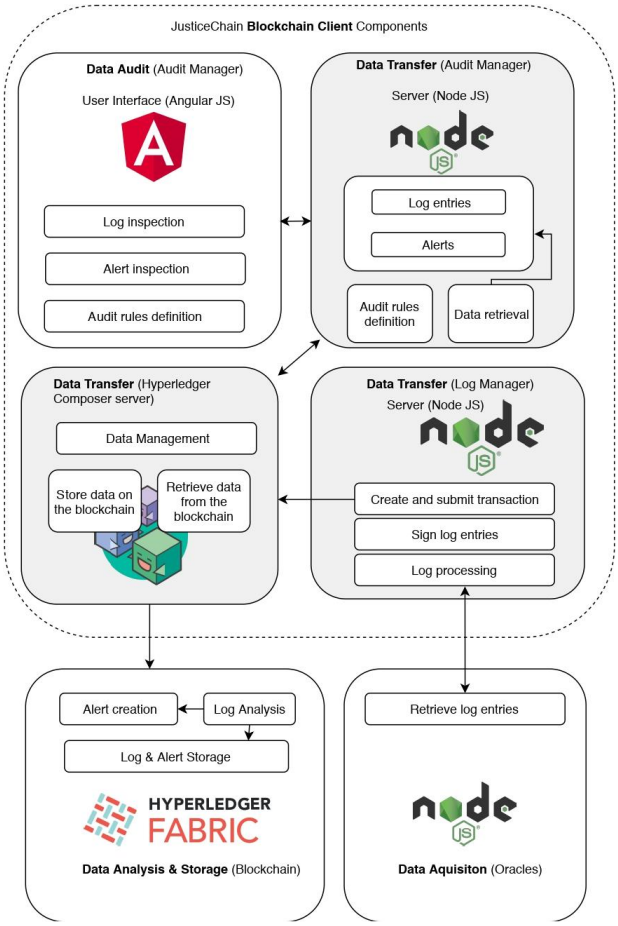

Figure 2: JusticeChain's Client-server architecture plus the auditing solution

A design decision the authors had to make was whether to store the logs on-chain or off-chain. If data is held off-chain, it can be linked to off-chain data to on-chain data through its proof of integrity (i.e., hash). The information recorded on the blockchain is, thus, the hash of the log entry. While this option requires less storage, it relies on a trusted off-chain storage, that also needs to yield data integrity assurances. Thus, the problem of storing and distributing data through the blockchain is transferred to several problems.

First, one needs to assure a safe communication channel between the off-chain and on-chain components. Secondly, one needs to assure not only the security and dependability of the blockchain, but also for the trusted off-chain storage. If such off-chain storage is successfully attacked, resulting in data lost, it invalidates the blockchain solution. A common solution for this would be increasing the resiliency of the off-chain storage, e.g., with replication. Consequently, one may just replicate the information on the blockchain.

Although saving data on the blockchain results in high storage costs [Xu et al., 2016], it yields important advantages. First, as the information is replicated, it becomes more robust and resilient to attacks, leading to a higher availability and dependability of systems that rely on that information. When audits are done on-chain, the blockchain can parse and analyse audits on the go, alerting stakeholders if there is a behaviour out of the ordinary. If such analysis is performed by off-chain components, there are no guarantees that its results are communicated to all stakeholders. Moreover, if logs are in an off-chain storage, the blockchain needs to retrieve them, resulting in higher latencies (and possible less availability guarantees).

Considering the use case, the CIS, in which trust distribution and dependability is crucial, the option was on storing logs on chain, aware of the high storage costs. A solution that can dramatically reduce storage is to map every possible state of CIS users, using an ontological layer. Such mapping allows an

information system to record deviations of standard flows. Those deviations could be stored in the blockchain for further analysis. Although a suitable solution that deserves some attention as future work, the paper opts for a simpler approach, saving all logs and analysing them.

Nonetheless, it is important to compact and minimise the information on logs, containing only the essential. Attributes that can be derived from others (such as the court name from the court id) may not be incorporated on such logs. The Log manager does this processing, adapting to dynamic needs of stakeholders. Another solution is to group several log entries in the same transaction (containing their unique timestamps).

A distributed, automatic audit system benefits if audit logs are decentralized and stored on-chain, allowing for easy retrieval and inspection. The system proposed does not depend on specialized hardware, or in trusted third parties.

5 Audit Data Access Control

Inspired by previous work, this research implements a simple data access control component that mediates data flow from the Data Audit layer to the Data Transfer layer, and vice-versa.

Access control regarding audits is enforced in a peer-to-peer fashion, in which Auditors allow other Auditors to inspect audit logs from the information system they are both related. Such permission is done via a Allow Audit transaction. By stating the permission grant on the blockchain, audits are only possible when auditors reach agreement, decreasing the risk of auditors accessing data unduly. Such transaction emits an event, Permission Given.

An auditor associated with an information system A belongs to the set $\text{Auditor}_{\text{IS-A}}$. When an element from $\text{Auditor}_{\text{IS-A}}$ gathers p permissions, higher than a threshold θ, one can issue a Start Audit transaction, obtaining access to audit logs. Thresholds are use-case specific. Some thresholds that one can consider are: (i) the majority, $\theta = \lceil Auditor_{IS-A}/2 \rceil$, (ii) all, θ $\text{Auditor}_{\text{IS-A}}$ and (iii) at least one, $\theta = 1$.

When an Auditor starts inspecting logs, their permissions are reset. After the audit, the auditor issues the End Audit transaction, marking the audit as complete. A Audit Ended event is emitetd.

This mechanism allows the correct usage of audit data that belongs to stakeholders with different incentives from the administrators of information systems.

6 Evaluation

Considering that this solution is built on top of JusticeChain, its performance is highly related to the performance of that system. In particular, the performance obtained in JusticeChain reduces, since the automatic analysis functionality is added. This section describes the environment used to evaluate the system, the used metrics and the evaluation methodology. Then, an approach to perform the system evaluation is presented, accompanied by the definition of metrics and goal targets. Later, the information provided by the evaluation is collected, analysed and discussed. Finally, the authors discuss about the system's limitations and possible approaches to address them.

A replication of the real production environment was set up, with several distributed clients, emulated by Caliper. A blockchain client corresponds to a Logger. A machine was deployed in Google Compute Engine (GCE). At GCE, an instance was set up at europe-west2-c, with 8vCPU and 7.2GB of RAM.

As a state database, instances of Hyperledger Fabric LevelDB images were used. A simplified Fabric v1.2.0 network with a solo channel comprising two organizations (Org1, Org2) with one peer each and one CA each was considered. The consensus scheme is solo ordered (broadcasts the transaction without establishing any real consensus, used for testing purposes). In this evaluation, the used block size was 128MB, the default batch timeout 250ms and the number of transactions per block is 10. We considered 5 blockchain clients.

7 Performance

To evaluate performance, the following three questions are tackled: (i) What is the maximum transaction throughput JusticeChain can achieve, i.e. how many audit logs per second can it record per time unit;

(ii) what is the transaction latency at the maximum throughput, i.e., what is the time window needed for audit logs to be secured; and (iii) what is the cost, in terms of storage, of protecting applicational logs, i.e., what is the scalability of JusticeChain?

To answer these questions, the performance metrics transaction throughput, transaction latency, and resource consumption, as advised by the Hyperledger community[3] are assessed. The transaction latency corresponds to the difference between the transaction confirmation time and the submit time. It is the time taken for the transaction to affect the network. The transaction throughput is the ratio between the total committed transactions and the total time in seconds. The evaluation does not consider transactions such as Start Audit, Give Permission, or Change Logger Level, as its execution time is very low and, thus, negligible. Instead, the focus is on the transaction Create Log, as it encompasses a write on the blockchain and the automatic analysis. The Hyperledger Caliper is used, a project baked by the Hyperledger Foundation, which aims to facilitate the evaluation of blockchain solutions.

7.1 Throughput and Latency

1,000 transactions are issued at a variable rate, in order to study the throughput and latency evolution. Three different audit rules that analyse each log entry are considered. π_1 is a simple comparison, involving 1 attribute. π_2, in its turn, makes 20 comparisons, on 20 attributes. π_3 makes 100 comparisons on 20 attributes. Figure 3 shows the throughput as a function of the send rate. The x axis shows the send rate, while the y axis shows the throughput in transactions per second (tps).

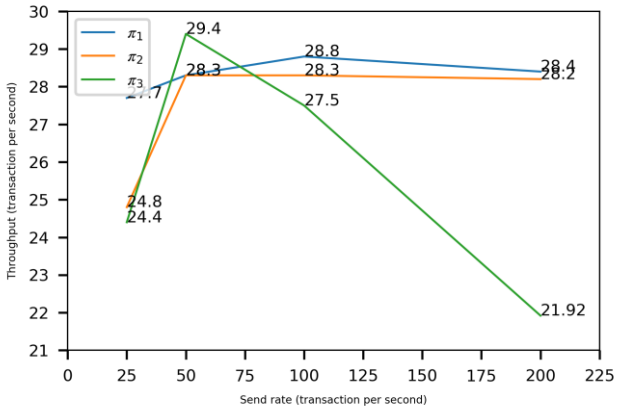

Figure 3: Average throughput of the CreateLog transaction, including the audit contract execution over such log.

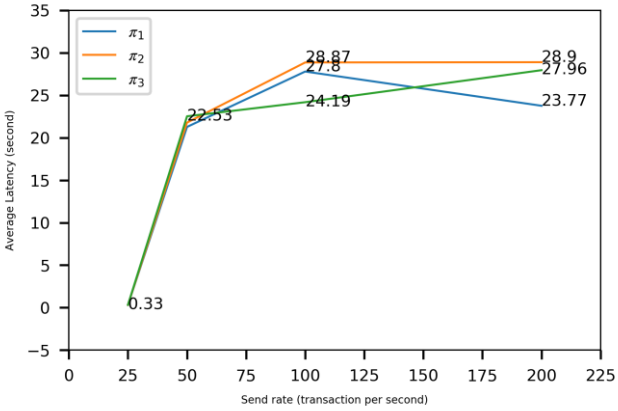

Figure 4: Average latency of the CreateLog transaction, including the audit contract execution over such log.

The maximum throughput is 28.8 tps, when the send rate is 100 tps, with the simplest policy, π_1. The minimum throughput occurs at 200 tps, with the most complex policy, π_3. Figure 4 shows the average latency of creating, storing and analysing an audit log as a function of the send rate. The x axis shows the send rate, while the y axis shows the average latency (seconds). A higher latency generally corresponds to a higher transaction rate. The minimum latency is obtained at a 25 transaction per second rate, being similar in all types of audit rules. Figure 3 shows the throughput as a function of the send rate.

7.2 Storage

Storage evaluation aims to predict how much storage the CIS needs for a long-term solution. An experimental evaluation is presented, taking into account the CIS.

We issued transactions that create audit logs. Attributes from such logs are compacted and minimized. Different types of Logs (namely with 10, 20 and 30 attributes) were tested to infer the impact of different

[3] https://www.hyperledger.org/resources/publications/blockchain-performance-metrics

audit log sizes. Log_{10} corresponds to a compact version of Log_{20}. Log_{30} corresponds to a non-minimized of Log_{20}.

Using Caliper, the storage needed for 100, 500, 2,500, and 5,000 transactions are measured, for each type of log. The results are in Figure 5.

Considering the linear regression in Figure 6, and assuming that 10,000 users from the CIS perform 100 operations that generate an audit log entry per day, one can predict the storage requirements. The total storage required for a year, for each peer node holding Log_{10}, Log_{20}, and Log_{30} is approximately 5.7TB, 6.3TB, and 7TB, respectively.

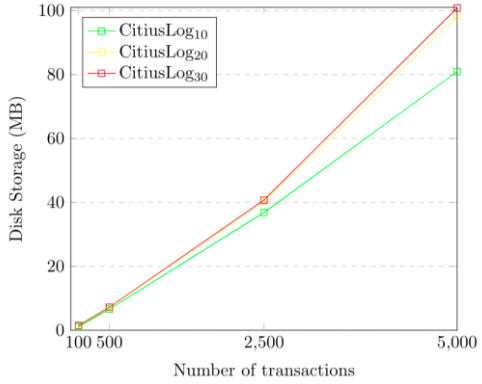

Figure 5: Disk Storage required in function of the type of Log.

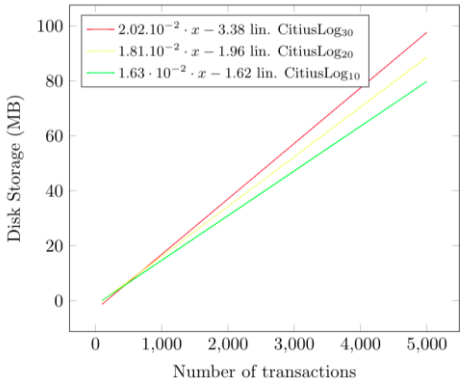

Figure 6: Linear regressions of Log_{10}, Log_{20} and Log_{30}

7.3 Discussion

First and foremost, a private blockchain was used to implement our solution. In particular, we found it to be the best option for several reasons. Firstly, applicational logs from a crutial governmental information system are very sensitive – they need strong privacy assurances and robust access control. Hence, a public blockchain is discarded. A second option could be taken: data could be stored off-chain, and the hashes of such data on a public blockchain. Nonetheless, data needs availability and redundancy, and to be resistant to tampering, which can be challenging in a centralized database. A suitable solution for increasing the robustness in data at centralized databases is distribution and redundancy. We obtain such properties with a permissioned blockchain. Hyperledger Fabric offers the best features to our use case, including the private data feature, the endorsement policy feature, and scalability up to around 2,000 transactions per second. Our choice follows, given the requirements.

Secondly, there is a throughput-latency relation. The higher the throughput, the higher the latency, as more transactions take longer to be committed to the ledger. A maximum throughput of around 25 tps is achieved, with a latency around 0.3 seconds. As the transaction rate increases to 50 tps, the system throughput increases as well, reaching its peak performance. After that point, performance is degraded, yielding lower throughput rates and higher latencies.

The throughput limitations are a consequence of the complexity of the transactions, and because of JusticeChain's several layers. The transaction is first generated at a blockchain client, sent to the Log Manager, and from there a blockchain transaction is created and sent to Composer. Composer then transforms the transaction into a Fabric transaction. Composer might be a considerable bottleneck, which we plan to remove in future work.

Regarding storage, one can verify that the difference between Log_{10}, Log_{20}, and Log_{30} it is not very significant, given the low storage price from several cloud providers. This also indicates that the solution can scale in terms of the number of attributes a log can hold.

The storage price compared to storing only hashes of log entries is much higher. Nonetheless, there is a trade-off between storage and dependability. Ultimately, the replication of audit logs on the blockchain can increase the availability and resiliency of audit logs.

A limitation from this evaluation is that the linear regression was made based on few (and fairly small) values. An improvement would be measuring the response of the system in terms of storage for as much transactions as generated in a year. For financial and time constraints, such evaluation was not performed.

Concerning security, the authors argue that this solution is suitable to minimize security threats interception, interruption, modification, and fabrication [Tanenbaum and van Steen, 2006].

The usage of safe channels (i.e., TLS encryption) minimises interception. The blockchain replicates audit logs and their analysis, contributing for a dependable, available repository of information, minimising interruption attacks.

Modification is related to fabrication, because if a subset of nodes colludes and controls the network, it can modify audit logs or create invalid audit logs. Given Fabric's intrinsic properties and the setting of our blockchain solution, audit logs cannot be updated or deleted, assuming that there is no collusion. In case of collusion (e.g., changing the transaction history, or to create an audit log illicitly), one could argue that consensus in Fabric is not byzantine fault-tolerant, or as secure as Bitcoin's proof of work. In Fabric, the execute-order-validate paradigm, allied with the transaction flow and endorsement policies, can provide the necessary security. Endorsement policies define from which peers a transaction has to be endorsed (validated) in order for consensus to be reached. Fabric achieves consensus through the endorsement policy and resolves transaction ordering with algorithms such as RAFT. If we define the endorsement policy as an "AND" between all participants (all participants have to agree on a transaction result to it to be considered valid), then we can find the blockchain safe as long as at least one participant is honest. In particular, if a third-party auditing entity participates in the system, we can assume it is secure.

Our system complies with suggestions [Akoka and Wattiau, 2010] for auditing information systems, in particular the security metrics (consistency, integrity, reliability), readability (auditability, evolutivity), and quality.

8 Related Work

Some authors are exploring the blockchain technology to create tamper-proof audit logs. Sutton and Samavi propose a mechanism for log integrity and authenticity verification that auditors can utilise [Sutton and Samavi, 2017], using Bitcoin. The proposed system logs proof integrity proofs on the blockchain. There are three main entities on the system, the logger, audit log and auditor. The logger creates the logs and stores them on the audit log. Auditors query the audit log, which contains the logs created by the logger.

As events need to be non-repudiable, the logger signs the transactions, and submits them to prove accountability. Furthermore, the logs need to have integrity assurances, so integrity proof digests of the log events (i.e., cryptographic hash) are generated and stored on the integrity preserver (i.e., blockchain). Those records can be retrieved to participate in the process of compliance checking, with log integrity verification. The solution has low throughput, as the Bitcoin blockchain can only hold 3-7 tps.

Cucurull et al. present a similar approach to Ma et al. since it combines MACs and DAs (data auditors). The Bitcoin blockchain is used, as it provides distributed immutability [Cucurull and Puiggali, 2016]. By using the blockchain, the logs are chained in the same order as they were generated.

Anderson and Smith propose AuditChain, a blockchain-based full-stack system to secure, standardise and simplify health record audit logs [Anderson and Smith, 2018]. In this work, identities are issued through Hyperledger Fabric certificate authority and linked to a local user profile

Logchain is a blockchain-assisted log storage system [Pourmajidi and Miranskyy, 2018]. Logchain tries to decentralise trust on stakeholders that use a third-party service. In particular, Logchain goes towards

granting the user protection against the tamper-motivation cloud providers might have (in case of dispute, e.g. accruing from improper service provisioning).

Some proposals take advantage of the off-the-shelf immutabilization capacity of the blockchain technology [Cucurull and Puiggali, 2016], [Sutton and Samavi, 2017], [Zyskind et al., 2015] to protect data (logs, personal data), albeit those entail a monetary cost due to the nature of public blockchains such as Bitcoin or Ethereum.

Privacy and performance problems with LogChain go *pari passu*, as the system entails the usage of a permissionless public blockchain. Blockaudit [Ahmad et al., 2019] takes a holistic approach to the problem of protecting audit logs from adversaries, by protecting both physical access attack and the remote vulnerability attack, using a custom blockchain infrastructure.

On a recent endeavour, Putz, Menges & Pernul present a solution that uses blockchain to store non-repudiable proofs for existence of generated logs, which does not rely on a third party or specialized hardware [Putz et al., 2019]. The authors argue that the blockchain technology is suitable for audit log storage. Furthermore, they present a four-step procedure which adapts to auditing needs: evidence generation, evidence transfer, evidence storage, and dispute resolution. The solution architecture comprises a logging infrastructure, a custom blockchain system and an auditor log verification.

9 Conclusion

This paper presents a blockchain-based system which increases trust in the audits - a context with several non-trusting stakeholders. The solution proposed is inspired by the design science research methodology, and it is based on a 4-layer architecture: the data acquisition, data transfer, data audit, and data analysis & storage layers. The JusticeChain Log Manager, Audit Manager, an oracle and a Hyperledger Fabric blockchain implement the four layers. The solution works without a trusted third party, as independent nodes agree on the shared ledger state. As a result, our solution permits data storage with integrity guarantees, mediated access to audit logs and decentralized, semi-automatic audits on the blockchain. The evaluation shows robustness to both physical and remote attacks concerning the security of the solution. The performance evaluation shows that although improvements can be made, the throughput and latency are enough to operate a medium sized information system at the CIS sector, yielding around 30 transactions per second with a 0.3 seconds latency, which can be greatly improved, by switching the underlying technologies from Composer and Fabric to only Fabric.

This paper provides contributions towards distributed, secure and automatic audits. A challenge to automate secure audits is to dynamically change audit rules that are used to evaluate audit logs, in an efficient way. Research is being developed regarding the suitability of a trusted oracle that provides such rules. Finally, the dependability increase of an audit system powered by blockchain, by exploring how different blockchains can interoperate, is a future research path.

Acknowledgments

We thank the anonymous reviewers for their constructive feedback and suggestions that helped us improve the paper. This work was partially supported by the EC through project 822404 (QualiChain), and by national funds through Fundação para a Ciência e a Tecnologia (FCT) with reference UIDB/50021/2020 (INESC-ID).

References

Abreu et al., 2018. Abreu, P. W., Aparicio, M., and Costa, C. J. (2018). Blockchain technology in the auditing environment. In 2018 13th Iberian Conference on Information Systems and Technologies (CISTI), pages 1–6.

Ahmad et al., 2019. Ahmad, A., Saad, M., and Mohaisen, A. (2019). Secure and transparent audit logs with BlockAudit. Journal of Network and Computer Applications, 145(1).

Akoka and Wattiau, 2010. Akoka, J. and Wattiau, I. (2010). A Framework for Auditing Web-Based Information Systems. In European Conference on Information Systems.

Anderson and Smith, 2018. Anderson, J. and Smith, S. (2018). Securing, standardizing, and simplifying electronic health record audit logs through permissions blockchain technology. PhD thesis, Dartmouth College.

Androulaki et al., 2018. Androulaki, E., Barger, A., Bortnikov, V., Cachin, C., Christidis, K., Caro, A. D., Enyeart, D., Ferris, C., Laventman, G., Manevich, Y., Muralidharan, S., Murthy, C., Nguyen, B., Sethi, M., Singh, G., Smith, K., Sorniotti, A., Stathakopoulou, C., Vukolic, M., Cocco, S. W., and Yellick, J. (2018). Hyperledger Fabric: A Distributed Operating System for Permissioned Blockchains. EuroSys '18: Proceedings of the Thirteenth EuroSys Conference.

Baskerville et al., 2018. Baskerville, R., Baiyere, A., Gregor, S., Hevner, A., and Rossi, M. (2018). Design science research contributions: Finding a balance between artifact and theory. Journal of the Association for Information Systems, 19(5):358–376

Belchior et al., 2019. Belchior, R., Correia, M., and Vasconcelos, A. (2019). JusticeChain: Using Blockchain To Protect Justice Logs. In CoopIS 2019: 27th International Conference on COOPERATIVE INFORMATION SYSTEMS.

Bible et al., 2017. Bible, W., Raphael, J., Riviello, M., Taylor, P., and Valiente, I. (2017). Blockchain Technology and Its Potential Impact on the Audit and Assurance Profession. Technical report, Deloitte & Touche LLP

Biswas et al., 2018. Biswas, S., Sohel, M., Sajal, M., Afrin, T., Bhuiyan, T., and Hassan, M. M. (2018). A Study on Remote Code Execution Vulnerability in Web Applications. In International Conference on Cyber Security and Computer Science.

Buterin, 2014. Buterin, V. (2014). Ethereum: A nextgeneration smart contract and decentralized application platform.

Casino et al., 2019. Casino, F., Dasaklis, T. K., and Patsakis, C. (2019). A systematic literature review of blockchain-based applications: Current status, classification and open issues. Telematics and Informatics, 36:55– 81.

Chen et al., 2013. Chen, Z., Yang, Y., Zhang, R., and Li, Z. (2013). An efficient scheme for log integrity check in security monitoring system. In IET Conference Publications, volume 2013, pages 246–250.

Cucurull and Puiggal´ı, 2016. Cucurull, J. and Puiggal´ı, J. (2016). Distributed Immutabilization of Secure Logs. In Barthe, G., Markatos, E., and Samarati, P., editors, Security and Trust Management, pages 122–137, Cham. Springer International Publishing.

Di Francesco Maesa et al., 2018. Di Francesco Maesa, D., Mori, P., and Ricci, L. (2018). Blockchain Based Access Control Services. In 2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), pages 1379–1386.

Ferraiolo et al., 2001. Ferraiolo, D., Sandhu, R., Gavrila, S., D.Kuhn, and Chandramouli, R. (2001). A Proposed Standard for Role-Based Access Control. ACM Transactions on Information and System Security, 4(3).

Gaetani et al., 2017. Gaetani, E., Aniello, L., Baldoni, R., Lombardi, F., Margheri, A., and Sassone, V. (2017). Blockchain-Based Database to Ensure Data Integrity in Cloud Computing Environments. In ITASEC.

Martini and Choo, 2012. Martini, B. and Choo, K.-K. R. (2012). An integrated conceptual digital forensic framework for cloud computing. Digital Investigation, 9:71–80.

Martins and Guerreiro, 2019. Martins, H. and Guerreiro, S. (2019). Access Control Challenges in Enterprise Ecosystems: Blockchain-Based Technologies as an Opportunity for Enhanced Access Control Hugo. In Christiansen, B. and Piekarz, A., editors, Global Cyber Security Labor Shortage and International Business Risk, pages 51–76. IGI Global.

Nakamoto, 2008. Nakamoto, S. (2008). Bitcoin: A peer-topeer electronic cash system.

Pourmajidi and Miranskyy, 2018. Pourmajidi, W. and Miranskyy, A. (2018). Logchain: Blockchain-Assisted Log Storage. In IEEE International Conference on Cloud Computing, volume 2018-July, pages 978–982. IEEE Computer Society.

Pourmajidi et al., 2018. Pourmajidi, W., Miranskyy, A., Diffie, W., Hellman, M. E., Sandhu, R. S., Samarati, P., and Cohen, F. (2018). New Directions in Cryptography. Computers & Security, 32(6):644–654.

Putz et al., 2019. Putz, B., Menges, F., and Pernul, G. (2019). A secure and auditable logging infrastructure based on a permissioned blockchain. Computers & Security, 87.

Rouhani et al., 2019. Rouhani, S., Belchior, R., Cruz, R. S., and Deters, R. (2019). Hyperledger Fabric Attribute-Based Access Control System : A Step Towards Distributed Access Control Using Blockchain. Unpublished Article. Future Generation Computer Systems.

Schneier and Kelsey, 1999. Schneier, B. and Kelsey, J. (1999). Secure Audit Logs to Support Computer Forensics. ACM Transactions on Information and System Security, 2(2):159–176.

Sutton and Samavi, 2017. Sutton, A. and Samavi, R. (2017). Blockchain Enabled Privacy Audit Logs. pages 645–660.

Tanenbaum and van Steen, 2006. Tanenbaum, A. and van Steen, M. (2006). Distributed Systems: Principles and Paradigms (2Nd Edition). Prentice-Hall, Inc., Upper Saddle River, NJ, USA.

TO Group, 2016. TO Group (2016). ArchiMateR 3.0 Specification. Van Haren Publishing.

Wilikens et al., 2002. Wilikens, M., Feriti, S., Sanna, A., and Masera, M. (2002). A context-related authorization and access control method based on RBAC:. pages 117–124

Wood, 2014. Wood, G. (2014). ETHEREUM: A SECURE DECENTRALISED GENERALISED TRANSACTION LEDGER.

Xu et al., 2016. Xu, X., Pautasso, C., Zhu, L., Gramoli, V., Ponomarev, A., Tran, A. B., and Chen, S. (2016). The Blockchain as a Software Connector. In 13th Working IEEE/IFIP Conference on Software Architecture (WICSA), pages 182–191.

Zyskind et al., 2015. Zyskind, G., Nathan, O., and Pentland, A. . (2015). Decentralizing Privacy: Using Blockchain to Protect Personal Data. In 2015 IEEE Security and Privacy Workshops, pages 180–184.