JusticeChain: Using Blockchain to Protect Justice Logs

Rafael Belchior[1,2], Miguel Correia[1,2], and André Vasconcelos[1,2]

[1] Instituto Superior Técnico, Portugal, Universidade de Lisboa, Lisboa, Portugal
[2] INESC-ID, Lisboa, Portugal
{rafael.belchior,miguel.p.correia,andre.vasconcelos}@tecnico.ulisboa.pt

Abstract. The auditability of information systems plays an essential role in public administration. Information system accesses are saved in log files so auditors can later inspect them. However, there are often distinct stakeholders with different roles and different levels of trust, namely the IT Department that manages the system and the government ministries that access the logs for auditing. This scenario happens at the Portuguese judicial system, where stakeholders utilize an information system managed by third-parties. This paper proposes using blockchain technology to make the storage of access logs more resilient while supporting such a multi-stakeholder scenario, in which different entities have different access rights to data. This proposal is implemented in the Portuguese Judicial System through JusticeChain. JusticeChain comprises the blockchain components and blockchain client components. The blockchain components grant log integrity and redundancy, while the blockchain client component is responsible for saving logs on behalf of an information system. The client allows end-users to access the blockchain, allowing audits mediated by the blockchain.

Keywords: Blockchain, auditing, audit logs, public administration

1 Introduction

Organizations have the responsibility of protecting their sensitive data, a valuable resource that often guides business decisions. *Access control* mechanisms aim to identify subjects that require access to resources and allow or deny them the access, based on the context of the request [7]. Users that utilize such systems leave digital traces recorded in *log files*. *Auditors* can later analyze such log files, for example, to assess that no parties are using the systems for illegal purposes or to gain an unfair advantage.

At the Portuguese judicial system, there is an information system to manage judicial processes at courts that supports several stakeholders. The entity that maintains that the system faces different incentives from the stakeholders that use it, leading to a multi-stakeholder scenario with uncertain trust among them. In such a scenario, separate entities have different access rights to data. Threats to *log integrity*, like *data tampering*, have to be minimized, as they can invalidate

audits because tampered data cannot be trusted [3]. This paper proposes the use of *blockchain* technology to overcome the problems with the integrity of logs and support auditing, by assuring that no entity or individual can tamper the logs, allowing to build a transparent and collaborative network. Blockchain technology has emerged as a vehicle for decentralization, transparency and trust while conserving security, privacy and control, which can leverage auditability and trust distribution [9], both critical requirements for information systems at public administration.

In particular, we introduce JusticeChain, a system to store, protect and decentralize applicational logs built on top of a permissioned private blockchain, *Hyperledger Fabric* [1]. JusticeChain receives log entries from a set of *oracles*, processes them at the Log Manager component, and acts as a client to the underlying Hyperledger Fabric infrastructure. The Audit Log Manager component can be used by authorized auditors to read logs from the blockchain.

2 Preliminaries on JusticeChain

Security risks such as data tampering, denial of service (DoS / DDoS), man-in-the-middle attacks, identity theft, and spoofing pose severe challenges concerning the security of any information system. We focus on the *data tampering* problem, for it is one of the most frequent security risks, and the one with the most impact on Portuguese public administration audits. We considered different blockchain infrastructures, both public and private. Public blockchains such as Bitcoin and Ethereum are not suitable, as sensitive information cannot be easily stored and retrieved efficiently. Private infrastructures, such as Quorum, Corda, and Multichain seem to be less stable and may lead to lower throughput rates than Fabric. Fabric was found to be the most appropriate blockchain, as it is backed by a large active community and has a significantive maximum throughput [1].

JusticeChain improves *log resilience* in two ways: it records applicational logs from information systems with different stakeholders and secures them on the blockchain; it decentralizes the storage of such logs, resulting in higher redundancy and availability. Therefore, it allows authorized auditors to analyze the usage of the system with integrity guarantees. The auditing process is decentralized and transparent for all participants on the network, due to programs (smart contracts, or *chaincode* in Fabric's lingo) that inspect logs. The threats to which logs are exposed and that are mitigated by JusticeChain are the following:

- **T1**: Log tampering from an external element.
- **T2**: Database tampering from an internal adversary
- **T3**: Log tampering by the system administrator
- **T4**: A participant tries to edit logs that are protected by the blockchain.
- **T5**: The majority of participants conspire and try to modify the logs.

The fact that Fabric allows the creation of a permissioned blockchain, where participants are vetted, allows reducing the risk of collusion. Fabric records updates to the configurations of the system and deployments of chaincode. This

process enables the straightforward identification of the subject that initiated specific actions, being a demotivating factor for adversaries.

2.1 System Model

The use case addressed in this paper presents three characteristics: (i) the participants are willing to cooperate but have limited trust in each other, (ii) the trust and responsibility of managing the logs belong to all stakeholders and (iii) one organization should be able to administer the network, in accordance with the governance model. Regarding the first and second aspects, the use of a consensus algorithm to reach agreement ensures that no single entity controls the blockchain. Concerning the third characteristic, permissioned blockchains as Hyperledger Fabric allow the delegation of a different level of control to specific participants [1].

We assume a worst-case scenario in which participants have limited trust on each other (e.g., have different political incentives). Fabric supports sub-blockchains called channels; we use a single channel that is used by all participants. Participants control peer nodes which maintain the ledger and endorse transactions. JusticeChain allows data management, through Log pre-processing (i.e., standardization, automatic analysis and attribute checks), which is leveraged by distributed chaincode execution (i.e., the execution of programs in blockchain nodes). Although those are useful features, JusticeChain focuses on assuring data integrity and distribution, by storing logs by authorized loggers and retrieving them to authorized auditors. There are three actors (participants) who take part in the ecosystem:

- *Logger*: receives log entries from information systems and uploads them to the blockchain, belonging to a member participant. They act as blockchain clients and can be considered to be oracles (in blockchain lingo).
- *Auditor*: audits secured applicational logs on behalf of an organization. Auditors have a set of permissions, allowing for fine-grain permissions for auditing purposes.
- *Network Administrator*: manages the blockchain configurations. Responsible for creating and managing participants within the blockchain

2.2 Data Model

JusticeChain has a data model that addresses the business concerns about managing applicational logs. Participants defined in Section 2.1 interact with the data in the following ways:

- *Network Administrator (Admin)*: can see the whole ledger, the whole transaction history and update participants. However, they cannot create, update or delete applicational logs.

- *Auditor*: minimizes the risk associated to threat T5 . An auditor member participates in the network, monitoring the flow. If the adversaries try to change their states, the auditor node would perceive such change, as there would be state inconsistencies across nodes. The auditor can be given permissions by the system administrator to enforce synchronization of the state of the ledger if needed. Auditors can only see part of the ledger – logs associated with the auditors organization and the transaction history that affects the network configurations.
- *Logger*: can create logs associated with one information system. For instance, a Logger associated with System A can create an asset type *Log-A*, but cannot create an asset of type *Log-B*.

The ecosystem aims to protect an asset, the Log.

- *Log*: has a unique identifier, *timestamp*, *log creation timestamp*, an associated Logger and case-specific attributes. In the Portuguese justice, there are attributes which represent the universal unique identifier of the user, the audit and also the related court. A timestamp attribute is associated with each entry, as latency issues can place gaps between the log generation and log recording on the blockchain. Several log types can be defined, depending on how many information systems participate in the network. Only Loggers can create Logs, and no entity can update or delete logs.

The process of executing and validating chaincode can produce *Events*, which applications may listen and take actions upon.

Transactions are requests to the blockchain to execute a function on the ledger. Transactions can affect participants and assets. Chaincode written in *NodeJS* creates logs that are recorded on the immutable ledger, via a transaction. We defined a transaction to create a Log type asset, as logs are created through the issuing of such transaction, by an authorized logger.

3 JusticeChain Overview

This section presents an overview of JusticeChain. JusticeChain allows decentralizing trust concerning logging. The assets to be protected are applicational logs generated by an information system related to the judicial system, in our case. The proposed solution is scalable when it comes to supporting different organizations, different types of logs and different auditors. The architecture is represented in Figure 1, using the Archimate modelling language [4].

The blockchain component stores applicational logs and enforces blockchain configurations concerning the different participants on the network. The blockchain client component ensures that participants can access the applicational logs via submitted transactions, and can audit logs, under certain circumstances.

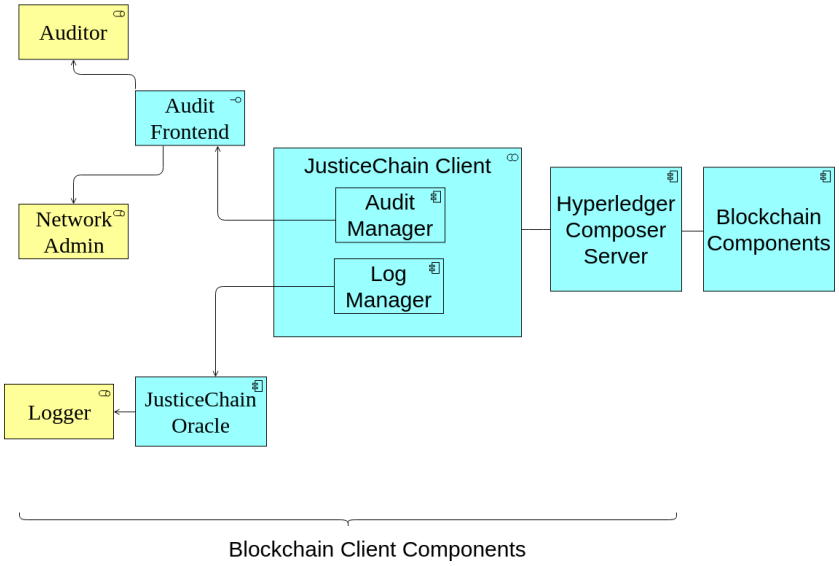

Fig. 1. JusticeChain Architecture

3.1 Blockchain Components

JusticeChain leverages *Hyperledger Composer* (or simply Composer)[3] that, in its turn, uses Hyperledger Fabric to launch and operate a blockchain ecosystem. Composer is an abstraction built on top of Hyperledger Fabric that simplifies the development of blockchain solutions. Through the definition of endorsement policies, one can put more or less trust in a specific set of endorser peers, making the trust system independent from the consensus algorithm to be used [1]. Unlike the public ledger whose truthfulness is guaranteed by the design of consensus processes, it is the endorsement policy that guarantees consensus on the network. For instance, one can define that for a transaction to be valid, peers from organization A and organization B must yield the same result with respect to the execution of specific chaincode. A custom trust system allows an organization to administer the network, while assuring that it cannot take unfair advantages out of it, thus distributing trust.

In this use case, we use two different chaincodes (i.e., programs): chaincode that creates instances of applicational logs (*S1*) and chaincode that accesses the ledger (*S2*), regains logs and retrieves them to the end-user. Chaincode S1 receives the attributes necessary to create a Log from the Log Manager, validate them, apply pre-processing (if needed) and commit the new data to the ledger, by issuing a transaction. Chaincode S2 queries the blockchain for a specific type of Logs.

[3] https://www.hyperledger.org/projects/composer

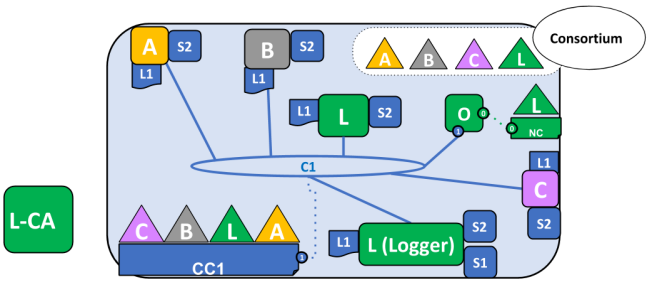

Fig. 2. JusticeChain Blockchain Architecture - Example with a consortium composed of four organizations (A,B,C,L), five peers (Peer A, Peer B, Peer C, Peer L, Peer L (Logger)), one orderer (O), one certificate authority (L-CA), a channel CC1, and network configurations (NC).

A custom certificate authority (CA) is used to issue identities for each participant on the network. Each organization that participates in the network, and thus interested in auditing a specific information system should maintain at least a peer node that holds an instance of the ledger. As applicational logs should not be shared amongst different member organizations, one has to define access control rules that manage that flow (e.g., only Admins and Auditors from organization A can see applicational logs from A, only Loggers from organization A can create type A Logs). Composer allows the definition of such access control rules, by associating an operation (READ, UPDATE, CREATE, DELETE) and an action (ALLOW, DENY) to an Asset (e.g., Log). JusticeChain supports several stakeholders, which do not need to belong to the same organization. Fabric supports different solutions to ensure data privacy between organizations (e.g., Auditor from organization A cannot see logs from organization B):: i) create a different channel; ii) use the private data functionality of Fabric; and iii) tune access control rules via Composer. Private data can be used in this case[4], which allows a subset of organizations the ability to endorse, commit, and query private data.

Client application JusticeChain Audit Manager can use channel C1 to connect to A, B, C, L, L (Logger), and Orderer O. Client application JusticeChain Log Manager can use C1 to access L (Logger) and Orderer O. Applications can only access the ledger L1 through the chaincode instantiated on their respective peers.

3.2 Blockchain Client Components

JusticeChain is a full-stack application that leverages Fabric to secure audit logs while providing support for automatized auditing techniques. The Composer

[4] https://hyperledger-fabric.readthedocs.io/en/release-1.4/private-data/private-data.html

REST Server is used to interact with the underlying Fabric blockchain; hence, it is a blockchain client component.

As presented in Figure 1, the blockchain client comprises mainly two entities: the *Audit Manager* and the *Log Manager*. Both the Audit Manager and the Log Manager expose application programming interfaces (APIs), which allow the *Audit Frontend* and *JusticeChain Oracle* to access JusticeChain functionalities. JusticeChain, on its turn, communicates with the blockchain via the Hyperledger Composer API. The JusticeChain blockchain client has several components:

- JusticeChain Client: is a collaboration between two components - Log Manager and Audit Manager that aims to problem exposed in this paper.
- JusticeChain Oracle (Oracle): overcomes the inability of communicating with the "outside" world. An oracle in the context of our problem is a component that retrieves applicational logs from an outward log repository.
- JusticeChain Log Manager (Log Manager): is connected to one or more oracles. When the Log Manager receives a log, preprocessing is applied, as anonymization or standardization. After that, the Log Manager submits a transaction to the blockchain, on the correspondent Logger's behalf.
- JusticeChain Audit Manager (Audit Manager): sends transactions to the blockchain on behalf of the corresponding Auditor or Admin who needs to audit logs. The transactions are, in fact, queries on the blockchain ledger.
- Audit Frontend: is a user interface that allows the stakeholders (Auditors or Network Admins) to retrieve the applicational logs, via the Audit Log Manager.
- 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 access Fabric using its API.

In addition to acting as a proxy between frontend applications and the blockchain, the JusticeChain Client, allows end-user authentication to the blockchain network. A local database stores local end user's credentials. This way, one can map local authentication credentials and the user's cryptographic identity on the blockchain network, providing traceability.

4 Related Work

In [2], a write-only logger creates log entries as a way to give integrity guarantees. More advanced solutions use a third-party notary service to prevent data-tampering, along with cryptographic hashing, and partial result authentication codes [8]. Such solutions, although efficient, have a single point of failure, where the centralized authority that grants integrity can collude with attackers. Several solutions support forward security but depend at least partially on a third-party [5]. Such solutions, although suitable, does not tackle the need for a trust distribution.

In [6], the authors propose Logchain, a blockchain-assisted log storage system. Logchain tries to decentralize trust on stakeholders that use a third party service. Cloud participants have access to logs but, unlike JusticeChain, fine-grain permissions related to audit are lacking.

5 Conclusion

This paper presented JusticeChain, a blockchain-based system which increases trust in information systems managed by third-parties, regarding logging and audits. In particular, JusticeChain aims to increase the resilience of applicational logs used in the Portuguese justicial system, by assuring integrity and redundancy. JusticeChain improves traditional logging systems by distributing logs, where stakeholders depend on a centralized information system to conduct their activities, which cause trust issues.

Acknowledgements This work was supported by national funds through Fundação para a Ciência e a Tecnologia (FCT) with reference UID/CEC/50021/2019 (INESC-ID) and by the European Commission program H2020 under the grant agreement 822404 (project QualiChain).

References

1. Androulaki, E., Barger, A., Bortnikov, V., Cachin, C., Christidis, K., De Caro, A., Enyeart, D., Ferris, C., Laventman, G., Manevich, Y., et al.: Hyperledger fabric: a distributed operating system for permissioned blockchains. In: Proceedings of the 13th ACM EuroSys Conference (2018)
2. Bellare, M., Yee, B.S.: Forward Integrity For Secure Audit Logs. Tech. rep. (1997)
3. Chen, Z., Yang, Y., Zhang, R., Li, Z.: An efficient scheme for log integrity check in security monitoring system. In: IET Conference Publications. vol. 2013, pp. 246–250 (2013). https://doi.org/10.1049/cp.2013.2026
4. Group, T.: ArchiMate® 3.0 Specification. Van Haren Publishing (2016)
5. Ma, D., Tsudik, G.: A new approach to secure logging. In: Atluri, V. (ed.) Data and Applications Security XXII. pp. 48–63. Springer Berlin Heidelberg, Berlin, Heidelberg (2008)
6. Pourmajidi, W., Miranskyy, A.V.: Logchain: Blockchain-assisted log storage. 2018 IEEE 11th International Conference on Cloud Computing (CLOUD) pp. 978–982 (2018)
7. Sandhu, R.S., Samarati, P.: Access control: principle and practice. IEEE Communications **32**(9), 40–48 (1994)
8. Snodgrass, R.T., Yao, S.S., Collberg, C.: Tamper Detection in Audit Logs. In: Proceedings of the Thirtieth International Conference on Very Large Data Bases - Volume 30. pp. 504–515. VLDB '04, VLDB Endowment (2004)
9. Zheng, Z., Xie, S., Dai, H.N., Chen, X., Wang, H.: An Overview of Blockchain Technology: Architecture, Consensus, and Future Trends (2017). https://doi.org/10.1109/BigDataCongress.2017.85