World Wide Web
https://doi.org/10.1007/s11280-021-00874-7

Distributed attribute-based access control system using permissioned blockchain

Sara Rouhani[1] · **Rafael Belchior**[2] · **Rui S. Cruz**[2] · **Ralph Deters**[1]

Received: 18 May 2020 / Revised: 21 February 2021 / Accepted: 1 March 2021 /
Published online: 23 March 2021
© The Author(s), under exclusive licence to Springer Science+Business Media, LLC, part of Springer Nature 2021

Abstract

Auditing provides essential security control in computer systems by keeping track of all access attempts, including both legitimate and illegal access attempts. This phase can be useful in the context of audits, where eventual misbehaving parties can be held accountable. Blockchain technology can provide the trusted auditability required for access control systems. In this paper, we propose a distributed Attribute-Based Access Control (ABAC) system based on blockchain to provide trusted auditing of access attempts. Besides auditability, our system presents a level of transparency that both access requesters and resource owners can benefit from it. We present a system architecture with an implementation based on Hyperledger Fabric, achieving high efficiency and low computational overhead. The proposed solution is validated through a use case of independent digital libraries. Detailed performance analysis of our implementation is presented, taking into account different consensus mechanisms and databases. The experimental evaluation shows that our presented system can effectively handle a transaction throughput of 270 transactions per second, with an average latency of 0.54 seconds per transaction.

Keywords Distributed access control · Attribute-based access control · Blockchain · Hyperledger fabric · Performance

This article belongs to the Topical Collection: *Special Issue on Emerging Blockchain Applications and Technology*
Guest Editors: Rui Zhang, C. Mohan, and Ermyas Abebe

✉ Sara Rouhani
sara.rouhani@usask.ca

Rafael Belchior
rafael.belchior@tecnico.ulisboa.pt

Rui S. Cruz
rui.s.cruz@tecnico.ulisboa.pt

Ralph Deters
deters@cs.usask.ca

[1] Department of Computer Science, University of Saskatchewan, Saskatoon, SK, S7N5C9, Canada

[2] Department of Computer Science and Engineering, Instituto Superior Técnico, Universidade de Lisboa, Lisboa, Portugal

1 Introduction

Access control systems exist to protect system resources from unauthorized access attempts. Based on the system policies, security procedures within the organization, and the level of the sensitivity of the resources, the access control systems follow one of the available access control models.

Attribute-Based Access Control (ABAC) [24, 60] is an access control model that regulates access permissions based on the characteristics (in this context called attributes) of subjects, resources, and context (or environment). Access decisions are made by evaluating these attributes based on defined policies. An overview of the ABAC model is shown in Figure 1.

ABAC provides fine-grained access control "appointed as a key requirement to minimize the risk of information leakage and confidentiality" [9]. It has some advantages over other access control models as, (a) It can provide fine-grained and flexible access control because it allows an arbitrary number of attributes in access control decisions; (b) The implementation of complex policies is simple and applicable; and (c) It can provide dynamic and effective access control decisions by involving environmental attributes in decision making.

Auditing is one of the essential controls in systems security. Auditing is the action of tracking all access attempts, including both legitimate and illegal access attempts. Keeping track of legitimate access attempts helps with non-repudiation, and keeping track of illegal access attempts helps identifying potential threats. Auditability is also one of the key characteristics of blockchain by providing a trustable history of traceable transactions [8]. Blockchain can exploit smart contracts to store access control policies, process access decisions, store the result of access decisions, and accountability regarding stakeholders with different incentives. Then, all access attempts toward a particular resource can be queried from the blockchain at any point in the future. This feature can be used as an authentic proof for non-repudiation, or it can be studied for further analysis to identify possible threats.

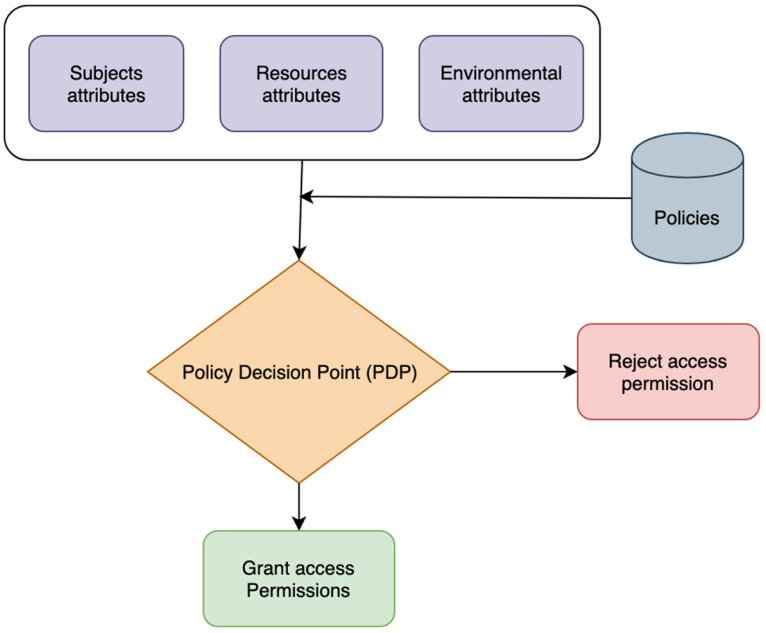

Figure 1 Attribute based access control

Besides, blockchain presents other beneficial features that are desirable for access control systems, such as immutability and transparency. For example, suppose a malicious system administrator changes a policy to grant or deny someone's access. In that case, it will be recorded on the blockchain, and it is not possible to delete the trace of updates on policies from the blockchain. For each policy, all history of changes applied in that policy can be queried by permissioned users in a permissioned blockchain. However, such a problem can be prevented by configuring smart contracts so that authenticated parties must approve any change in access control policies before execution.

This work utilizes blockchain as a tamper-proof auditable record of data, required to regulate the ABAC model's access control, and proposes a complete end-to-end solution for implementing an ABAC system with the focus on policy-based architecture based on Hyperledger Fabric permissioned blockchain[1]. Our contributions are summarized as follows:

- We propose an architecture for implementing a flexible access control system based on ABAC and permissioned blockchain.
- We discuss our access control components including Policy Information Point (PIP), Policy Decision Point (PDP), Policy Administration Point (PAP), which are implemented as smart contracts (chaincode).
- We provide a specific use case of digital libraries to represent the system operation modelling.
- We carried out experiments, and we analyzed the performance of the presented access control application using Hyperledger Caliper[2] based on multiple configurations, including different databases and consensus methods.

The remainder of this paper is organized as follows. The initial concepts of blockchain and access control systems are presented in Section 2, followed by Section 3, which reviews related studies. Section 4 explains the system model, architecture and representation of designed components. A case study based on access to the digital libraries' resources is introduced in Section 5. Section 6 presents the evaluation of results based on the represented case study and multiple configurations. Finally, Section 7 draws conclusions on the developed work and refers to our future developments.

2 Background

This section presents the background on blockchain, smart contracts, and access control.

2.1 Blockchain and smart contracts

Blockchain is a particular type of distributed ledger technology. The data is recorded on the blockchain as a group of transactions called blocks. Each block has a hash value, and it links to the previous block by referencing the hash value of the previous block in the header of the current block. Consequently, data manipulation is not possible in the blockchain, as even a slight change leads to an inconsistency between linked blocks, which can be recognized easily. In order to attach a valid block to the blockchain, a consensus mechanism is applied. There are several consensus mechanisms with a trade-off between performance and security.

[1] https://www.hyperledger.org/projects/fabric
[2] https://www.hyperledger.org/projects/caliper

Before the development of smart contracts, blockchain applications were limited to creating cryptocurrencies and simple monetary transactions. The development of smart contracts has provided the infrastructure for creating more diverse blockchain-based applications. Smart contracts are executable logic encoded in a blockchain with the ability to be enforced automatically.

Blockchain networks can be categorized as public and permissioned.

Public blockchains are open to the world, and every user can join the blockchain with an anonymous identity, submit a transaction, and participate in consensus. Permissioned blockchains include an additional membership layer, such that only authenticated users can join the blockchain and interact with different components.

Today, many blockchain platforms exist, and they are geared toward implementing smart contracts and decentralized applications. They differ in several aspects, such as the type of the network (public or permissioned), built-in cryptocurrency, transaction workflow, performance, privacy, cost and, most importantly, maturity. Some blockchain platforms—such as Ethereum, Hyperledger Fabric or Corda—have mature tools, while others have very little support for their users and developers.

Hyperledger Fabric [4] is a famous implementation of a permissioned blockchain, hosted by the Linux Foundation. Hyperledger Fabric has a modular structure that allows component pluggability, such as consensus, membership, and database.

The membership layer authenticates users and grants access to users based on their access level and system policy. Because of the initial authentication in permissioned blockchains, participants are expected to be trusted or at least semi-trusted. The endorsement policy identifies, for every transaction, which peers must endorse and validate the transaction. Based on the ordering service (consensus mechanism), the system can tolerate different byzantine behaviour levels. Hyperledger Fabric has integrated the ABAC mechanism, so it is possible to build permission groups for access control inside the blockchain network by checking members' attributes. These attributes define the users' access to different elements in the Fabric network, such as transaction submission, consensus, transaction validation and reading the state of the ledger. However, access control parameters and permission groups are predefined tasks involved in the Fabric network, and are not suitable for managing access control in an off-chain application requiring different attributes and policies from those defined in Fabric permission groups.

The modular structure of Fabric allowed to evaluate our solution based on different consensus mechanisms, such as Raft and Kafka, various databases, such as GoLevelDb and CouchDB, and customizable network configuration.

Hyperledger Fabric uses a novel execute-order-validate architecture for transaction flow, unlike other platforms that follow order-execute transaction flow, which has multiple advantages over order-execute architecture. First, Fabric separates the transaction flow into three steps. Different entities may run each step in the blockchain network. Separating transaction execution from consensus allows modular building and including elements of scalable replicated databases. Second, it allows parallel execution in the execute step, which will be validated later against any non-deterministic result, which is an existing drawback in order-execute architecture. Third, it enables policy-based endorsement, resulting in optimized usage of resources based on an application requirement. Hyperledger Fabric has integrated the ABAC mechanism, so it is possible to build permission groups for access control by checking members' attributes. However, access control parameters and permission groups have to be predefined. It is not suitable for applications that require dynamic and flexible access control.

In order to protect the privacy of users' data, Hyperledger Fabric provides a private data feature to protect sensitive users' data. We have used this feature to represent the required attributes for access permissions based on the organizations' defined policies. The private data is hashed, and then it will be endorsed and ordered like other data. Finally, the chaincode writes hashed data on the ledgers of every peer. However, only organizations that require these private attributes to give access permission have access to them. Using Zero knowledge proofs (ZKP) [19] is another alternative that can be applied for highly privacy-preserving case studies that require protecting all users' attributes from all access providers and data owners. However, ZKP requires additional time and computational resources when compared to our solution based on Hyperledger Fabric private data feature.

Our system provides a solution for off-chain parties that look for a flexible and distributed access control service, compatible with their authentication service. Our solution can be easily integrated with any off-chain system, while access control functionality is achieved through blockchain and smart contracts.

Smart contracts correspond to logic encoded in the blockchain that can be programmed and deployed as an automation program. Accordingly, smart contracts can create complex transactions and enforce their conditions automatically [48]. Chaincode is a term introduced by Hyperledger Fabric for smart contracts. Chaincode may consist of multiple smart contracts or include only one smart contract. Both terms—chaincode and smart contract—are used in this work interchangeably.

Before the development of smart contracts, blockchain applications were limited in creating cryptocurrencies and simple monetary transactions. The development of smart contracts provided the infrastructure for creating more diverse blockchain-based applications, such as healthcare [5, 13, 29], Internet of Things (IoT) [27, 37], resource sharing [55, 64], justice [6, 8] and business process management [31, 44, 57]. In a previous work [47], we discussed that although the applications of these systems are different, their primary purpose is similar, as they aim to control access over particular data. The data domain is their main difference, for example, a patient healthcare data or data generated by IoT devices.

2.2 Access control models and ABAC

Access control refers to any action to prevent data and resources from unauthorized access, disclosure or modification. In traditional databases, an authorization is described by a triplet $< o, s, p >$ defining that subject s is authorized to execute privilege p on object o [11].

Conventional access control models follow such definition, considering the context in which an access control request is performed, i.e., a) Discretionary Access Control (DAC) [12], b) Mandatory Access Control (MAC) [10] , and later c) Role-Based Access Control (RBAC) [18] that are the three initial access control models [50].

DAC restricts access permissions based on the subjects' identity, and the resource owner defines the policy rules.

In MAC, the system defines access policies through the security labels. This model is typically used for controlling access to sensitive and confidential data.

In RBAC, there are predefined roles in the system, and users have different access levels depending on their roles.

ABAC is a logical access control that comprises access control lists, role-based access control, and its own method for providing access based on the evaluation of attributes [24]. ABAC controls access to the system resources by evaluating policies (system rules) against entities' attributes, including subject, object, and environmental attributes. Attributes are characteristics of the subjects (users) and protected objects (resources). The environment

conditions—as the environment's attributes—can also be taken into account for ABAC decision making.

ABAC is a flexible and fine-grained mechanism capable of enforcing the other three methods. Distributed systems also adopted ABAC as they require federation and autonomy control among coordinated systems, and ABAC enables granular and meta attribute capabilities that support privilege delegation in a distributed application [25]. Likewise, blockchain-based applications mostly adopt ABAC [47].

The eXtensible Access Control Markup Language (XACML) [3] introduces a policy-based architecture for the specification and enforcement of access control policies. The architecture comprises the following components.

- *Client*: the device that requests access to a resource, possibly on behalf of a user;
- *Policy Enforcement Point (PEP)*: the network device on which access decisions are carried out. PEP serves as the gatekeeper to the intended resource;
- *Policy Information Point (PIP)*: the repository that holds information (attributes) about the client and provides this information to the PDP;
- *Policy Decision Point (PDP)*: the component that decides to allow or deny the client access to the resource;
- *Policy Administration Point (PAP)*: the component that is responsible for managing access control policies;
- *Accounting or Auditing*: The component that is responsible for tracking access attempts.

Figure 2 illustrates how these components interact with the client and with each other. A client requests access permission, and PEP forwards the request to PDP. PDP queries the related policy and attributes from PAP and PIP. After receiving the required information, PDP assesses the access decision and sends the result to PEP for enforcing the access decision.

3 Related work

Many studies on blockchain technology focus on presenting an access control system, either in the context of specific applications, such as healthcare [5, 13, 45, 46, 58, 61], IoT [14, 15, 33, 38, 39, 43, 63], and cloud federation [2, 17] or they introduce a general access

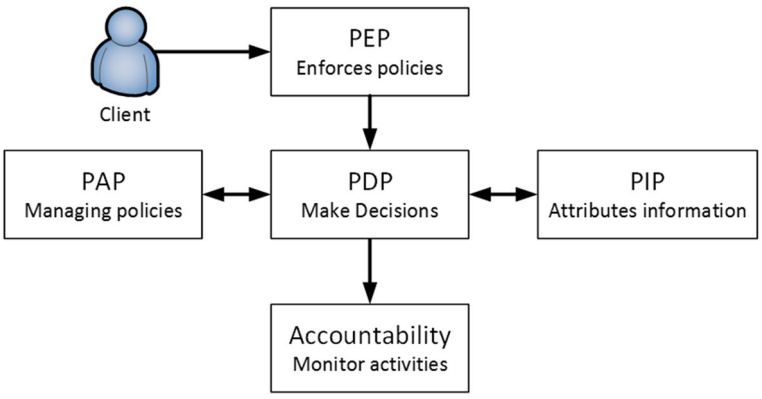

Figure 2 ABAC logical components based on Policy based architecture [3]

control system, which can be employed for different applications. In a previous study [47], we have investigated the state of the art of access control systems based on blockchain. This section provides an overview of similar studies, namely the ones presenting an attribute-based access control based on blockchain. As illustrated in Table 1, many studies use the attributed-based method for their access control system due to the granularity, flexibility, and dynamic features that ABAC provides.

Guo et al. [20] introduce a hybrid architecture for access control over Electronic Health Record (EHR) data using blockchain and edge nodes. The blockchain acts as a tamper-proof validation component to verify identities and access control policies. The edge nodes store the EHR data off-chain and enforce the access controls. The smart contracts include the address of EHR data on the edge nodes by using one-time self-destructing URLs [3]. Based on performance results against unauthorized retrieval for the average transaction, processing time was 40 ms, and the average response time was 30 ms. Also, the test result based on a high number of patients does not affect the response time and indicates their solution's scalability.

Zyskind et al. [66] conceptualizes the blockchain technology as an access control moderator, complemented by an off-blockchain storage solution. Blockchain clients representing users, who provide their data to a service provider, are the owners of their data. Based on that premise, their solution is meant to empower users, so they have the information on which data is collected about them by third parties and how their data is used. For achieving that goal, each data owner can issue transactions to change the set of permissions granted to a service or entity. Each transaction is recorded on the blockchain, allowing for auditability and traceability.

Zhang et al. [63] propose a solution directed to IoT blockchain-based access control. The authors introduce the concepts of *Judge Contract* (JC), *Register Contract* (RC) and *Access Control Contracts* (ACC). Access control contracts store access control policies for a subject-object pair. In this system, both JC and RC are essential pieces regarding achieving a distributed and reliable access control. The JC receives misbehaviour reports and applies penalties according to them. The RC stores the misbehaviour information from the JC and manages it through the judging method. Moreover, it stores information such as name, subject, object, and smart contract for access control.

Zhu et al. [65] propose a transaction based access control (TBAC) system that integrates the ABAC model into the bitcoin blockchain. There are four implemented transactions, including subject registration, object escrowing and publication, access request and grant. They also present a cryptosystem associated with their system as an additional security layer. The system is evaluated in terms of security, but its performance and scalability are not examined.

In federated cloud services, access control enforcement is still vulnerable to privacy violations.

Alansari et al. [2] present an attribute-based access control system based on Pedersen commitment scheme [42] and blockchain. The system is designed to keep the users' attributes private from the federated organization. Users' identity attributes and access control policies are stored in the blockchain to guarantee their integrity. They also employed Trusted hardware technology to guarantee the integrity of the policy enforcement process.

Zhang and Posland [61] propose an architecture for granular access authorization that supports flexible queries and secure authorization, at different levels of granularity. The

[3] https://1ty.me/

Table 1 A summary of blockchain-based access control applications

Research paper	Domain	Access control method	Privacy Support	Scalability
Jemel and Serhrouchni [26]	Data sharing	ABAC + Attribute-based Encryption	✓	x
Wang et al. [56]	Data sharing	ABAC + Attribute-based Encryption	✓	x
Zhu et al. [65]	Resource sharing	ABAC	✓	x
Hu et al. [23]	Knowledge sharing	Fine-grained	x	x
Ferdous et al. [17]	Cloud federation	–	x	x
Alansari et al. [2]	Cloud federation	ABAC	✓	x
Zhang and Posland [61]	Health care	ABAC	✓	x
Rouhani et al. [46]	Health care	Role-based	✓	✓
Guo et al. [20]	Health care	ABAC	x	✓
Asaph et al. [5]	Health care	–	✓	✓
Xia et al. [58]	Health care	–	✓	x
Dagher et al. [13]	Health care	Role-based	✓	✓
Rajput et al. [45]	Health care	Role-based	✓	✓
Zyskind et al. [66]	Mobile applications	Policy-based	✓	✓
Novo [38]	IoT	–	x	✓
Outchakoucht et al. [40]	IoT	Policy-based	x	x
Zhang et al. [63]	IoT	Policy-based and dynamic access control	x	x
Dukkipati et al. [15]	IoT	ABAC	✓	x
Pinno et al. [43]	IoT	ABAC	✓	✓
Ouaddah et al. [39]	IoT	–	✓	✓
Ding et al. [14]	IoT	ABAC	x	✓
Ma et al. [33]	IoT	Generic	✓	✓

Table 1 (continued)

Research paper	Domain	Access control method	Privacy Support	Scalability
Rouhani et al. [49]	Physical access control	Role-based	x	✓
Es-Samaali et al. [16]	Big data management	ABAC	✓	✓
Xh et al. [59]	Space situation awareness	Capability-Based	x	✓
Lyu et al. [32]	Information centric networking	matching-based	✓	x
Paillisse et al. [41]	Multi-administrative domain	–	✓	x
Laurent et al. [36]	General access control	Access Control List	✓	x
Maesa et al. [35]	General access control	ABAC	x	✓
Guo et al. [21]	General access control	ABAC	✓	x
Lee et al. [30]	General access control	Role-based	✓	x
Belchior et al. [7]	General access control	Self Sovereign Identity Based Access Control	✓	✓

designed architecture offers a robust infrastructure without requiring the public key infrastructure (PKI), so it decreases the computation time needed, and is suitable for devices with limited resources in Electronic Medical Record (EMR) systems. As a result, their system can efficiently respond to a requester without exposing unauthorized private data.

Maesa et al. [34, 35] implemented an access control service on top of Ethereum [4]. The blockchain is used to store smart contracts that represent access control policies represented in XACML [3] and to perform the decision process. Such smart contracts are called *Smart Policys (SP_s)*, responsible for the policy evaluation process, by embedding a PDP for a specific access control policy. Each time, an access request needs to be evaluated to make an access decision, the blockchain executes it in a distributed way. The decision is made based on information concerning the users. For this purpose, the concept of Attribute Manager (AM) is introduced. AM_s are the components that manage the attributes of the entities involved in the process, such as subjects, resources, and environmental context. AM_s can update and retrieve their values and are created by an entity, the Attribute Provider (AP) (Figure 3).

Maesa et al. studies in [34, 35] are the closest to our research work. Multiple advantages come from the permissioned nature and novel transnational flow in Hyperledger Fabric. Implementations based on the Ethereum blockchain is costly since, for every operation, a fee called "gas" must be paid. Although it is generally unavoidable for public blockchain systems, for permissioned implementations, it represents an unnecessary additional cost imposed on the system.

Hyperledger Fabric introduces a novel execute-order-validate [4] architecture for transactional flows. This enables modularity and pluggability, parallel execution of transactions, and policy-based endorsement. As a result, Fabric provides more efficiency, customizable configuration, and optimized resource usage based on an application requirement. By exploiting Fabric's modularity, we have evaluated our solution based on different consensus mechanisms, such as Raft and Kafka, various databases, such as GoLevelDb and CouchDB, and customizable network configuration. In comparison, Maesa et al.'s studies [34, 35] are examined through "proof of work" consensus mechanism and Ethereum's default database.

The privacy and security of the data generated by IoT devices are the major concerns of the IoT system due to the extensive scale and distributed nature of IoT networks. In order to protect the users' privacy, many studies have considered blockchain in order to provide secure access control to the IoT data. The authors in [14, 15, 43] presented an ABAC for IoT systems. Dukkipati et al. solutions [15] store system policies off-chain while the work in [43] stores the policies on the Ethereum platform is less vulnerable to security breaches, but more costly. Ding et al. in [14] focus on simplifying the access control protocol to make it lightweight and suitable for IoT devices with limited computing capability and energy resources.

Zhang and Posland in [62] present an architecture for a blockchain-based EMR access with granularity control that supports flexible data queries. The user layer first sends a query that could have different granularity levels, such as block query, attribute query, or mixed query. The agent layer aggregates the query data and authorizes the user, who has access permission for that query. If it is a valid query, it passes the query to the storage layer which would then return the data to the agent layer, that encrypts the query and sends it to the user layer. Lastly, the query is decrypted using the provided access keys. The represented architecture offers a flexible infrastructure to achieve granular access control without requiring Public Key Infrastructure (PKI), so it leads to a decrease in computation time processing.

[4] www.ethereum.org

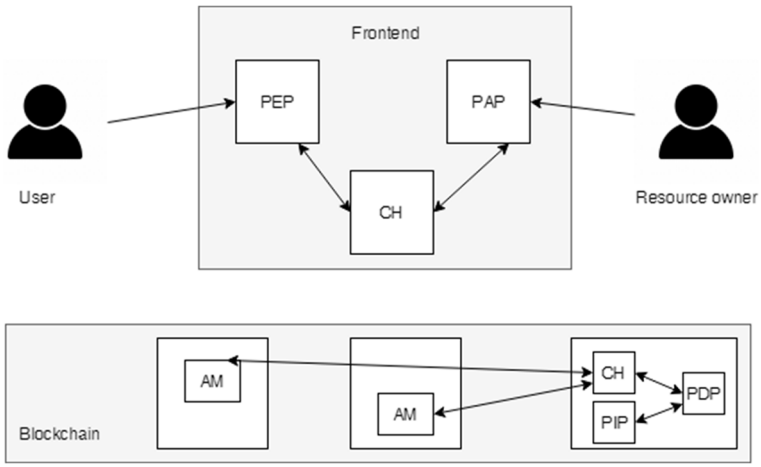

Figure 3 Blockchain-based access control service architecture [34]

Belchior et al. [7] propose an access control system rooted in the self-sovereign identity paradigm. In this scheme, the parties have a decentralized identifier [53] (DID, representing identity) and verifiable credentials [51] (VC, representing attributes). A party controls its VCs and DIDs because they are rooted on a blockchain, and they hold the private keys corresponding to the DIDs and VCs. Access control requests are translated to verifiable presentation requests [51], that leverage ZKP [19] and allow a party to know the other in respect to a certain access control policy, but without disclaiming additional information. Some other works explore this thematic but do not propose an access control model on top of it [22, 28, 54].

As reviewed, many studies have investigated blockchain as a back-end infrastructure for the distributed access control system. However, most of the prior works in this area are domain-specific, meaning that, their access control solutions are designed for a particular domain, such as healthcare data or IoT data. Besides, most of those studies lack the details on their implementation and on performance analysis. As a result, it remains unclear if a blockchain can be the basis for access management at a large scale.

4 System model and architecture

Centralized access control systems suffer from various problems such as [7]: a) the risk of privacy leakage, and b) the risk of a single point of failure; c) interoperability issues; d) unreliability of the access control system, and e) the presence of third parties.

An access control system can utilize blockchain technology to address these problems. The decentralized nature of the blockchain resolves the problem of a single point of failure as, i) cryptography methods ensure the reliability of the ledger; ii) consensus mechanisms ensure that the state of the ledger is valid, and that it is the same for every participant; iii) smart contracts allow monitoring and enforcement of sophisticated access control decisions; iv) with automatic enforcement, it can address privacy issues v) with automatic enforcement of smart contracts, a system does not require third parties in order to manage access control, and vi) privacy is enhanced by avoiding exposing data to third parties, which is a critical data breach source.

This solution empowers both resource owners and subjects (typically access requesters). Details of each granted or revoked access permission are queried from the ledger, including the policies that have been applied, the attribute values and the time of access request. In practice, under no circumstances should resource owners deny access to a resource by a rightful requester. On the other hand, the resource owners leverage provided audit trails while ensuring that no user had subverted the system.

To provide a solution to these problems, we used the Hyperledger Fabric blockchain. In the blockchain network, it is essential to have at least two endorsing nodes to establish credibility. These nodes are responsible for executing SP_s. In our solution, Clients would be systems that use this system for access control management, as depicted in Figure 4.

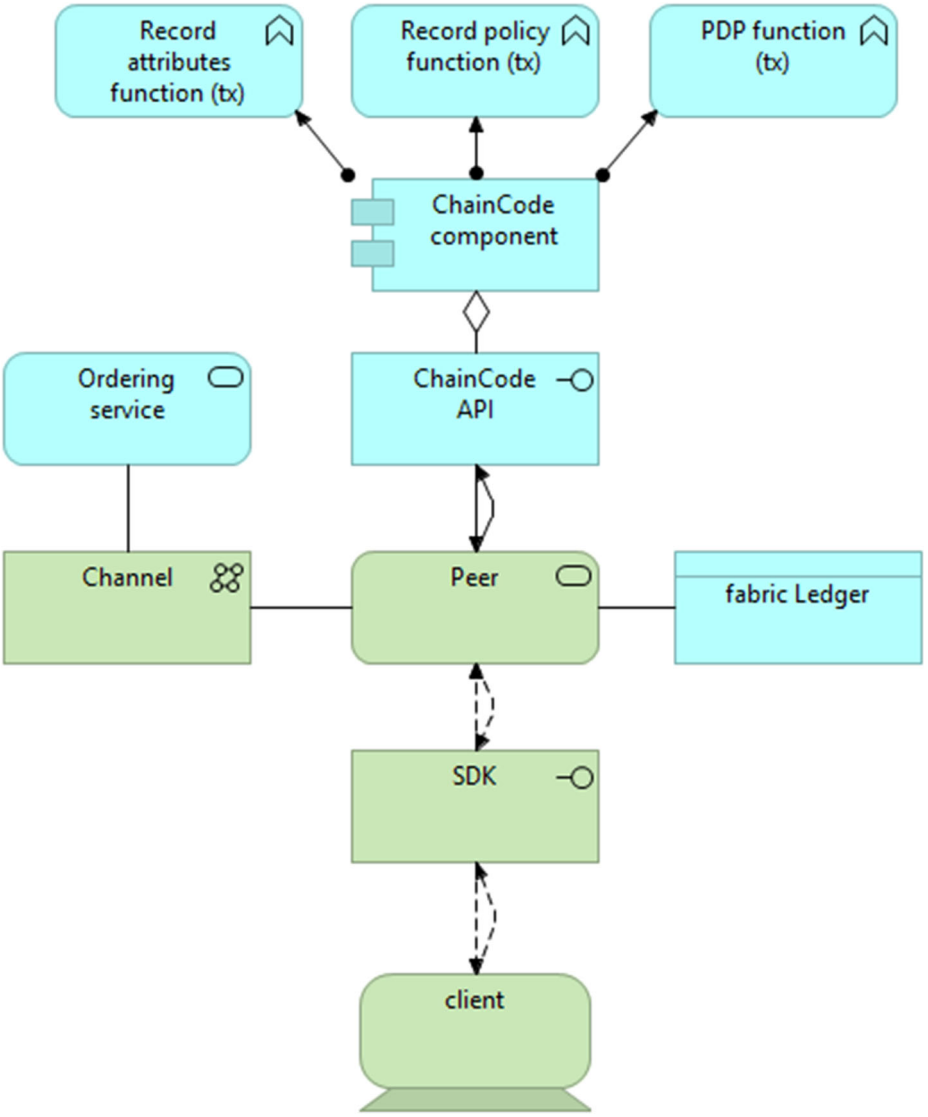

Figure 4 High level system architecture using the Archimate modeling language [52]

The workflow of the users remains unaltered. The only difference is that the authorization requests are now mediated through one or more nodes representing the given system. The evaluation of access control policies could be not trusted for the request's subject, who may request unauthorized access to resources.

4.1 System architecture

In the presented solution, as depicted in Figure 5, the blockchain acts as a mediator between the entity that requests access to a specific resource and the entity that manages that resource. The system includes two main components. The first component is an off-chain system that relies on permissioned blockchain to store its access control attributes and query access permissions. The second component is a permissioned blockchain that manages different access control components through smart contracts and stores the data on a tamper-proof ledger.

The three main smart contracts are PIP contract, PAP smart contract and the PDP contract. Subject (user) and resource (object) attributes are stored in a JavaScript Object Notation (JSON) data format through the PIP contract. The PIP is also responsible for checking write conflicts and updating attributes. Policies are also recorded in the system as JSON data format, and the PAP contract is responsible for managing policies and updating policies. The system could be implemented to work with multiple PAPs, each of which run by a different organization. Even if there is only one PAP, the transparency offered by this solution distributes the trust and the responsibility of these access policies. Conflict detection and resolution among policies is out of the scope of this paper. PDP contract evaluates policies to make an access decision. Figure 6 shows the architecture of the implemented smart contracts.

The pseudocodes of transactions implemented in smart contracts are presented in Algorithms 1, 2, 3 and 4. Algorithm 1 shows how the list of resources and subjects attributes are stored in the ledger. Algorithm 2 presents how policies are recorded in the ledger. Algorithm 3 explains how to query policies and the history of changes to the policies from the ledger. Algorithm 4 shows the detailed implementation of the PDP transaction, which includes querying the relevant attributes and policies from the ledger and making a decision based on their values.

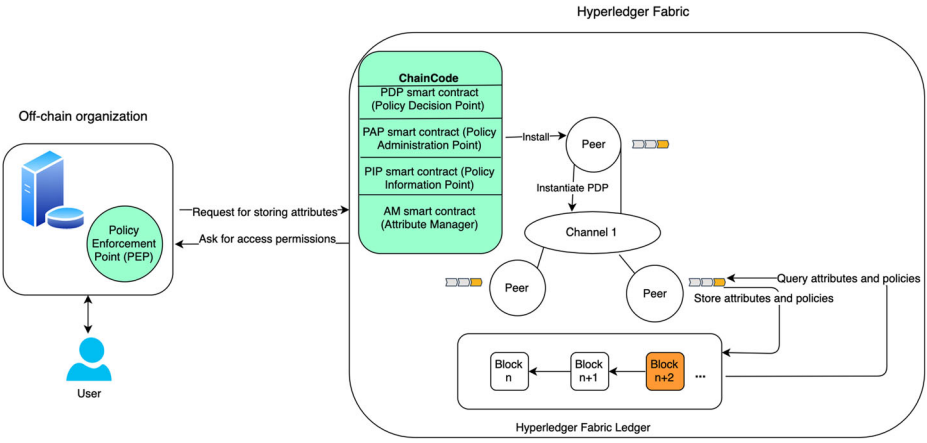

Figure 5 Blockchain access control system architecture

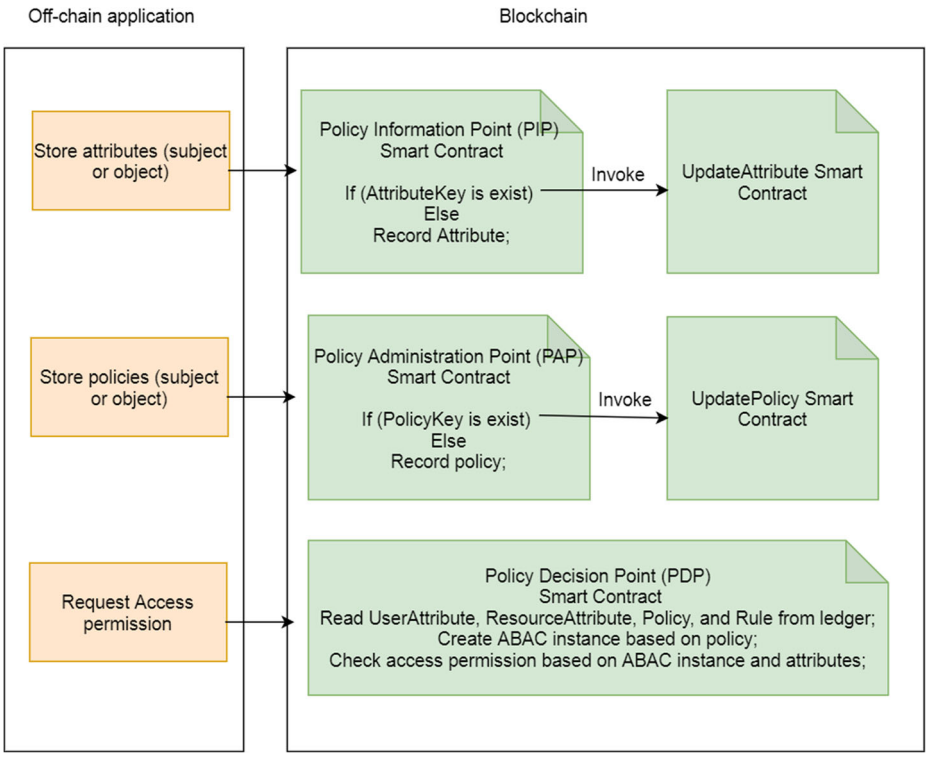

Figure 6 Smart contracts architecture

Algorithm 1 Record attributes transaction.

Input: attributeKey, attributes (subject or resource attributes)
Result: add attributes on the ledger
attributeKey (subject or resource key) ← getState(attributeKey)
if *subjectKey* **then**
    throw error 'attributekey' is already exist you can update the attributes using Update transaction
end
putState(attributeKey, attributes)

Algorithm 2 Record polices transaction.

Input: policyKey, policy
Result: add policies on the ledger
policy ← getState(policyKey)
if *policyKey* **then**
    throw error 'policyKey' is already exist you can update the policy using UpdatePolicy transaction
end
putState(policyKey, policy)

Algorithm 3 Query and trace policies.

```
Input: policyKey
Result: query policies info
policyBytes ← getState(policyKey)
if !policyBytes then
    throw error 'policyKey' does not exist
end
policy ← JSON(policyBytes)
resultsIterator ← getHistoryForKey(policyKey)
policyHistory ← allResults(resultsIterator)
```

Algorithm 4 Policy Decision Point (PDP) transaction.

```
Input: subjectKey, resourceKey, policyKey, rule
Result: permit
subjectBytes ← getState(subjectKey)
resourceBytes ← getState(subjectKey)
policyBytes ← getState(subjectKey)
subjectAttributes ← JSON(subjectBytes)
if !subjectBytes OR subjectBytes.length == 0 then
    throw error subject's attributes does not exist
end
resourceAttributes ← JSON(resourceBytes)
if !resourceBytes OR resourceBytes.length == 0 then
    throw error resource's attributes does not exist
end
policy ← JSON(policyBytes)
if !policyBytes OR policyBytes.length == 0 then
    throw error policy does not exist
end
ABAC = Abac(policy);
permit = ABAC.enforce(rule, subjectAttributes, resourceAttributes)
```

After smart contracts evaluate SP_s against their respective attributes, the PDP returns its decision to the PEP. This process allows a decoupling between users and the blockchain administration (as users do not need to have a node on the blockchain, which is desirable).

Our solution not only logs all access requests in a very secure way but also provides a framework to control and manage all access controls concerning the participants in the network. Nodes from the private network can access the blockchain, check transactions' history, and audit the history of access requests and results. Automatic auditing techniques can be further developed by analyzing the history of access request transactions.

A fine-grained access control solution is therefore provided that enforces access validation through blockchain-based service providers.

5 Case study

A digital library is a collection of documents in an organized electronic form that allows users to access them online. A highly dynamic user population and the numerous collection

of resources in digital libraries require a fine-grained and dynamic access control method such as ABAC [1]. Such a solution requires that access policies be specified based on users' attributes and characteristics rather than users' roles in the system.

In this section, we have selected a case study for an implementation of access control in digital libraries, to illustrate and explain the application of our ABAC system.

For every subject, we store it with a subject ID (SID) together with the set of its attributes and values, and the same is done for Objects attributes. Attribute ID is a required field to store attributes in the Hyperledger Fabric database, in order to later retrieve the respective attribute for permission decision. Both Hyperledger Fabric supported databases (CouchDB and LevelDB) are key-value stores, and the ID field is used as key.

S_nA corresponds to the set of attributes associated with the $subject_n$ and S_nID is defined as a key for storing the correlated attributes. Similarly, O_nA corresponds to the set of attributes associated with the $object_n$ and O_nID is defined as a key for storing the correlated attributes.

P_nSA and P_nOA are the sets of determinative attributes in the nth policy of subjects and Objects. Same as attributes, Policy ID (P_nID) is a required field to store attributes. For every policy, we store the determinative attributes (P_nSA, P_nOA) along with the policy's rules.

$$S_nA = \{S_nID, \{\{S_nA_1, value\}, ..., \{S_nA_n, value\}\}\}$$
$$O_nA = \{O_nID, \{\{O_nA_1, value\}, ..., \{O_nA_n, value\}\}\}$$

Defining SA as the set of all Subjects' attributes and OA as the set of all Objects' attributes, we have:

$$SA = (S_1A \cup S_2A \cup ...S_nA)$$
$$OA = (O_1A \cup O_2A \cup ...O_nA)$$
$$P_nSA \subseteq SA$$
$$P_nOA \subseteq OA$$
$$P_nSA, P_nOA \in P$$
$$P_n = \{P_nID, P_nSA, P_nOA, rules\}$$

The resulting attributes for the Subject S_1A and Object O_1A are consequently (as example):

$$S_1A = \{\text{“}s001\text{”}, \{\text{“}status\text{”}, true\},$$
$$\{\text{“}expiration\text{”}, \text{“}2020-05-12\text{”}\}, \{\text{“}libraryGroup\text{”}, 12\}\}$$

$$O_1A = \{\text{“}r001\text{”}, \{\text{“}libraryGroup\text{”}, 12\}\}$$

The policy P_1 with the identifier “policy01” becomes as follows (as example):

$$P_1 = \{\text{“}policy01\text{”}, S_1A, O_1A,$$
$$\{\text{“}status == true\text{”} \wedge \text{“}expiration\text{”} > \text{“}1Day\text{”} \wedge$$
$$, \text{“}user.libraryGroup\text{”} == \text{“}resource.libraryGroup\text{”}\}\}$$

5.1 JSON data format

In our implemented solution, the access control data (attributes and policies) follow the JSON format, as it is a widespread data format that a broad range of applications can use.

An example of a policy, including Subject (user) attributes and Object (resource) attributes, is illustrated in the following JSON code snippet.

Based on the presented example policy, for the the user (Subject) and the resource (Object) attributes, our usert has valid access permissions to the resource, as the user has valid Identifier (ID), active status, non-expired membership and the user library group matches with the resource library group. If one of the Subject's attributes does not pass the policy rules, the access permission will be denied.

Example: Definition of a sample Access Control Policy, Subject and Resource followed by the JSON format

```
policy = {
''policyID'': ''policy01'',
attributes: {
''user'': {
    ''status'': ''Active'',
    ''expiration'': ''Date of expiration'',
    ''libraryGroup'': ''Group ID''
        },
''resource'': {
    ''libraryGroup'': ''Group ID''
        }
    },
''rules'': {
    ''user.status'': {
    ''comparison_type'': ''boolean'',
    ''comparison'': ''boolAnd'',
    ''value'': true
    },
    ''user.expiration'': {
    ''comparison_type'': ''datetime'',
    ''comparison'': ''isMoreRecentThan'',
    ''value'': ''1DAY''
    },
    ''user.libraryGroup'': {
    ''comparison_target'': ''libraryGroup'',
    ''comparison_type'' ''numeric'',
    ''comparison'': ''isStrictlyEqual'',
    ''field'': ''resource.libraryGroup''
        }
    }
}

subject = {
    subjectID: ''s001'',
    attributes:{
        ''status'': true,
        ''expiration'': ''2020-05-12'',
        ''libraryGroup'': 12
    }
}

resource = {
    resourceID: ''r001'',
    Attributes:{
         ''libraryGroup'': 12
    }
}
```

6 System evaluation

In this section, we evaluate the performance of the system. We used Hyperledger Caliper to generate workloads and measure the performance of our system. For that purpose, our own written benchmark was defined, as well as the various configurations for the different components and network infrastructure, described as follows.

6.1 Environment configuration, performance parameters, and assumptions

We evaluate every component of our system against two different databases, Couchdb and Goleveldb, and two orderer services, Raft and Kafka. We also present the evaluation results based on the Solo orderer to illustrate the effect of other parameters separated from the effect of the involved consensus method.

Raft is a crash fault-tolerant (CFT) ordering service based on the implementation of Raft protocol. Raft follows the "leader and follower" model. The leader makes decisions, and the followers follow the leader. The peer that represents the leader is changed frequently. Every follower has the chance to be a candidate to become the leader of the next round.

Like Raft, Kafka is a CFT ordering service, which follows the "leader and follower" model. However, Kafka uses ZooKeeper[5] to manage clusters. Zookeeper keeps track of the status of the Kafka cluster nodes and partitions.

Solo is an implementation of a single ordering node, and it is not fault-tolerant. It is meant to be used for testing purposes.

We used the Google Cloud Platform to run a Virtual Machine (VM) instance, in order to test our application and collect performance analysis data. The machine type is n2-highcpu-8, which includes eight virtual CPUs and 8GB of memory. All the tests are run on the same virtual machine, as Caliper emulates workload distribution between several clients.

The default number of blockchain clients is 10 (each client emulated by a different NodeJS[6] process). The default number of transactions is 5,000. The default database for Raft and Kafka is GoLevelDB. The default number of organizations is two, and the default number of peers is one.

6.2 Network topology

Hyperledger Fabric offers a distributed infrastructure that allows multiple organizations to interact. The concept of "channel" has been introduced in order to separate the communication between organizations and to provide private data. Each organization can join multiple channels and establish private communication resulting in a separate ledger for each channel. Each organization can contribute with multiple peers, and these peers are responsible for maintaining a copy of the ledger and executing smart contracts. Smart contracts are installed on peers and then defined on the respective channel. The Ordering service consists of multiple nodes—called orderer—responsible for reaching deterministic consensus (unlike public platforms such as Ethereum and Bitcoin with the possibility of a fork) and ordering the transactions and bundle them into blocks. The Endorsement policy, which the involved organization agrees, defines which organization's peers must endorse which transaction to be confirmed.

[5] https://zookeeper.apache.org/
[6] https://nodejs.org/en/

Figure 7 illustrates an example of a Hyperledger Fabric network.

For our use case, it is assumed that we have three organizations offering digital library services. Library A and Library B are interested in offering an integrated service to their users, and the service is limited to their registered users. The first step is defining a consortium and storing it in the network configuration. Channel 1 will then be created based on the network configuration. Each organization (library) contributes one peer to the channel (there would be more peers in a real-world implementation to increase network reliability). In the next step, the *chaincodes* responsible for implementing the ABAC system will be installed on both peers and defined on Channel 1. The data stored on the ledger will be only accessible to the members of Channel 1. It is also possible to restrict the users' access level through Fabric membership service and implement more granular access at the Fabric network level. So far, this reflects the network topology for our implemented application. It includes two organizations (org1 and org2) connecting through Channel 1, two peers (peer0.org1 and peer0.org2).

We have tested our application through two different indorsing services. Raft ordering service was established through three orderer nodes (ordered 0, orderer 1, orderer 2). Raft is a CFT that can withstand one failure out of three nodes. Kafka is established through two orderer nodes (orderer 0 and orderer 1).

As an extension to this scenario, we consider that Library B is also interested in integrating another service with a new organization (Library C). To implement the ABAC access control, it is required to store the related attributes and policies on the ledger. Therefore, they can establish a new channel (Channel 2) and maintain a separated ledger on their peers. The data stored on the new ledger is not exposed to Library A. Library B can either use the same peer (Peer B) for both channels or provide two separate peers to contribute separately in Channel 1 and Channel 2.

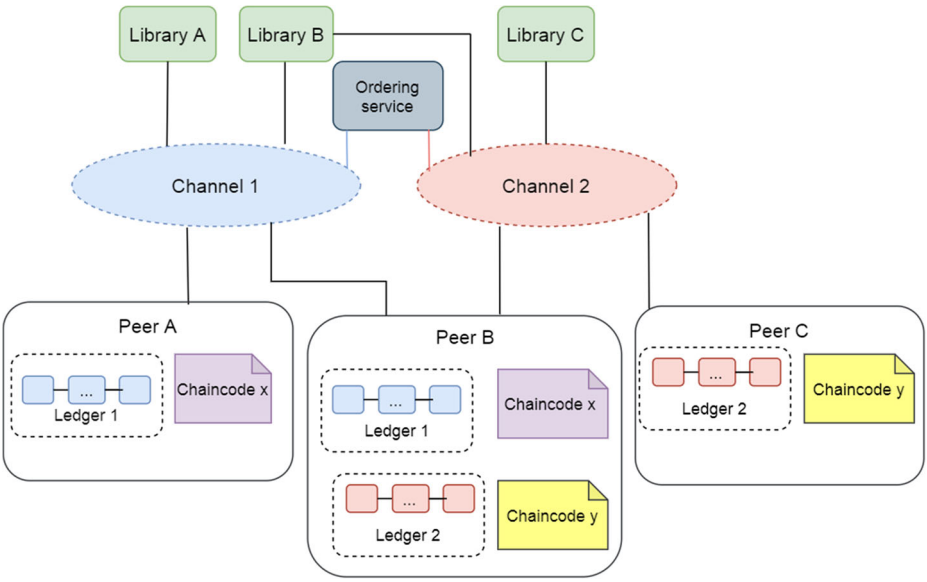

Figure 7 Network topology

6.3 Performance evaluation results

Figure 8 shows the average latency (in seconds) for three different transactions, *Record* attributes and policies, PDP, and *Query* data from the ledger based on Kafka orderer. The *Record* transaction includes storing both subjects and objects resources attributes, and policies; there are three transactions, *Record* objects' attributes, *Record* subjects' attributes and *Record* policies. Since these three transactions basically do the same task, their extracted results were very similar. The average latency of these three transactions is considered for *Record*. The *Query* performs queries to the stored data in the ledger, including attributes and policies. The PDP queries the related data from the ledger based on the access request and determines the result of the access request. In every round of the test, we configured it with a different number of transactions. Although the average latency increases with the number of transactions, the increase is not sharp, increasing very slowly. Analyzing the results from the tests (described in the following) indicates that the system was able to process with a throughput of around 270 transactions per second (PDP decisions) and with an average latency of 0.54 seconds.

Figure 9 shows the average latency (in seconds) for the same three different transactions, based on the Raft orderer. The test result is based on a different number of transactions. Similarly, the average latency increases with the increase in the number of transactions. The test run failed for the Raft orderer when reaching 10,000 transactions due to an "*out of memory*" error. This happened due to the limited memory resources of the testing environment. The resource consumption result (Table 2) indicates that Raft consumes 4.33 times more memory in comparison with Kafka. This fact explains why the test failed in the middle of executing 10,000 transactions with the Raft orderer.

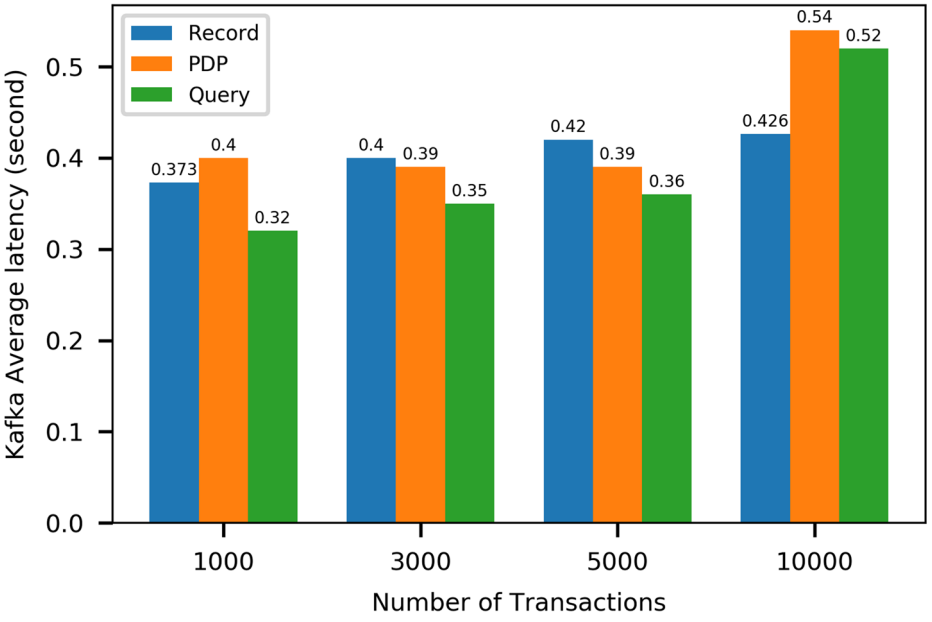

Figure 8 Average latency of Kafka as a function of the type of transactions

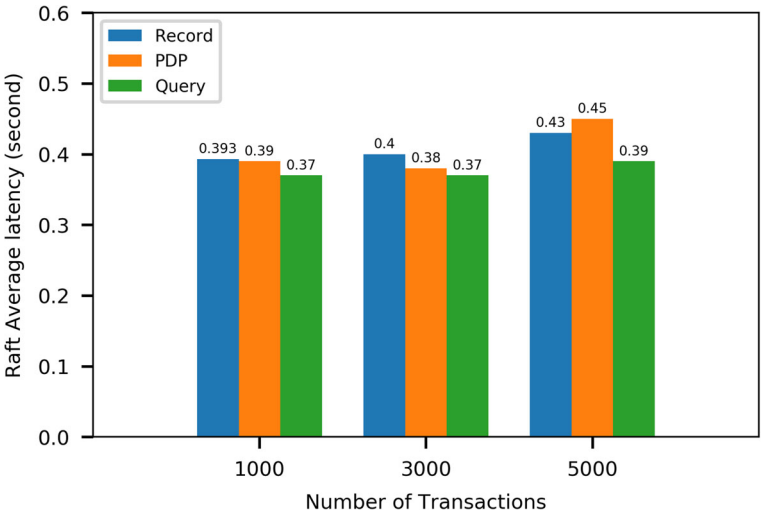

Figure 9 Average latency of Raft under different transactions

For both Raft and Kafka orderers, increasing the number of transactions increases the average latency for the *Record* attributes transaction. Policy decision transaction has the minimum average latency for both Kafka and Raft, based on 3,000 transactions. For *Record* attribute and *Query* data transactions, the average latency with Raft is slightly higher than with Kafka. For the Policy decision transaction, the average latency with Raft for 1,000 and 3,000 transactions is slightly lower than with Kafka's, but for 5,000 transactions, the result is the opposite.

Figures 10 and 11 show the average latency and throughput for the Policy decision transaction for Raft and Kafka based on different sending rates. The number of transactions for these two tests is 5,000. The dendrite of 200 transactions per second (tps) is an optimal point for Kafka as it exhibits the lowest average latency, and the average latency increases sharply after the sending rate of 200 tps.

The maximum throughput for both Raft and Kafka orderers is at the sending rate point of 300 tps; afterwards, the throughput drops for both of them. Overall, in terms of throughput and average latency, Raft performed better than Kafka when the throughput passed 200 tps, a turning point for Kafka.

Figure 12 shows the effect of increasing the number of organizations and peers and comparing two different databases, GoLevelDB and CouchDB. Increasing the number of organizations and peers increases the average latency for all three transactions. For the CouchDB database with three organizations and two peers, the average latency increases sharply to 43.81 seconds for the Policy decision transaction, which is 17.73 times more than GolevelDB and 64.42 times more than the same database, with two organizations and one peer. It also shows that CouchDB performs inadequately in comparison with GoLevelDB.

Figure 13 shows the average latency for both Raft and Kafka based on the different number of clients. This test is run based on 5,000 transactions and the Policy decision transaction. For Kafka, 25 is likely the optimal point for the number of clients, for which we observed the lowest average latency (0.3 seconds). However, 25 is an optimal point for the system with the current resources. As the graph shows, for Raft, there are two points for the

Table 2 Resource consumption for Raft and Kafka

Orderer	Name	Memory(max)	Memory(avg)	CPU(max)	CPU(avg)	Traffic In	Traffic Out	Disc Read	Disc Write
Raft	dev-peer0.org1	74.4MB	71.MB	32.25%	28.21%	13.8MB	5.3MB	0B	0B
	dev-peer0.org2	73.3MB	69.7MB	33.23%	28.28%	13.8MB	5.3MB	0B	0B
	peer0.org1	379.3MB	369.4MB	67.21%	56.32%	31.1MB	23.7MB	0B	21.8MB
	peer0.org2	284.0MB	274,1MB	70.78%	55.66%	31.1MB	23.6MB	4.0KM	21.8MB
	orderer1	554.1MB	535.4MB	26.11%	15.41%	22.4MB	59.1MB	0B	37.2MB
	orderer2	525.3MB	506.5MB	18.78%	11.88%	27.6MB	28.7MB	0B	37.0MB
	orderer0	513.3MB	494.7MB	19.40%	11.63%	27.5MB	10.6MB	0B	37.2MB
Kafka	dev-peer0.org1	73.5MB	72.8MB	17.87%	15.26%	7.5MB	2.5MB	0B	0B
	dev-peer0.org2	64.5MB	62.8MB	18.73%	15.69%	7.5MB	2.5MB	0B	0B
	peer0.org1	295.9MB	286.2MB	52.08%	49.15%	27.1MB	17.0MB	368.0KB	21.2MB
	peer0.org2	294.1MB	282.7MB	51.54%	48.38%	27.1MB	17.1MB	152.0KB	21.2MB
	orderer0	121.1MB	113.1MB	25.80%	23.30%	29.6MB	11.1MB	4.0KB	18.4MB
	orderer1	124.1MB	115.2MB	21.87%	20.25%	29.7MB	46.5MB	276.0KB	18.4MB

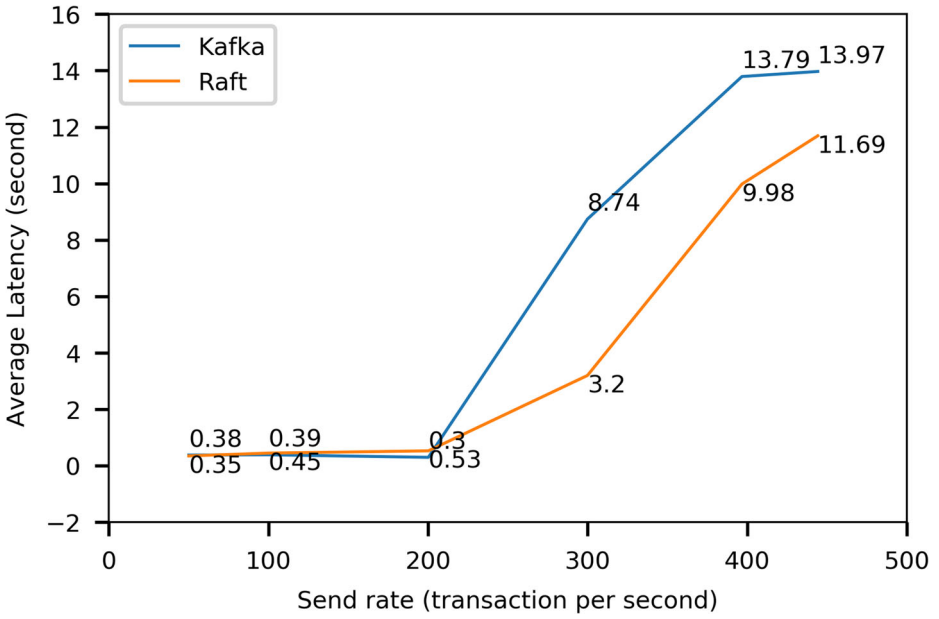

Figure 10 Average latency of Raft and Kafka based on different transactions

minimum average latency, corresponding to 20 and 25 clients. In general, Kafka performs better under 25 clients, but after 25 clients, it shows a sharp increase in the average latency. Although these results appear to show that the system does not scale beyond 25 clients, it

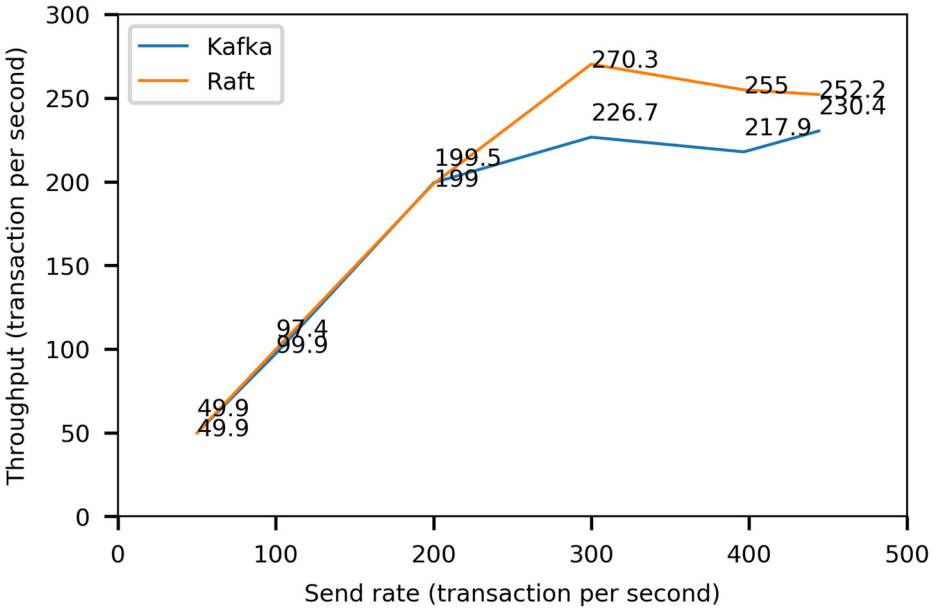

Figure 11 Throughput of Raft and Kafka under different transactions

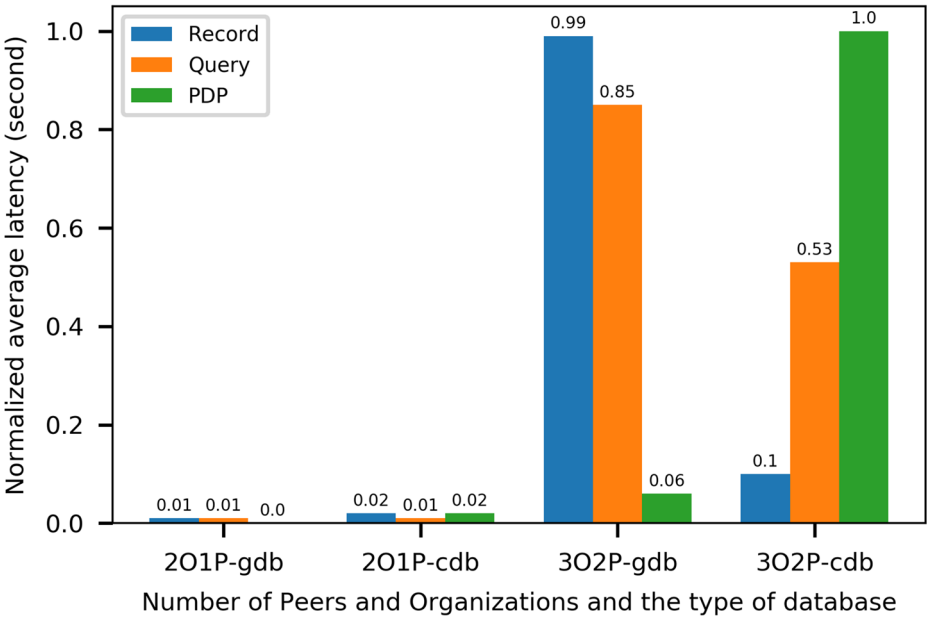

Figure 12 Throughput of Raft and Kafka under different transactions. Legend: xOnP = x Organizations and n Peers; gdb = GoLevelDB; cdb = CouchDB

significantly depends on the computation power and limitations of the VM instance. We have repeated the test with a more powerful VM instance, and the average latency was 0.36 second for Raft and 0.31 second for Kafka, with 60 clients.

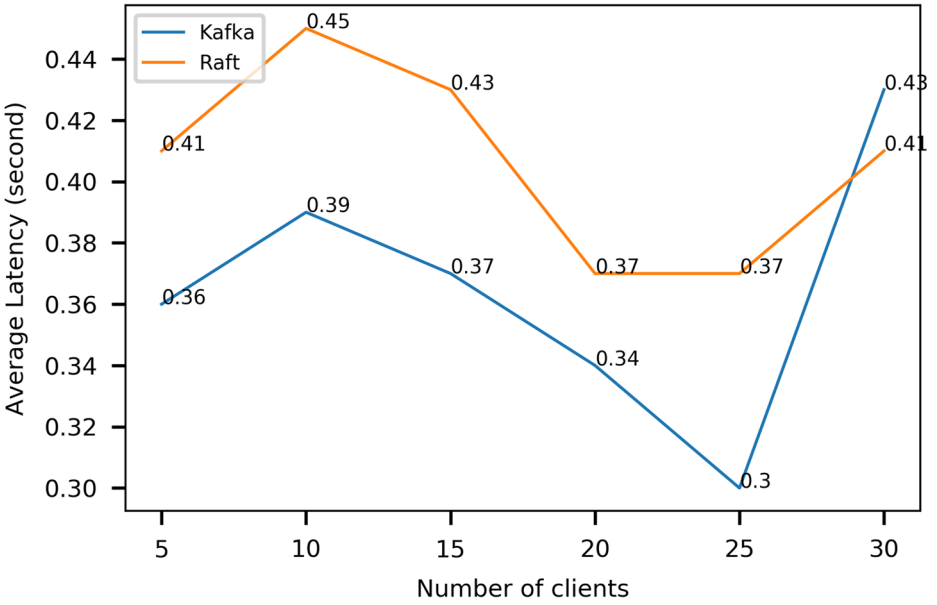

Figure 13 Average latency based of Raft and Kafka with different number of clients

Table 2 presents the resource consumption for Policy decision transactions based on 5,000 transactions for Kafka and Raft. As can be observed in that table, Raft consumes 4.33 times more memory on average in comparison with Kafka. This clarifies that our early test with 10,000 transactions with Raft orderer failed because the VM ran out of memory.

6.4 Security considerations

Our system alleviates security and privacy concerns that centralized access control systems suffer by offering a pluggable distributed application for access control that provides reliable auditing data for accountability and non-repudiation. However, there are additional precautions that need to be taken. With our design, engineers are allowed to reuse our access control component, which is decoupled from the underlying resources that are protected. While this openness provides ease of implementation, new concerns arise. We need to ensure that there are no other ways to access protected resources. It is also essential to ensure that the local world state database is not tampered with or accessed by the system administrators. A possible solution to address those limitations includes enforcing only one access control channel (the blockchain) and a secure, tamper-proof storage for logs (for example, another blockchain) [8]. Although using blockchain to store resources increases its resiliency, there are data integrity vulnerabilities [8]. Further research is needed to assess the practical implications of such an approaches.

7 Conclusion

Reliable accountability mechanisms are essential for audits. In this paper, we discussed how permissioned blockchains could be helpful as trustable backends in access control systems, thus providing a solid basis for audits. We proposed a distributed ABAC system based on Hyperledger Fabric, focusing on audibility and scalability. We validated our solution through a Use Case of a decentralized access control management application in digital libraries. First, we presented a comprehensive review of studies focusing on blockchain-based access control studies. Then we presented the system architecture and implementation details, where the PDP, PAP, and AM components have been implemented using smart contracts on-chain, and the PEP was implemented off-chain—based on the blockchain clients' requirements.

The experimental evaluation of our solution considered various parameters based on the Hyperledger Caliper framework in terms of system performance. The analysis of the results indicate that our proof-of-concept system effectively handle a throughput of 270 transactions per second, with an average latency of 0.54 seconds per transaction.

Future work is in progress in two directions: first, building a robust framework and platform-independent solution towards distributed access control while emphasizing data integrity and privacy threats on distributed access control methods; second, integrating user authentication into our authorization solution.

Acknowledgements This research is supported by the Linux Foundation in the context of the Hyperledger Fabric Based Access Control Project.

References

1. Adam, N.R., Atluri, V., Bertino, E., Ferrari, E.: A content-based authorization model for digital libraries. IEEE Trans Know Data Eng **14**(2), 296–315 (2002)
2. Alansari, S., Paci, F., Sassone, V.: A distributed access control system for cloud federations. In: Distributed Computing Systems (ICDCS), 2017 IEEE 37th International Conference On, pp. 2131–2136. IEEE (2017)
3. Anderson, A., Parducci, B., Carlisle Adams, E.: Oasis extensible access control markup language (xacml). Presentation to XML Community of Practice Architecture and Infrastructure Committee of the CIO Council (2006)
4. Androulaki, E., Barger, A., Bortnikov, V., Cachin, C., Christidis, K., Caro, A.D., Enyeart, D., Ferris, C., Laventman, G., Manevich, Y., et al.: Hyperledger fabric: A distributed operating system for permissioned blockchains. In: Proceedings of the thirteenth EuroSys conference, pp. 1–15 (2018)
5. Azaria, A., Ekblaw, A., Vieira, T., Lippman, A.: Medrec: Using blockchain for medical data access and permission management. In: Proceedings - 2016 2nd International Conference on Open and Big Data OBD 2016, pp. 25–30 (2016). https://doi.org/10.1109/OBD.2016.11
6. Belchior, R., Correia, M., Vasconcelos, A.: Justicechain: Using blockchain to protect justice logs. In: OTM Confederated International Conferences on the Move to Meaningful Internet Systems, pp. 318–325. Springer (2019)
7. Belchior, R., Putz, B., Pernul, G., Correia, M., Vasconcelos, A., Guerreiro, S.: SSIBAC: Self-Sovereign identity based access control. In: The 3rd International Workshop on Blockchain Systems and Applications. IEEE (2020)
8. Belchior, R., Vasconcelos, A., Correia, M.: Towards secure, decentralized, and automatic audits with blockchain. In: European Conference on Information Systems (2020)
9. Belchior, R., Vasconcelos, A., Guerreiro, S., Correia, M.: A survey on blockchain interoperability: Past, present, and future trends. arXiv **1**(1), 58 (2020). arXiv:2005.14282
10. Bell, E.D., La Padula, J.L.: Secure computer system: Unified exposition and multics interpretation (1976)
11. Bertino, E., Weigand, H.: An approach to authorization modeling in object-oriented database systems. Data Knowl Eng **12**(1), 1–29 (1994)
12. Biba, K.: Integrity considerations for secure computer systems. Tech. rep., Bedford MA: Mitre Corporation (1977)
13. Dagher, G.G., Mohler, J., Milojkovic, M., Marella, P.B.: Ancile: Privacy-preserving framework for access control and interoperability of electronic health records using blockchain technology. Sustain Cities Soc **39**(February), 283–297 (2018). https://doi.org/10.1016/j.scs.2018.02.014
14. Ding, S., Cao, J., Li, C., Fan, K., Li, H.: A novel attribute-based access control scheme using blockchain for iot. IEEE Access **7**, 38431–38441 (2019)
15. Dukkipati, C., Zhang, Y., Cheng, L.C.: Decentralized, blockchain based access control framework for the heterogeneous internet of things. In: Proceedings of the Third ACM Workshop on Attribute-Based Access Control, pp. 61–69. ACM (2018)
16. Es-Samaali, H., Outchakoucht, A., Leroy, J.P.: A blockchain-based access control for big data. Int J Comput Netw Commun Secur **5**(7), 137 (2017)
17. Ferdous, M.S., Margheri, A., Paci, F., Yang, M., Sassone, V.: Decentralised runtime monitoring for access control systems in cloud federations. In: Distributed Computing Systems (ICDCS), 2017 IEEE 37th International Conference On, pp. 2632–2633. IEEE (2017)
18. Ferraiolo, D., Sandhu, R., Gavrila, S., Kuhn, D., Chandramouli, R.: A proposed standard for Role-Based access control. ACM Trans. Inform. Syst. Secur. **4**(3) (2001)
19. Goldreich, O., Oren, Y.: Definitions and properties of zero-knowledge proof systems. J. Cryptol. **7**, 1–32 (1994)
20. Guo, H., Li, W., Nejad, M., Shen, C.C.: Access control for electronic health records with hybrid blockchain-edge architecture. arXiv:1906.01188 (2019)
21. Guo, H., Meamari, E., Shen, C.C.: Multi-authority attribute-based access control with smart contract. In: Proceedings of the 2019 International Conference on Blockchain Technology, pp. 6–11. ACM (2019)
22. Houtan, B., Hafid, A.S., Makrakis, D.: A survey on Blockchain-Based Self-Sovereign patient identity in healthcare. IEEE Access **8**, 90478–90494 (2020)
23. Hu, S., Hou, L., Chen, G., Weng, J., Li, J.: Reputation-based distributed knowledge sharing system in blockchain. In: Proceedings of the 15th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services, pp. 476–481. ACM (2018)

24. Hu, V.C., Ferraiolo, D., Kuhn, R., Friedman, A.R., Lang, A.J., Cogdell, M.M., Schnitzer, A., Sandlin, K., Miller, R., Scarfone, K., et al.: Guide to attribute based access control (abac) definition and considerations (draft). NIST Spec. Publ. **800**(162) (2013)
25. Hu, V.C., Kuhn, D.R., Ferraiolo, D.F.: Access control for emerging distributed systems. Computer **51**(10), 100–103 (2018). https://doi.org/10.1109/MC.2018.3971347
26. Jemel, M., Serhrouchni, A.: Decentralized access control mechanism with temporal dimension based on blockchain. In: 2017 IEEE 14th International Conference on E-Business Engineering (ICEBE), pp. 177–182. IEEE (2017)
27. Khan, M.A., Salah, K.: Iot security: Review, blockchain solutions, and open challenges. Futur. Gener. Comput. Syst. **82**, 395–411 (2018). https://doi.org/10.1016/j.future.2017.11.022
28. Kondova, G., Erbguth, J.: Self-sovereign identity on public blockchains and the gdpr 342–345 (2020)
29. Kuo, T.T., Kim, H.E., Ohno-Machado, L.: Blockchain distributed ledger technologies for biomedical and health care applications. J. Am. Med. Inform. Assoc. **24**(6), 1211–1220 (2017)
30. Lee, Y., Lee, K.M.: Blockchain-based rbac for user authentication with anonymity. In: Proceedings of the Conference on Research in Adaptive and Convergent Systems, pp. 289–294. ACM (2019)
31. López-Pintado, O., García-bañuelos, L., Dumas, M., Weber, I.: Caterpillar: A blockchain-based business process management system. In: Proceedings of the BPM Demo Track and BPM Dissertation Award co-located with 15th International Conference on Business Process Modeling (BPM 2017), Barcelona, Spain (2017)
32. Lyu, Q., Qi, Y., Zhang, X., Liu, H., Wang, Q., Zheng, N.: Sbac: a secure blockchain-based access control framework for information-centric networking. J. Netw. Comput. Appl. **149**, 102444 (2020)
33. Ma, M., Shi, G., Li, F.: Privacy-oriented blockchain-based distributed key management architecture for hierarchical access control in the iot scenario. IEEE Access **7**, 34045–34059 (2019)
34. Maesa, D.D.F., Mori, P., Ricci, L.: Blockchain based access control. In: IFIP International Conference on Distributed Applications and Interoperable Systems, pp. 206–220. Springer (2017)
35. Maesa, D.D.F., Mori, P., Ricci, L.: A blockchain based approach for the definition of auditable access control systems. Comput. Secur. **84**, 93–119 (2019)
36. Maryline, L., Nesrine, K., Christian, L.: A blockchain based access control scheme. In: Proceedings of the 15th International Joint Conference on e-Business and Telecommunications, pp. 168–176 (2018)
37. Novo, O.: Blockchain meets iot: An architecture for scalable access management in IoT. IEEE Int. Things J. **5**(2), 1184–1195 (2018). https://doi.org/10.1109/JIOT.2018.2812239
38. Novo, O.: Blockchain meets iot: an architecture for scalable access management in iot. IEEE Int. Things J. **5**(2), 1184–1195 (2018)
39. Ouaddah, A., Abou Elkalam, A., Ait Ouahman, A.: Fairaccess: A new blockchain-based access control framework for the internet of things. Secur. Commun. Netw. **9**(18), 5943–5964 (2016)
40. Outchakoucht, A., Hamza, E., Leroy, J.P.: Dynamic access control policy based on blockchain and machine learning for the internet of things. Int. J. Adv. Comput. Sci. Appl **8**(7), 417–424 (2017)
41. Paillisse, J., Subira, J., Lopez, A., Rodriguez-Natal, A., Ermagan, V., Maino, F., Cabellos, A.: Distributed access control with blockchain. arXiv:1901.03568 (2019)
42. Pedersen, T.P.: Non-interactive and information-theoretic secure verifiable secret sharing. In: Annual International Cryptology Conference, pp. 129–140. Springer (1991)
43. Pinno, O.J.A., Grégio, A.R.A., De Bona, L.C.: Controlchain: a new stage on the iot access control authorization. Concur. Comput. Pract. Exper. e5238 (2019)
44. Pourheidari, V., Rouhani, S., Deters, R.: A case study of execution of untrusted business process on permissioned blockchain. 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) (September), pp. 1129–1136 (2018). https://doi.org/10.1109/Cybermatics
45. Rajput, A.R., Li, Q., Ahvanooey, M.T., Masood, I.: Eacms: emergency access control management system for personal health record based on blockchain. IEEE Access **7**, 84304–84317 (2019)
46. Rouhani, S., Butterworth, L., Simmons, A.D., Humphery, D.G., Deters, R., Medichain, T.M.: A secure decentralized medical data asset management system. 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) (September), pp. 1129–1136 (2018). https://doi.org/10.1109/Cybermatics
47. Rouhani, S., Deters, R.: Blockchain based access control systems: State of the art and challenges. In: IEEE/WIC/ACM International Conference on Web Intelligence, WI '19, pp. 423–428. ACM, New York (2019). https://doi.org/10.1145/3350546.3352561

48. Rouhani, S., Deters, R.: Security, performance, and applications of smart contracts: A systematic survey. IEEE Access **7**, 50759–50779 (2019). https://doi.org/10.1109/ACCESS.2019.2911031
49. Rouhani, S., Pourheidari, V., Deters, R.: Physical access control management system based on permissioned blockchain. 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) (2019)
50. Sandhu, R.S., Samarati, P.: Access control: Principle and practice. IEEE Commun. **32**(9), 40–48 (1994)
51. Sporny, M., Longley, D., Chadwick, D.: Verifiable credentials data model 1.0. https://www.w3.org/TR/vc-data-model/ (2020)
52. TO Group: ArchiMate®3.0 Specification. Van Haren Publishing, Netherlands (2016)
53. W3C: Decentralized identifiers (DIDs) v1.0. https://w3c.github.io/did-core/ (2020)
54. Wang, F., De Filippi, P.: Self-Sovereign Identity in a globalized world: Credentials-Based identity systems as a driver for economic inclusion. Front. Blockchain **2**, 28 (2020)
55. Wang, S., Zhang, Y., Zhang, Y.: A blockchain-based framework for data sharing with fine-grained access control in decentralized storage systems. IEEE Access **6**, 38437–38450 (2018). https://doi.org/10.1109/ACCESS.2018.2851611
56. Wang, S., Zhang, Y., Zhang, Y.: A blockchain-based framework for data sharing with fine-grained access control in decentralized storage systems. IEEE Access **6**, 38437–38450 (2018)
57. Weber, I., Xu, X., Riveret, R., Governatori, G., Ponomarev, A., Mendling, J.: Untrusted business process monitoring and execution using blockchain. In: International Conference on Business Process Management, pp. 329–347. Springer (2016)
58. Xia, Q., Sifah, E.B., Asamoah, K.O., Gao, J., Du, X., Guizani, M.: Medshare: Trust-less medical data sharing among cloud service providers via blockchain. IEEE Access **5**, 14757–14767 (2017)
59. Xu, R., Chen, Y., Blasch, E., Chen, G.: Exploration of blockchain-enabled decentralized capability-based access control strategy for space situation awareness. Opt. Eng. **58**(4), 041609 (2019)
60. Yuan, E., Tong, J.: Attributed based access control (Abac) for Web services. In: IEEE International Conference on Web Services (ICWS'05). IEEE (2005)
61. Zhang, X., Poslad, S.: Blockchain support for flexible queries with granular access control to electronic medical records (Emr). In: 2018 IEEE International Conference on Communications (ICC), pp. 1–6. IEEE (2018)
62. Zhang, X., Poslad, S.: Blockchain support for flexible queries with granular access control to electronic medical records (Emr). In: 2018 IEEE International Conference on Communications (ICC), pp. 1–6. IEEE (2018)
63. Zhang, Y., Kasahara, S., Shen, Y., Jiang, X.: Jianxiongwan: Smart contract-based access control for the internet of things. IEEE Int. Things J. **6**(2), 1594–1605 (2019)
64. Zhu, Y., Qin, Y., Gan, G., Shuai, Y., Chu, W., Cheng, C.: TBAC: Transaction-Based access control on blockchain for resource sharing with cryptographically decentralized authorization. Proc. Int. Comput. Softw. Appl. Conf. **1**, 535–544 (2018). https://doi.org/10.1109/COMPSAC.2018.00083
65. Zhu, Y., Qin, Y., Gan, G., Shuai, Y., Chu, W.C.C.: Tbac: Transaction-based access control on blockchain for resource sharing with cryptographically decentralized authorization. In: 2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC), vol. 1, pp. 535–544. IEEE (2018)
66. Zyskind, G., Nathan, O., Pentland, A.S.: Decentralizing privacy: Using blockchain to protect personal data. In: IEEE Security and Privacy Workshops, pp. 180–184 (2015)

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.