Exploiting Blockchain Technology for Attribute Management in Access Control Systems

Damiano Di Francesco Maesa[1 2], Alessio Lunardelli[2], Paolo Mori[2], and Laura Ricci[3]

[1] Department of Computer Science and Technology, University of Cambridge, UK
d.difrancesco@for.unipi.it
[2] Istituto di Informatica e Telematica, Consiglio Nazionale delle Ricerche, Italy
alessio.lunardelli@iit.cnr.it, paolo.mori@iit.cnr.it
[3] Department of Computer Science, University of Pisa, Italy
ricci@di.unipi.it

Abstract. Access Control systems are a key resource in computer security to properly manage the access to digital resources. Blockchain technology, instead, is a novel technology to decentralise the control and management of a shared state, representing anything from a data repository to a distributed virtual machine. We propose to integrate traditional Access Control systems with blockchain technology to allow the combined system to inherit the desirable properties blockchain technology provides, mainly transparency and, consequently, auditability. Depending on the application scenario considered, for some systems it may not be desirable to employ a fully decentralised approach. As such, in this paper we outline how our proposal can be adapted to allow for the minimal possible integration of blockchain technology in a traditional Access Control system. In particular, we consider the scenario where Attribute Managers only may be managed on chain through smart contracts. We provide a proof of concept implementation based on Ethereum, and show its performance through experimental results.

Keywords: Distributed Ledger · Blockchain · Smart Contract · Ethereum · Access Control · XACML

1 Introduction

Thanks to the wide availability of Internet connection, a very large number of digital resources of any kind are nowadays shared among a potentially huge number of users. Guaranteeing the security of such resources is a main concern resulting from their sharing, and regulating the related accesses is one of the security measures which must be taken. Several Access Control models have been proposed in the scientific literature to define which factors have to be taken into account in the decision process to determine whether a given user should be granted the right to access a resource in a certain environment, and several Access Control systems have been developed to implement such models.

A widely adopted Access Control model is the Attribute-Based Access Control one (ABAC) [1], where the decision process exploits the attributes describing the features of the subject, resource and environment involved in the access context. Examples of attributes of subjects could be: their IDs, the IDs of the companies they work for, their role in such companies, the name of the projects assigned to them. Examples of resources could be: documents, computers, and Internet of Things devices, while some examples of attributes could be: the project a document belongs to or the privacy level assigned to such document (e.g., *public*, *internal* or *confidential*). A very simple example of ABAC policy exploiting the previous attributes could be the following: a subject S is allowed to access a given document D if D belongs to one of the projects assigned to S.

The attributes required for the evaluation of ABAC policies are stored and managed by Attribute Managers (AMs), which are queried by the Access Control system to get the current values of such attributes when an access decision has to be taken. AMs can be embedded in the Access Control systems and/or they can be external, such as third party services (e.g., companies' attribute Data Bases, LDAP services or SAML Attribute Authorities for subjects' attributes).

This paper proposes to exploit blockchain technology for implementing Attribute Managers, and to integrate them in traditional Access Control systems alongside traditional ones. The main advantage introduced by blockchain based AMs is the auditability of attribute values. As a matter of fact, the blockchain eternally keeps trace of all the changes in the attribute values. This way, at any moment, the subject could check the values of the attributes at any time in the past, in order to verify whether the response to an access request he submitted was correct or not. It is important to notice that integrating blockchain based AMs in an Access Control system to manage some attributes would not prevent such system to use, alongside them, also traditional AMs to manage other attributes.

The rest of the paper is structured as follows. Section 2 provides some background about Access Control systems and blockchain technology. In Section 3 we analyze the available related works. Our proposal is presented in detail in Section 4, while Section 5 evaluate its main advantages and drawbacks. In Section 6 we measure the experimental performance of our proof of concept implementation. Finally, Section 7 presents our conclusions and possible future work.

2 Background

2.1 XACML based Access Control Systems

The eXtensible Access Control Markup Language (XACML) standard defines a XML based language to write policies, access requests and responses, and a reference architecture for policy evaluation and enforcement (Figure 1). In the following, we give a brief overview of the reference architecture, while a detailed description of the XACML standard can be found in [2].

The *Policy Enforcement Point (PEP)* is the component of the XACML reference architecture tasked with intercepting the access requests performed by

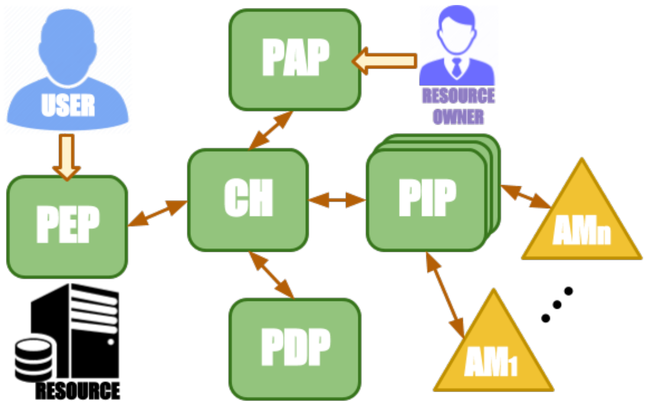


Fig. 1. XACML reference architecture (an arrow represents a communication link between different components).

users, in order to trigger the policy evaluation process and to enforce the related result by actually performing or blocking the execution of the requested access. The *Context Handler (CH)* receives requests from the PEP and it coordinates the execution of the decision process interacting with the other components. The *Policy Administration Point (PAP)* is in charge of storing and managing policies, in order to retrieve them when they are necessary for evaluating access requests. The *Policy Decision Point (PDP)* is the engine that takes an access request and the current attribute values as input, and evaluates the policy to return to the CH the related access decision. *Attribute Managers (AMs)* are the components that actually store and manage the attributes of subjects, resources, and environment. They are queried each time an access request must be evaluated in order to retrieve the updated values of the required attributes. AMs could be part of the Access Control system itself, they could be run by the resource owner, or they can be services run by third parties. In the latter case, the resource owner should trust these AMs, because they could alter the decision process by providing malicious attribute values. Existing services, such as LDAP services, SAML Attribute Authorities, or Attribute Data Bases can be exploited as AMs. *Policy Information Points (PIPs)* act as plugins of the Access Control System, providing the interfaces for interacting with each AM, thus allowing the Access Control System to retrieve the latest values of attributes and to update them.

2.2 Blockchain Technology

Blockchain technology allows the distributed creation and management of an unique data repository (the ledger), thus maintaining a common state among different untrusted parties. The state of a blockchain can be as simple as a collection of records, as in traditional cryptocurrencies such as Bitcoin [3], or as complex as a virtual machine representation, as in smart contract enabling blockchains such as Ethereum [4]. The state updates are recorder through data structures named *transactions* that are grouped into *blocks* cryptographically linked to each other in a ordered list, i.e., a *chain*. To update the state means to add new blocks, and so new transactions, to the chain. Such process is performed by mutually untrusted byzantine entities, and so a distributed consensus algorithm is employed. The consensus algorithm guarantees, under certain conditions (often on the percentage of honest participant) that the chain keeps growing with honest majority in control [5]. Coupling the distributed consensus with a P2P communication network that anyone can join to listen or submit information and chain replication among the peers provides *decentralisation*. The fact that transactions once written in the chain can not be changed (unless the distributed consensus algorithm is overcome by malicious entities), guarantees *persistency* (information remains publicly visible), *timestamping* (information exists at a given discrete time) and *immutability* (information can not be changed). The previous three properties altogether provide *auditability*, i.e., proof that a given information do exist at a given time and cannot be changed later.

Transactions submitted to be included in new blocks are first validated by the consensus participants. Depending on the blockchain protocol, the transaction payload can be a simple record update, or an executable code. In the latter case the code is executed by the validators to update the common state accordingly. Such blockchain protocols are called *smart contract supporting* protocols. A *smart contract* can be, in general, any kind of executable code. The code can be stored on chain organized into callable functions of programmable objects, named *contracts*. Transactions can either create new contracts or invoke functions of existing ones, specifying the possible parameters. The execution of such calls is replicated among all nodes taking part in the consensus and so, the call results are beyond the control (and can not be tampered with) by the transaction creator. It is the contracts' code that deterministically determines the outcome. A reward mechanism for the validation effort is often enforced by blockchain protocols, e.g., the smart contract supporting Ethereum protocols uses the concept of *gas* to price contracts execution. Each basic operation has a gas cost, and the total gas of a given code execution is calculated as sum of all its basic operations costs. Transaction creators can then specify a voluntary *gasprice* to advertise how much they are offering for each unit of gas spent. The transaction gas cost times the advertised gas price is the total amount spent (in the blockchain backed cryptocurrency) to have the transaction validated, i.e., inserted in a new block.

3 Related Work

This work follows the same line of our previous works [6, 7] whose main contribution is the idea of merging a traditional access control system with blockchain technology. Such generality is unique in the current literature. A preliminary attempt [8] only used the blockchain as secure repository and log for policies management. In [9], instead, the entire decision process is delegated to the blockchain. The current work starts from the insight obtained by such previous work, as well as the intuition that a fully blockchain integrated system might not be desirable for some scenarios. As such we show how integrating AMs only on chain would still provide interesting properties.

Research on blockchain solutions for Access Control systems have mainly been proposed in the IoT field [10–12]. In [13] the authors propose an entire blockchain protocol with modified transactions dedicated only to the Access Control management. Differently, in our proposal we leverage an already existing, and, potentially, well established and secured, blockchain, without any change needed. Another interesting proposal is [14] where the blockchain is used to distribute the access control process, but there is no traditional access policy involved. Instead a single global policy is used and adapted through machine learning to allow the policy to dynamically self adjust. The proposal uses a novel concept of policy, differently from our aim of disrupting traditional Access Control systems as little as possible. Also no experimental evaluation is provided in [14] to evaluate the actual feasibility of the proposal.

Another research field that has sparked a lot of interest is to try and apply blockchain technology to Access Control systems in health care [15–17]. Such proposals revolve around the implementation of an Electronic Medical Record, and the advantages of security, availability and interoperability that blockchain integration could provide. In such a sensitive field, the potential privacy issues introduced by blockchain use need to be properly addressed. The main difference of such systems from our proposed one is their lack of generality. We provide a possible integration with blockchain technology of any traditional Access Control system independently of their application scenario. Our proposal is not tailored for IoT or health care applications, it can be applied wherever a traditional system already is.

4 Blockchain-based Attribute Managers

This paper proposes to enhance traditional XACML based Access Control systems by integrating smart contract managed Attribute Managers (called SMART AMs). SMART AMs are smart contracts able to store, compute, and manage attributes values, and they are invoked by Access Control systems to retrieve the current values of such attributes when required for evaluating access control policies. The integration of SMART AMs within XACML based Access Control systems is straightforward, since the XACML reference architecture has been designed to be modular and to allow the integration of any kind of AM. In other words, this paper proposes to use a traditional XACML Access Control system, where the PEP, the CH, the PDP, the PAP, and a set of AMs are deployed off chain (e.g., on the machine related to the resource to be protected, or on a VM running on the Cloud), and to extend this system by introducing a set of blockchain-based AMs, which will be enabled to interact with the rest of the

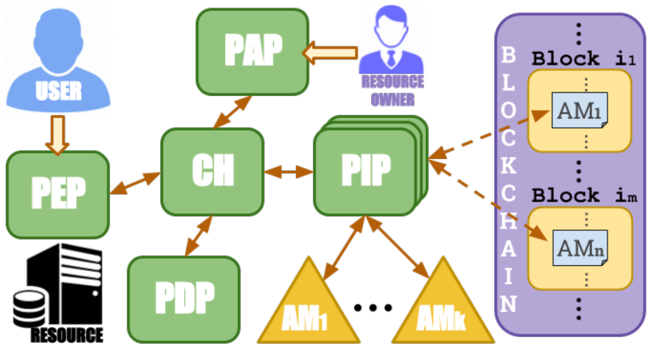


Fig. 2. Architecture of the proposed system.

Access Control system through the development of a proper set of PIPs (as required by the XACML standard).

With respect to our previous solution described in Section 3 which, instead, moves the entire Access Control system on the blockchain, the approach proposed in this paper allows for the minimal disruption while still retaining the blockchain benefits, even if only applied to the management of attributes, instead of to the whole access decision process. As a matter of fact, this solution also allows an Access Control system to exploit traditional AMs and SMART AMs at the same time, thus being applicable to real application scenarios where some of the attributes required for the policy evaluation are already available through existing traditional AMs.

4.1 Integration in the XACML Reference Architecture

This section shows how SMART AMs can be integrated within a traditional XACML Access Control System to retrieve the attribute values required for the decision process.

First, we note that the XACML reference architecture (see Section 2.1) is already designed to integrate external AMs in the decision process. In fact, in several real application scenarios, AMs are usually third party services (e.g., attribute Data Bases, LDAP services or SAML Attribute Authorities). The way for integrating SMART AMs in XACML Access Control systems is through PIPs, the pluggable components for interacting with AMs. Hence, we only need to define new PIPs capable of interacting with the blockchain in order to invoke smart contracts, and to integrate such PIPs with the CH which will invoke them when necessary to carry on the decision process, as shown in Figure 2. Obviously, an Access Control system can exploit traditional AMs and SMART AMs at the same time, simply integrating a set of proper PIPs.

We imagine an already existent ecosystem of SMART AMs advertised by their respective owners. We assume the owner lets the potential users know at least the contract address and the mean to retrieve attribute values from it, e.g., they could advertise the signature of its public methods returning values for certain attributes. Of course it is in the interest of the owner to make those information reachable to potential users (customers in case of pay to use SMART AMs). Furthermore, the owner can optionally also advertise the source code of the contract to increase its transparency, if this is a property required.

4.2 Implementation Details

To validate our proposal, we have developed a proof of concept implementation based on the WSO2 OASIS `balana`[4] framework, acting as Policy Decision Point, and on a PIP written in Java, same as the other components of the XACML based Access Control system.

The PIP has been integrated in the `balana` based Access Control system by extending the *Functionbase* `balana` class[5]. In particular, the *evaluate* function of the extended class is the one which includes the code for accessing the blockchain. Assuming that our PIP wants to retrieve the current value of an attribute, say *a*, from a SMART AM, say *am*, it only needs to call the corresponding function of the smart contract *am*. This means sending a transaction to the blockchain (that can result in a simple read of a value from its state). The PIP then uses the `web3j` library [18] to send the request to a `geth` client that will use it again to send back the answer to the PIP.

It is worth noting that the basic function of an AM, returning the attribute value, would be natural to be implemented as a *constant* function (also known as *pure* function in Ethereum), i.e., a function that does not cause any state update. This may not hold in all scenarios (e.g. when the request for an attribute causes some logging inside the attribute manager, for example for statistical purposes), but in general a fetch request of some value, should not modify an AM state. Furthermore it is not true if some pricing of value read is in place, since the micro-payment exchange needs to be recorded. In general having constant functions only, leads to huge advantages. In fact, no transaction needs to be sent to the network, since the user's client can execute the contract function code locally (it has no need to notify the other nodes since the result does not affect the common state). No transactions validation means no fees to be paid and no wait for blocks to be mined. Do note that the relevant contract function code is still executed and it can still be arbitrary complex, not necessarily just "reading a value".

5 Considerations

This section discusses the main advantages and drawbacks introduced by using SMART AMs instead of traditional AMs.

[4]http://xacmlinfo.org/category/balana/

[5]https://github.com/wso2/balana/blob/master/modules/balana-core/src/main/java/org/wso2/balana/cond/FunctionBase.java

5.1 Transparency

In case of public blockchain, the adoption of SMART AMs allows increased transparency since the attribute values cannot be tampered by the AM owner. In practice, using a public blockchain to manage AMs has the effect of making attribute values and their updates publicly visible (unless an obfuscation layer is purposely introduced). This allows auditability, in the sense that any given attribute is proven to have had a certain value in any point in time. Moreover, the actual attribute value fetch is executed in a distributed fashion, so the AM owner can not intervene or tamper with the process. The AM owners do still retain the ability to update their controlled attribute values, but such updates' history is remembered by the chain, exposing any fraudulent behaviours. The other side of the coin is of course a lack of privacy. All values publicly visible are readable by all users. AMs managing sensible data can not follow this simple approach.

5.2 Data Replication

In a blockchain, data are replicated among all the participants. This has the clear advantage that AMs information is locally retrievable by querying an updated local copy of the blockchain (as noted in Section 4.2). Hence, SMART AMs guarantee availability by removing the problem of single point of failure and attack. The obvious drawback is that more resources (computational power, memory, bandwidth and storage) need to be dedicated at maintaining the system. Such burden can, however, be eased, for the client, by employing a *light client*. A light client is a minimalistic client that does not locally store the entire chain, but instead relies on other nodes to know of the information it requires. The advantage is a dramatic reduction in the resources needed at the expense of a communication delay (since the required information needs to be fetched by remote nodes, see Section 6) and possible privacy leaks (since the queried nodes can infer private information by knowing what the client is interested in).

5.3 Smart Contracts

Another novel advantage of using SMART AMs instead of traditional AMs concerns the automatization capabilities granted by smart contract. Due to the self executing nature of smart contracts, entirely new scenarios are possible. For example, a SMART AM could be defined to automatically update its attribute values depending on some events happening on chain. In a traditional system, users need to trust the managers of the AMs they are retrieving attribute values from. Using an automatic SMART AM would instead require users to only inspect and trust the code that defines it, relieving any need for trusted third parties. The novel ability of creating trust between byzantine entities is a main contribution of blockchain technology, and it could be leveraged in the Access Control field by making trustworthy AMs possible.

6 Experimental Results

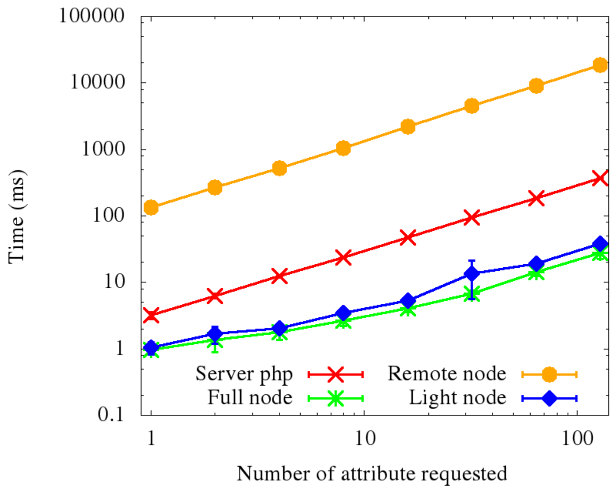
In order to validated the proposed approach, we performed a set of experiments with our proof of concept implementation. The goal of the experiments is to measure the time required to evaluate XACML policies exploiting SMART AMs, comparing this time with the one required employing a traditional Attribute Manager, i.e., a database service. We do remark that the proof of concept modifies a traditional Access Control system only to allow it to also recover attribute values from Ethereum smart contracts. So the same general system is used for both tests, the only thing we change is the policy considered, i.e., whether it contains references to traditional AMs or to SMART AMs.

For the experiments involving traditional AMs we used a PHP web server operating a MySQL DataBase. Do note that the `balana` framework and the PHP web server are deployed on distinct machines. The PIP for interacting with this AM simply requests the required attribute via a http get operation.

For the SMART AM tests we considered three different options. First of all, we performed our experiments on an official Ethereum testnet, i.e., *Rinkeby* [19], not to bloat the main net with temporary test data. On such network we deployed a few hundred contracts representing each a different SMART AM. For this experiment we chose to keep the SMART AMs as simple as possible, and so they only advertise two methods without parameters, to return a boolean and an integer respectively, and a single method accepting one parameter, representing an user name, and returning the value of an integer attribute associated to that user (remembered by a map inside the contract). All of the methods are constant (see Section 4.2).

Rinkeby is one of the official global test nets used by the Ethereum community. To connect to such network we need a client to receive and send messages (including transactions) as well as read the data on the blockchain. Different type of clients are possible. For the first blockchain test we decided to use the less invasive solution possible. As such we relied on a third party node, i.e., a remote node with respect to the one where the Access Control system is deployed, that we connect to through `http` requests. We chose *Infura* [20], the most widespread such service at the time of writing. The advantage of relieving the user from any need to deploy and maintain a local blockchain node, provided by such approach, is paid with the time spent waiting for `http` requests. For the second and third tests we installed, respectively, a blockchain light node and full node (see Section 5) with which we interact through the `geth` client installed on the same machine as the `balana` framework.

Figure 3 compares the results of our experiments. We measured the overall time to fetch the attribute values needed to evaluate XACML policies requiring to collect the values of {1, 2, 4, 8, 16, 32, 64, 128} attributes in each of the four scenarios detailed above. For each experiment we perform one hundred executions and we show the mean and standard deviation. We note that when a light node needs a given smart contract code, it has to request it to other nodes but, once collected, it stores it locally (*caching*) and it has no need to query it again until it is updated on chain. As such, by repeating the same attribute



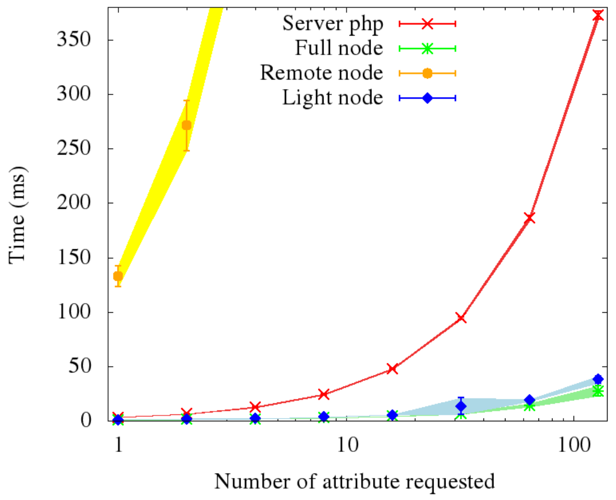


Fig. 3. Time required to fetch attributes with SMART AMs and traditional AMs. The results are shown for a traditional AM implemented as a PHP server operating a MySQL DataBase, a remote *Infura* node, a `geth` light node and a `geth` full node. The upper graph represents the complete results in logarithmic scale, while the bottom graph shows a zoom in linear scale, focusing the scale on the three fastest methods only (PHP server, full node and light node).

request many times, the client would actually use the local value instead of requesting it again. This results in very efficient information collection, as long as the attribute values considered are fairly static. In fact such a solution has the advantage that only the information actually needed is stored locally (and updated fairly rarely), while a full node would have to store the entire blockchain and keep it updated all the time. To test the effectiveness of light node caching we performed a worst case experiment, i.e., we prevented caching. We deployed one hundred SMART AMs, each in a different block, to avoid possible block caching. We then updated a given attribute value once for SMART AM, making sure that each update transaction would end up in a different block. For each of those SMART AMs we defined a policy requiring only the attribute value which has been previously updated. The resulting average wait time to evaluate all hundred policies with only one attribute value (one from every different SMART AM) is 227.25 ms, about one hundred times higher that the time needed to evaluate a policy only using one cached attribute value, i.e., 2.7625 ms. So a light node is an highly efficient solution, but mainly in the case of several requests to the same, fairly static, Attribute Providers. Of course, maintaining a local full node is the better solution in terms of time performance (since all the blockchain data is kept locally by the node), but it is costly in terms of resources needed to be maintained (see Section 5).

7 Conclusions & Future Work

In this paper we have presented an approach to exploit blockchain based Attribute Managers in traditional Access Control systems. A relevant advantage of this approach is that it is not disruptive, i.e., it can be adopted in those scenarios where traditional Attribute Based Access Control systems are already in place, by simply integrating new Policy Information Points.

As future work, we are planning to compare the performance of blockchain based Attribute Managers with other services typically exploited as Attribute Managers, such as LDAP services, and to extend the adoption of blockchain based Attribute Managers to more complex Access Control systems, such as the one implementing the Usage Control model [21], where the role of attributes is crucial because they change their values as normal consequence of the system operation, thus requiring the repeated evaluation of the Access Control policies while an access is in progress.

References

1. Vincent C. Hu, David, F., Rick, K., Adam, S., Sandlin, K. Robert, M., Karen, S.: Guide to attribute based access control (ABAC) definition and considerations (2014)
2. OASIS: eXtensible Access Control Markup Language (XACML) version 3.0 (January 2013)
3. Nakamoto, S.: Bitcoin: A peer-to-peer electronic cash system (2008)
4. Wood, G.: Ethereum: A secure decentralised generalised transaction ledger. Ethereum Project Yellow Paper **151** (2014)
5. Miller, A., LaViola Jr, J.J.: Anonymous byzantine consensus from moderately-hard puzzles: A model for bitcoin. Available on line: http://nakamotoinstitute.org/research/anonymous-byzantine-consensus (2014)

6. Di Francesco Maesa, D., Mori, P., Ricci, L.: Blockchain based access control services. In: IEEE International Symposium on Recent Advances on Blockchain and Its Applications (BlockchainApp), 2018 IEEE International Conference on Blockchain, IEEE (2018) 1379–1386
7. Di Francesco Maesa, D., Ricci, L., Mori, P.: Distributed access control through blockchain technology. Blockchain Engineering (2017) 31
8. Di Francesco Maesa, D., Mori, P., Ricci, L.: Blockchain based access control. In: IFIP International Conference on Distributed Applications and Interoperable Systems, Springer (2017) 206–220
9. Di Francesco Maesa, D., Mori, P., Ricci, L.: A blockchain based approach for the definition of auditable access control systems. Computers & Security **84** (2019) 93–119
10. Novo, O.: Blockchain meets IoT: An architecture for scalable access management in IoT. IEEE Internet of Things Journal **5**(2) (2018) 1184–1195
11. 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, ACM (2018) 61–69
12. Tapas, N., Merlino, G., Longo, F.: Blockchain-based IoT-cloud authorization and delegation. In: 2018 IEEE International Conference on Smart Computing (SMARTCOMP), IEEE (2018) 411–416
13. Ouaddah, A., Abou Elkalam, A., Ait Ouahman, A.: Fairaccess: a new blockchain-based access control framework for the internet of things. Security and Communication Networks **9**(18) (2016) 5943–5964
14. 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) (2017) 417–424
15. Azaria, A., Ekblaw, A., Vieira, T., Lippman, A.: Medrec: Using blockchain for medical data access and permission management. In: Open and Big Data (OBD), International Conference on, IEEE (2016) 25–30
16. Dias, J.P., Reis, L., Ferreira, H.S., Martins, Â.: Blockchain for access control in e-health scenarios. arXiv preprint arXiv:1805.12267 (2018)
17. 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. Sustainable cities and society **39** (2018) 283–297
18. Svenson, C.: Blockchain: Using cryptocurrency with Java. Java Magazine, January/February (2017) 36–46
19. Rinkeby Ethereum Testnet. `https://github.com/ethereum/EIPs/issues/225` Retrieved 15 Feb 2019
20. Infura - Scalable Blockchain Infrastructure. `https://infura.io/` Retrieved 15 Feb 2019
21. Carniani, E., D'Arenzo, D., Lazouski, A., Martinelli, F., Mori, P.: Usage control on cloud systems. Future Generation Comp. Syst. **63**(C) (2016) 37–55