Smart-contract based Access Control on Distributed Information in a Smart-City scenario

Francesco Buccafurri, Cecilia Labrini, and Lorenzo Musarella

University Mediterranea of Reggio Calabria, Italy
bucca,cecilia.labrini,lorenzo.musarella@unirc.it

Abstract

In the smart-city paradigm, data sharing is one of the pillars needed for its full implementation. Among the other aspects, we refer to the opportunity for users (citizens, companies, organizations) of exploiting data sources managed both by institutional parties and third parties involved in the smart-city life. Open-data is an answer to above need, but, sometimes data cannot be disclosed publicly, coming to the concept of *closed data*. In this case, access control takes a fundamental role. The problem is not trivial, since we deal with a highly open and dynamic environment, and, at the same time, that a certain level of accountability should be guaranteed to contrast misbehaviour and solve possible legal controversies. In this paper, we propose a solution based on the combination of Ethereum smart contracts, eIDAS-based attribute and identity management, and the distributed file system IPFS.

1 Introduction

In these years we are spectators of a fast and incredible technology revolution that involves everything that surrounds us, from mobile devices to cars with autonomous driving, from the development of smart grids to new communication protocols. In this new world, there is a need of a new vision of what cities are and what cities should provide to populations. One of the features that smart cities should provide is the easy yet secure access to data which represent the substrate of the smart-city life. Although a lot of attention has been devoted to interoperability and open-data [9], larger space of investigation exists in the domain of *closed-data*, regarding different aspects, ranging from the way data sharing is implemented, to how the access is controlled, and how to assign responsibilities to the different involved parties, with the aim to make the access effective but accountable. This paper tries to give a concrete solution leveraging the power of Ethereum smart contracts, the Interplanetary File System (IPFS) and the eIDAS European Regulation Ecosystem [26]. The idea is to implement on top of the above components, an Attribute-Based Access Control mechanism (ABAC). As a matter of fact, ABAC represents the emerging approach for large environments. Gartner predicts that by 2020, 70% of organizations worldwide will have moved to the ABAC model. However, one of the main issues to deal with for ABACs is how to assess attributes in a feasible way. In this paper, we propose an approach in which institutional bodies are responsible for attribute certification, according to the eIDAS paradigm of *Attribute* and *Identity Providers*, and access control enforcement is done in a trust way, thanks to Ethereum smart contracts.

The structure of the paper is the following. In Section 2 we introduce some basic and required concepts. In Section 3, the scenario and the motivations that drove us into facing this

problem are described. Then, in Section 4, we explain the architecture of our proposal and the actors involved in it. We briefly analyze, in Section 5, the most important security aspects and properties of our solution. In Section 6, some features of the implementation phase of the proposal are described. In Section 7 we survey the related work. Finally, in Section 8 we draw the conclusions and the we sketch future works.

2 Background

In this section we present some background concepts to better introduce, later, our proposal.

1. *Ethereum*

   Ethereum is a public blockchain-based platform that allows the development of decentralized applications (Dapps) able to interact with each other in a secure and fast way [13, 29]. Ethereum provides a decentralized and Turing-complete virtual machine able to execute scripts and code. In Ethereum, there are two different kinds of accounts: the *Externally Owned Account* (EOA) and the *Contract Account* (better known as Smart Contract). The former is controlled by a private key (like an account of the Bitcoin blockchain), while the latter is controlled by its code, which is executed by every peer of the blockchain so that the output of the execution is well-known to them. Moreover, a smart contract can be written easily thanks to Solidity, a high-level programming language that implements the Ethereum Virtual Machine bytecode [11].

   Smart contracts are used also as agreements between users that do not trust or do not know each other and, for these reasons, they definitely represent the *killer feature* of Ethereum. Furthermore, everything that is on a smart contract is permanently stored in the Ethereum blockchain to guarantee the traceability of actions.

2. *Attribute-Based Access Control (ABAC)*

   Access control is one of the hottest and trend topics of the last decades in the IT world. Indeed, to guarantee a right, secure and non-invasive mechanism that enable or not the access to the resource is always necessary. In particular, an access control mechanism should satisfy, among the others, the following properties: it does not have to allow non-authorized people to access the resource and it does not have to deny authorized people not to access the resource. The type of access must be subject of authorizations as well.

   There exist many families of access control, such as *Mandatory Access Control* and *Discretional Access Control*, for what concerns the flexibility of authorization rules, *Role-Based Access Control*, *Context-Based Access Control*, *Attribute-Based Access Control*, etc., for what concerns the way to associate authorizations to subjects.

   In particular, Attribute-Based Access Control (ABAC) is defined as an access control mechanism in which authorization is computed evaluating the fulfillment of one (or more) required attribute. The National Institute of Standards and Technology (NIST) defines ABAC as follow [17]: *"An access control method where subject requests to perform operations on objects are granted or denied based on assigned attributes of the subject, assigned attributes of the object, environment conditions, and a set of policies that are specified in terms of those attributes and conditions."*

   As a matter of facts, after some decades of supremacy Role-Based Access Control (RBAC), ABAC is emerging as definitely more suitable than RBAC to large environments, in which

defined roles must be set by the management and associated with people, resulting in what is known as *role explosion*. In our context (i.e., smart city), ABAC is the elective choice.

3. *Smart City*

   Nowadays, cities face different problems and challenges to improve their citizens' quality of life [25]. Governments, communities, and businesses increasingly rely on technology to overcome the problems that arise daily [28]. Smart cities can make an intelligent response to different kinds of needs, including public safety and services, industrial and commercial activities, transportation, and healthcare [24]. In detail, a city becomes a smart city when it combines the usage of network infrastructure, software systems, server infrastructure, and client devices to better connect critical city infrastructure components and services. Smart cities are an effective integration of smart planning ideas, smart development approaches, and smart management methods. On the other hand, a city cannot be defined as smart if it adopts limited and sectorial improvements.

   Indeed, a smart city must involve different elements such as smart governance, smart economy, smart mobility, etc. Smart cities make use of new types of information and communications technology to support common sharing which is one of their most important characteristics. It is well-known that the features of blockchain technology may contribute to smart city development through sharing services.

4. *InterPlanetary File System (IPFS)*

   IPFS is a peer-to-peer distributed file system that connects all computing devices with the same system of files [27]. It provides a high-throughput content-addressed block storage model, with content-addressed hyperlinks. IPFS employs content-addressing to uniquely identify each file in a global namespace connecting all devices. Furthermore, it identifies, verifies and transfers files relying on the cryptographic hashes of their contents [22]. It also integrates technologies such as self-certifying namespaces, an incentivized block exchange, and distributed hash tables (DHT), etc. IPFS is built around a decentralized system of user-operators who own a portion of the total data, creating a resilient system of file storage and sharing. As a consequence of this decentralized approach, IPFS has not a unique point of failure, and nodes do not need to trust each other as well.

   The IPFS user, when uploads a file to the system, will have back a unique cryptographic-hash string (IPFS-identifier of the document) through which she/he can retrieve the file every time and everywhere. Indeed, it is not required that the user stores the original file in her/his devices, but it is enough to know the IPFS-identifier of the document to obtain it. The hash string can be seen as a Uniform Resource Locator (URL) of the World-Wide-Web.

3 Scenario and Motivation

In the widest interpretation of the concept of a smart city, one of the main challenges is to guarantee a secure, trusted and fast data sharing. This may have a significant impact both in open-government policies that are crucial in smart cities [30], and in the smart fruition of information to deliver complex services that need multiple data sources.

In this scenario, it is fundamental to implement an access control mechanism that is able to decide who can read what, by taking into account the fact that we operate in an open environment, in which interested subjects cannot be predetermined. It is worth noting that,

although smart cities should provide *open data*, which are accessible with no limitation, also *closed data* are relevant for the full implementation of smart-knowledge-based communities. Therefore, access control becomes necessary.

To better explain, consider the following example. Suppose that some closed data are produced by a smart city entity like a hospital or a court. Since we are talking about sensitive data, it is reasonable to think of some policies for which only people belonging to the medical board (in the case of healthcare data) or lawyers (in the case of law data) can access.

As seen above, there are many access control models but, among all, due to the open nature of our scenario, Attribute-Based Access Control (ABAC) seems the most suitable one because neither identities nor roles are able to capture all the conditions which should be satisfied by subjects to access information. Moreover, ABAC allows us to implement anonymous-credential mechanisms to avoid that sensitive data of subjects are disclosed to possibly untrusted parties and, thus, to preserve privacy.

Obviously, if we think about a smart city and its data it is clear that it is necessary to ensure properties like accountability, privacy, trust and non-repudiability (among the others). In this sense, Ethereum blockchain perfectly fits with these requirements. Another motivation that led us to choose Ethereum is that the mechanism of verification of attributes must be trusted. Furthermore, the usage of an Ethereum smart contract enforces the attribute-based access control mechanism without any privacy leakage, since attribute-based authorizations are anonymous and prevent any disclosure of personal, sensitive, and not required information.

Anyway, Ethereum, as every blockchain, is not the most suitable platform for sharing and storing large files since the blockchain is replicated on many nodes and a lot of storage space is required without serving an immediate purpose [22]. Moreover, the blockchain becomes bloated with data that has to be propagated within the network and the price of operating blockchain nodes increases because more data needs to be processed, transferred and stored. File sharing platforms can be leveraged to solve these problems. Users can easily share large files and still benefit from the blockchain.

InterPlanetary File System (IPFS) is a particularly interesting protocol peer to peer file sharing platform that combines file sharing and hashes. Cryptographic hashes serve to securely identify a file's content. IPFS makes it possible to store and share large files more efficiently and it is based on cryptographic hashes that can easily be stored on a blockchain. Unlike existing cloud storage, IPFS has the advantage that data is distributed and stored in different parts of the world and not on a central server [27]. Finally, a solution exploiting IPFS guarantees data availability.

4 Our Proposal

In this section, we propose our solution regarding the scenario discussed above.

First, we define all actors involved in it, then we present all the steps needed to reach our goal.

Our idea is the following. Suppose that the smart city produces, owns and provides data that it wants to share not to everybody but only to who fulfills some requirements. To be more concrete, but without loss of generality, we describe our solution in the healthcare setting, although it can be easily extended to every typical context of smart cities (e.g., transport, commercial and law data, etc.). In our proposal, we assume that documents stored on IPFS are already encrypted and, for this reason, objects of our access control authorizations are keys instead of final resources.

In our solution, we define the following actors:

- User U, a citizen that asks for data;
- Identity Provider IP, whose task is the management and verification of user identities;
- Attribute Provider AP, whose task is the management and verification of user attributes;
- Access Service Provider ASP;
- Publish Smart Contract PSC, an Ethereum Smart Contract used for the publication of documents on IPFS;
- Access Smart Contract ASC, an Ethereum Smart Contract used for verifying the policy and, where appropriate, for granting the key for the decryption of the document;
- Oracle O, that is used by ASC for checking the validity of the certificate of U;
- Content Manager CM, who is in charge of encrypting documents, publishing them on IPFS, associating them to the right policy and addressing the key-request when U fulfills its requirements;
- IPFS, used for storing data in a distributed way;
- Ethereum, a public blockchain allowing the development of smart contracts.

Once defined all entities involved in our proposal, let's describe the steps of our solution:

1. *Policy Setup*

   In this first phase, the CM associates the document d_i with the policy $\hat{p}$. If $\hat{p}$ does not exists, CM will generate it compliant with the XACML standard.

   In particular, every category has a set of attributes related to and, as a consequence, a different policy. In our scenario, we refer to the category of healthcare data, where, for the sake of presentation, the attribute to be fulfilled is *to be a doctor*.

2. *Encryption*

   CM encrypts d_i with a symmetric encryption function (such AES) by using the key related to the policy associated with healthcare data. Indeed, in accordance with the above, every category (and every policy) has a different encryption key. Let us denote by $\hat{k}$ the key associated with the policy $\hat{p}$. Now, CM obtains the encrypted document e_i.

3. *Publication*

   The goal of this step is to publish on IPFS the encrypted document e_i and the related policy $\hat{p}$. This could be achieved through different ways. Indeed, CM could simply publish it directly with an IPFS client. Anyway, accordingly to the accountability features aimed by our proposal, we allow the publication only trough the Ethereum smart contract.

   In detail, CM calls a function of the PSC through her/his Ethereum address ETH_{CM} with e_i as input obtaining as output the IPFS-hash h_i related to e_i. At the same time, CM calls another function of the same smart contract PSC to publish on IPFS the policy $\hat{p}$, if it still does not exist in the Ethereum environment, thus obtaining the IPFS-hash $h_{\hat{p}}$.

   At this point, CM maps h_i with the corresponding policy $h_{\hat{p}}$ on the PSC. The result is a mapping between the policy $\hat{p}$ and the list of documents associated with. Furthermore, there is a mapping between the area of interest (in this case, healthcare) and $\hat{p}$.

4. *Attribute Verification*

   In this phase, the user U requests to ASC the policy that she/he has to satisfy related to healthcare, obtaining $h_{\hat{p}}$. Now, similarly to the *Publication* phase, U could obtain the document directly with an IPFS-client, but again, our protocol enforces U to get it only via ASC. At this point, U knows $\hat{p}$ and she/he can see that the attribute required is *to be a doctor*. For the assessment of the attributes owned by users, we apply an eIDAS-based approach [26], in which a SAML-2 authentication process is established to involve both an Identity Provider and one or more Attribute Providers which are institutional entities responsible for providing information (like title, licences, qualifications, age, etc.) about digital identities. To be compliant with real-life regulations, we cannot imagine that every user plays the role of the service provider in an eIDAS authentication loop. Therefore, we introduce an intermediate service, called Access Service Provider (ASP), needed to perform the authentication request and to obtain the valid corresponding assertion. In detail:

   - U goes to ASP to request, by playing the role of Service Provider the assertion in which there is the information certifying the attribute *to be a doctor*;
   - through a SAML2-compliant schema, ASP forwards the request to the Identity Provider IP;
   - after that, IP contacts the Attribute Provider AP (in the use case, the medical board) asking for the certification of the attribute *being a doctor*;
   - now, AP sends the reply to IP, which will contact ASP to communicate the information obtained through an assertion. Finally, ASP returns to U the assertion and the nonce related to. In particular, the nonce allows applications to correlate the identifier of the assertion with the initial authentication request and it is used also to avoid replay attacks (see Section 5).

   ETH_U sends a transaction to ASC calling a function in which she/he puts the hashed identifier of the assertion and the nonce as input. The smart contract, now, has to verify the validity of the assertion and, as a consequence, the real possession of the attribute required by the policy $\hat{p}$. To do that, ASC invokes the Oracle O, that is in charge of checking the overall validity of the previous steps with IP.

   If the check succeeds, ASC emits an event in which it confirms the satisfaction of the policy.

5. *Key Granting*

   In this step, CM sees the event of success on the blockchain (it can be done via a client application that is able to show, and possibly filter, events) and CM sends a transaction blockchain to ETH_U with the information about $\hat{k}$. Obviously, before to send the transaction, CM should encrypt the key to prevent from its disclosure to all blockchain. To reach this goal, CM encrypts $\hat{k}$ with U's public key, obtaining $E^U_{pk}(\hat{k})$. The result is an ethereum transaction from CM to ETH_U through the ASC having as `input_data` $E^U_{pk}(\hat{k})$. We remind that `input_data` is an optional field of an ethereum transaction that can be used to share other information.

   Finally, U is the only one who can decrypt, with her/his private key, the chipered key $E^U_{pk}(\hat{k})$. After the derivation of $\hat{k}$, U is able to dechiper also e_i, thus obtaining d_i.

5 Security Aspects

In this section, we briefly analyze the main security aspects that are involved in our proposal. This paper, indeed, can be view as a position paper presenting some results obtained in an industrial research project still alive. A more detailed analysis is then forwarded to the next steps of our work.

Let us begin with the definition of the *adversarial model*, that is particularly relevant for our proposal because by modeling the role of attackers, with their capabilities and goals, we could help to improve the cyber defense [12] and, since we are facing the case of closed data in the smart city scenario, it is necessary to ensure that some fundamental security evaluations are valid. In detail, in our proposal, we assumed that the Content Manager, the Identity Provider and the Attribute Provider are trusted parties and that the attacker can be either a user or the Access Service Provider.

In particular, the Access Service Provider could operate in a malicious way since it could give the wrong assertion to the wrong users. Anyway, this attack is contrasted by using the SAML-2 standard because the *Authentication Request* and the *Authentication Response* must coincide, so the ASP is not able to give the wrong response to the wrong applicant.

In sum, we do not require the ASP more trust then that one required by the eIDAS regulation. In addition to the standard security properties and assumptions, such as those ones related to the Ethereum blockchain and IPFS protocols, we assume that user's secret information and data are not disclosed to the public world and that users do not collude each other as well.

The goal of the attacker is to break, at least, one of the following secure properties: availability, non-repudiation, accountability, integrity and confidentiality.

In our proposal, data are stored on IPFS while their identifiers are stored on the Ethereum blockchain. This combination of these two technologies contrasts attacks on availability. Indeed, the usage of IPFS avoids the central and unique point of failure, since data are duplicated on multiple and random IPFS peers. Moreover, the DoS attack, in which the attacker floods the Ethereum network with a huge amount of requests, would be very expensive because every Ethereum transaction and every call to a function of an Ethereum smart contract has a cost in terms of gas.

Non-repudiation is obtained. In fact, every action is logged into Ethereum and it can be verified at any time and, in addition, Ethereum transactions are not editable after been mined. They are also signed by the Ethereum private key, that is known and kept only by the owner of the address. Furthermore, non-repudiation is ensured by our protocol also during the phases involving the publication of documents and policies on IPFS and the downloading of such files from IPFS. In particular, although these operations could be carried out by using a standard IPFS client application, we implemented an alternative approach, based on smart contracts, enforcing the non-repudiability of the overall protocol.

We can distinguish two different domains of interest: that one on the blockchain and the other one off-chain. Concerning the former, accountability is ensured similarly to the non-repudiation property. Instead, if we think to the off-chain side of our proposal, accountability is reached because only the Identity Provider and the Attribute Provider(s) know exactly the link between the identity of the user and its related Ethereum address. So, if for any reason it is necessary to reveal this mapping, it could be done by merging information from different parties.

Data integrity is, again, reached thanks to IPFS by the usage of the IPFS-cryptographic-hash that is carried out for every document published on the InterPlanetary File System.

Confidentiality is obtained as well, because sensitive and *closed* data are chipered by the Content Manager and the decryption key is given only to those users that fulfill the policy associated with them. Furthermore, the Content Manager, before sending to the user the key, encrypts it with the ethereum public key of herself/himself, that can be easily derived from the Ethereum address. So, even if the Content Manager sends the key by using Ethereum, nobody can actually understand it excepts for the interested user.

Finally, our protocol reaches the goal of privacy requirement, since the Content Manager is not aware of personal and sensitive information about users except for their attributes and their Ethereum addresses. In this case, the level of protection of these data is that one related to the pseudonymity provided by the Ethereum blockchain itself, that is not full. However, if the user wants to preserve better her/his privacy, she/he can generate a new Ethereum wallet for every operation, making attacks on pseudonymity not realizable anymore.

6 Implementation Issues

To implement our proposal, we used many different technologies and framework to integrate IPFS and XACML with ethereum smart contracts.

In particular, these are, among the others, the most relevant ones:

- *RemixIDE* [3], an online IDE for the development of ethereum smart contracts;
- *Truffle Suite* [4], a suite of tools useful for interfacing smart contracts (e.g., Ganache);
- *Metamask* [1], a browser extension that allows us to run dApps on the browser without running a full Ethereum node;
- *Web3.js* [5], a lightweight JavaScript library for integration with Ethereum clients;
- *Provable* (ex Oraclize) [2], the most known oracle used for Ethereum.

For the sake of presentation, we show only the most interesting details we faced during the implementation of our solution and we miss some other details.

First, we developed a JavaScript web-page containing a form allowing the submission of files on IPFS that interfaces with the smart contract showed in Figure 1 in a transparent way.

```
pragma solidity 0.5.8;

contract SimpleStorage {
    string ipfsHash;

    function set(string memory x) public{
        ipfsHash = x;
    }

    function get() public view returns (string memory){
        return ipfsHash;
    }
}
```

Figure 1: Code of the smart contract `SimpleStorage` used to publish on IPFS

Moreover, the web-page returns the IPFS-cryptographic-hash identifier of the document submitted and all this operation is permanently stored in Ethereum. This is done thanks to the JavaScript code shown in Figure 2.

```
this.state.web3.eth.getAccounts((error, accounts) => {
  simpleStorage.deployed().then((instance) => {
    this.simpleStorageInstance = instance
    this.setState ({ account: accounts[0]})
    return this.simpleStorageInstance.get.call(accounts[0])
  }).then((ipfsHash) => {
    return this.setState({ ipfsHash })
  })
})

captureFile(event) {
  event.preventDefault()
  const file = event.target.files[0]
  const reader = new window.FileReader()
  reader.readAsArrayBuffer(file)
  reader.onloadend = () => {
    this.setState({ buffer: Buffer(reader.result)})
    console.log('buffer', this.state.buffer)
  }
}

onSubmit(event) {
  event.preventDefault()
  ipfs.files.add(this.state.buffer, (error, result) => {
    if(error){
      console.error(error)
      return
    }
    this.setState({ipfsHash: result[0].hash})
    console.log('ipfsHash', this.state.ipfsHash)
  })
}
```

Figure 2: Portion of the code of App.js

In particular, after the connection with the library web3js, `captureFile` is the function to buffering the file once submitted on IPFS and `onSubmit` is used for the acquisition of the hash computed by IPFS. The operation carried out for the submission of the policy on IPFS via smart contract are the same. In Figure 3 there is the portion of the smart contract that is in charge of mapping the IPFS-hash of the document and its related IPFS-hash policy.

```
address content_manager;
mapping(string => File) public fileMap;
string[] public filePolicy;

constructor () public {
    content_manager = msg.sender;
  }

modifier onlyCM(){
    require (content_manager == msg.sender);
     _;
  }

function createMapping(string memory ipfsHash, string memory policyHash) public onlyCM
    {
    fileMap[ipfsHash].ipfsHash = policyHash;
    fileMap[ipfsHash].policyHash = policyHash;
    filePolicy.push(ipfsHash);
}
```

Figure 3: Code of the smart contract for the mapping between document and its policy

It is worth noting that the function `createMapping` can be called only by the Content Manager (in this case, the deployer of the smart contract) thanks to the modifier `onlyCM`. Indeed, the modifier is a function of Solidity that, when it is added to the declaration of a function, limits the access to the function itself to those users who satisfy its requests.

Another interesting aspect about the implementation regards the request of the policy from the Ethereum smart contract to the IPFS network because the policy is written with XACML, an XML-like language and there is need to parse the result. This has been solved as shown in Figure 4.

```
pragma solidity 0.5.8;

import "./Oraclize.sol";

contract Policy is usingOraclize{
    bytes32 public oraclizeID;
    string public results;
    event LogOraclizeResult(string result);

    function ipfs() payable{
        OAR = OraclizeAddrResolverI ( "address" );
    }

    function getAttribute() payable {
        oraclize_query("URL", "xml(https://ipfs.io/ipfs/"IPFS-hash identifier").
            AttributeValue");
    }

    function __callback(bytes32 myid, string result) {
        results = result;
        LogOraclizeResult(result);
    }
}
```

Figure 4: Code of the smart contract query with parser XML

Now that the smart contract has obtained the attribute (or attributes) required by the policy, it compares it to the attribute that is in the certificate presented by the user to the same smart contract. If the information completely overlaps, then the Content Manager generates a transaction to the user in which she/he specifies the key associated with the policy after encrypting it with the user's ethereum public key.

7 Related Work

In this section, we investigate the state of the art regarding data access control on distributed information in a smart city environment.

For smart cities, cloud computing has become an important infrastructure as it can provide secure and reliable data storage and sharing. However, in the cloud storage system, the cloud server cannot be considered completely reliable. Therefore, several studies have focused on access control for smart city data using the cloud. In particular, the study [14] proposes a revocable access control scheme of cloud data for smart cities. They design a proxy-assisted access control framework for multi-authority cloud storage system and they construct a new multi-authority Chipertext-Policy Attribute-Based Encryption (CP-ABE) scheme with efficient decryption to realize data access control in the cloud storage system, and design an efficient user and attribute revocation method for it.

In [16] an advanced solution is proposed, which is based on Virtual EnviRonment (CLEVER) enabled for CLoud. The purpose of this proposal is to regulate user access to certain areas and to provide useful data for business intelligence oriented to multipurpose management. In particular, it aims to collect data on people's access and electricity consumption to provide information and services for public, private or governance use. The study [18] presents an Integrated Component for Cloud Services (ISCS) that enables secure and trusted access to data and related services in the cloud. The ISCS controls and handles access-related aspects such as authentication, authorization and registration. It is realized using OAuth and OpenID.

Always in the context of smart cities, several studies were carried out relating to access control and IoT. The IoT protocols must provide data security, in particular, they must guarantee data updating, integrity, confidentiality, authentication, and access control. The access control is perhaps the most important aspect of intelligent cities since the unauthorized access to critical infrastructures can endanger the inhabitants of smart cities (i.g. unauthorized access to traffic lights can cause accidents and traffic congestion and cause huge financial losses). Access control also faces several challenges such as limited resources of IoT devices, often with a limited power budget. Furthermore, access control must provide a high degree of scalability. The study [21] analyzes use cases of smart city and defines requirements of access control for the smart city IoT platform. Attribute-based access control is also analyzed to satisfy the requirements. The requirements of internal access control of smart city IoT platform are analyzed in-depth and an access control mechanism based on information flow history is proposed to control information flow between components of the platform. The most promising work in this area is on Delegated CoAP Authentication and Authorization Framework (DCAF) [15] and Capability Based Access Control (CBAC). CBAC as proposed by Hernández-Ramos et al. solves the scalability issue of access control by decentralizing the validation of permissions, yet the AS remains a single point of failure and a possible bottleneck. The study [8] proposes an efficient format for capacity tokens that is used completely without status and decentralized. This allows deploying access control in scenarios of previous CBAC implementations and DCAF are impossible.

Unfortunately, the data collected and processed by IoT systems are vulnerable to threats of availability, integrity, and privacy. The work [20] takes advantage of the blockchain technology for the protection of privacy and the secure IoT data sharing in smart cities. The blockchain network is divided into various channels to preserve data privacy; each channel includes a finite number of authorized organizations and processes a specific type of data such as health, smart car, smart energy or financial details. Access to users' data is controlled by embedding access control rules into smart contracts and data within a channel is further isolated and protected using private data collection and encryption respectively. A reward system in the form of a digital token is also proposed for users who share their data with interested parties / third parties.

Many studies have focused on the use of blockchain technology as an access control manager for distributed systems. In [19] an approach based on blockchain technology is proposed to publish the policies that express the right to access a resource and to allow the distributed transfer of such right among users. Each user can know the policy associated with a resource and the subjects who currently have the rights to access the resource because the policies and the exchange of rights are publicly visible on the blockchain. This solution allows distributed auditability and a possible working implementation based on XACML policies is also shown. The authors of [23] use a modified version of the InterPlanetary Filesystem (IPFS) that exploits Ethereum's smart contracts to provide file-controlled access to files. IPFS interacts with the smart contract whenever a file is uploaded, downloaded or transferred.

Moreover, the authors of [6] proposed a solution based on Attribute-Based Encryption

(ABE) and the Ethereum blockchain for facing the problem of service delivery with accountability and privacy requirements.

In the paper [10] the authors propose a blockchain-based framework, called Ancile, that allows safe and efficient access to medical records by patients, suppliers, and third parties, while preserving the privacy of patients' sensitive information. Ancile uses smart contracts in an Ethereum-based blockchain for greater access control and obfuscation data, and employs advanced cryptographic techniques for added security. The document shows how blockchain technology can be exploited in the health sector to achieve the delicate balance between privacy and accessibility of electronic health records. In [7], the authors integrate the Ethereum blockchain and the Identity-Based Solution (IBE) by using the Public Digital Identity to overcome the blockchain limitation regarding the fact that the recipient of transactions must be signed up to the blockchain before using it. Indeed, in this work, authors allow transaction between subject not yet registered to the system.

To the best of our knowledge, our proposal is the only one that tries to exploit the advantages of both smart contracts and ABAC into a smart city scenario with distributed information.

8 Conclusions

In this paper, we propose a solution based on Ethereum smart contracts for the access control on distributed information in a smart city scenario. In general, enforcing access control via smart contracts has the drawback that, although in pseudonymous form, the logic of access control is part of the smart-contract storage. Therefore, it is public, and this could be in general a serious threat to privacy of users and organizations. This drawback does not regard our approach, because the scenario we consider is that of access control policies attribute-oriented, of public utility for proper categories of users. Thus we can obtain the advantages in terms of transparency, verifiability, trustworthiness, accountability given by smart contracts. The integration with the eIDAS ecosystem for attribute certification, contributes to make concrete our approach, together with the state-of-the-art technologies used in our solution that are, besides Ethereum, XACML for the implementation of the enforcement into smart contracts and IPFS as distributed file system. As a position paper, this work does not include a careful security analysis. This is planned as future work, together with a full implementation of all the components of the solution.

Acknowledgement

This paper is partially supported by the project "SecureOpenNets-Distributed Ledgers for Secure Open Communities", funded by Ministry of Research and Education (MIUR), project id ARS01_00587. The authors are grateful to Laura Manganaro for her suggestions regarding the implementation aspects of our proposal.

References

[1] Metamask. https://metamask.io (2019)
[2] Provable. https://provable.xyz/ (2019)
[3] Remix - Solidity IDE. https://remix.ethereum.org (2019)
[4] Truffle Suite. https://www.trufflesuite.com/ (2019)
[5] web3js. https://github.com/ethereum/web3.js/ (2019)

[6] Buccafurri, F., De Angelis, V., Lax, G., Musarella, L., Russo, A.: An attribute-based privacy-preserving ethereum solution for service delivery with accountability requirements. In: Proceedings of the 14th International Conference on Availability, Reliability and Security. p. 24. ACM (2019)

[7] Buccafurri, F., Lax, G., Musarella, L., Russo, A.: Ethereum transactions and smart contracts among secure identities. In: DLT@ ITASEC. pp. 5–16 (2019)

[8] Buschsieweke, M., Güneş, M.: Securing critical infrastructure in smart cities: Providing scalable access control for constrained devices. In: 2017 IEEE 28th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC). pp. 1–6. IEEE (2017)

[9] Clarke, A., Margetts, H.: Governments and citizens getting to know each other? open, closed, and big data in public management reform. Policy & Internet **6**(4), 393–417 (2014)

[10] 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**, 283–297 (2018)

[11] Dannen, C.: Introducing Ethereum and Solidity. Springer (2017)

[12] Do, Q., Martini, B., Choo, K.K.R.: The role of the adversary model in applied security research. Computers & Security **81**, 156–181 (2019)

[13] ethereumWiki: Ethereum project. https://github.com/ethereum/wiki/wiki (2016)

[14] Fan, K., Wang, J., Wang, X., Yang, Y.: Proxy-assisted access control scheme of cloud data for smart cities. Personal and Ubiquitous Computing **21**(5), 937–947 (2017)

[15] Gerdes, S., Bergmann, O., Bormann, C.: Delegated coap authentication and authorization framework (dcaf). draft-gerdes-ace-dcafauthorize-02. Work in progress **66** (2015)

[16] Giacobbe, M., Coco, M., Puliafito, A., Scarpa, M.: A cloud-based access control solution for advanced multi-purpose management in smart city scenario. In: 2014 International Conference on Smart Computing Workshops. pp. 35–40 (Nov 2014). https://doi.org/10.1109/SMARTCOMP-W.2014.7046680

[17] Hu, V.C., Ferraiolo, D., Kuhn, R., Schnitzer, A., Sandlin, K., Miller, R., Scarfone: Guide to attribute based access control (abac) definition and considerations. NIST special publication **800**(162) (2014)

[18] Lämmel, P., Tcholtchev, N., Schieferdecker, I.: Enhancing cloud based data platforms for smart cities with authentication and authorization features. In: Companion Proceedings of the10th International Conference on Utility and Cloud Computing. pp. 167–172. ACM (2017)

[19] 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)

[20] Makhdoom, I., Zhou, I., Abolhasan, M., Lipman, J., Ni, W.: Privysharing: A blockchain-based framework for privacy-preserving and secure data sharing in smart cities. Computers & Security **88**, 101653 (2020)

[21] Sasaki, T., Morita, Y., Jada, A.: Access control architecture for smart city iot platform. In: 2019 18th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/13th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE). pp. 717–722. IEEE (2019)

[22] Steichen, M., Fiz, B., Norvill, R., Shbair, W., State, R.: Blockchain-based, decentralized access control for ipfs. 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). pp. 1499–1506 (July 2018)

[23] Steichen, M., Fiz, B., Norvill, R., Shbair, W., State, R.: Blockchain-based, decentralized access control for ipfs. 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). pp. 1499–1506. IEEE (2018)

[24] Su, K., Li, J., Fu, H.: Smart city and the applications. In: 2011 International Conference on

Electronics, Communications and Control (ICECC). pp. 1028–1031 (Sep 2011)

[25] Sun, J., Yan, J., Zhang, K.Z.: Blockchain-based sharing services: What blockchain technology can contribute to smart cities. Financial Innovation **2**(1), 26 (2016)

[26] Union, E.: Regulation EU No 910/2014 of the European Parliament and of the Council (23 July 2014), http://eur-lex.europa.eu/legal-content/EN/TXT/HTML/?uri=CELEX%3A32014R-0910&from=EN

[27] 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)

[28] Washburn, D., Sindhu, U., Balaouras, S., Dines, R.A., Hayes, N., Nelson, L.E.: Helping cios understand "smart city" . Growth **17**(2), 1–17 (2009)

[29] Wood, G., et al.: Ethereum: A secure decentralised generalised transaction ledger. Ethereum project yellow paper **151**, 1–32 (2014)

[30] Yigitcanlar, T., Velibeyoglu, K., Martinez-Fernandez, C.: Rising knowledge cities: the role of urban knowledge precincts. Journal of knowledge management **12**(5), 8–20 (2008)