SECURITY AND COMMUNICATION NETWORKS
Security Comm. Networks 2016; **9**:5943–5964
Published online 19 February 2017 in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/sec.1748

RESEARCH ARTICLE

FairAccess: a new Blockchain-based access control framework for the Internet of Things

Aafaf Ouaddah*, Anas Abou Elkalam and Abdellah Ait Ouahman

Oscars Laboratory, ENSA of Marrakesh, Cadi Ayyad University, BP 575, 40000 Marrakech, Morocco

ABSTRACT

Security and privacy are huge challenges in Internet of Things (IoT) environments, but unfortunately, the harmonization of the IoT-related standards and protocols is hardly and slowly widespread. In this paper, we propose a new framework for access control in IoT based on the blockchain technology. Our first contribution consists in providing a reference model for our proposed framework within the Objectives, Models, Architecture and Mechanism specification in IoT. In addition, we introduce FairAccess as a fully decentralized pseudonymous and privacy preserving authorization management framework that enables users to own and control their data. To implement our model, we use and adapt the blockchain into a decentralized access control manager. Unlike financial bitcoin transactions, FairAccess introduces new types of transactions that are used to grant, get, delegate, and revoke access. As a proof of concept, we establish an initial implementation with a Raspberry PI device and local blockchain. Finally, we discuss some limitations and propose further opportunities. Copyright © 2017 John Wiley & Sons, Ltd.

KEYWORDS
Internet of Things; security; privacy; access control; bitcoin; blockchain

***Correspondence**
Aafaf Ouaddah, Oscars laboratory, ENSA of Marrakesh, Cadi Ayyad University, BP 575 40000 Marrakech, Morocco.
E-mail: aafafouaddah@gmail.com

1. INTRODUCTION

Thanks to the advancement in chip design, sensors and actuators are nowadays cheap enough to be embedded in any device. Internet of Things (IoT) has extended the digital world to our real and social life by enabling any things and objects that surround us, to be connected to the Internet. IoT is making things, which heretofore were blind and mute, talk, wash, hear, and even think. These billions of devices are pervading our hospitals, factories, roads, airways, offices, retail stores, public buildings, homes, cities, and even our bodies. For the sake of improving our lifestyle, they are unfortunately tracking us and increasingly encroaching on our private and intimate spaces. Indeed, smart meters deduce when we shower, cars know when we do not go to work, wearable medical devices know our weight, and mobiles know how we feel [1]. Moreover, in this open environment where our devices are interconnected and connected to the Internet, unauthorized alteration, access to confidential data, or even denial of service may be much easier. Subsequently, the success or failure of this revolutionary evolution of IoT will be determined by two key challenges: security and privacy. To address those issues, it is imperative to define and implement effective access control mechanisms while preserving privacy. How to provide an adequate access control model for IoT services is a vital but challenging topic. Indeed, authentication and authorization issues have been intensively investigated through existing protocols for use cases outside constrained environments. However, in constrained environments, those issues are still in their infancy. In fact, additional and different requirements pose challenges for the use of various security protocols. In particular, the need arises for a user-driven and fine-grained access control mechanism, where users/resources are constrained.

1.1. Motivation

Most access control solutions today provide the ability for centralized authorities, whether governments, manufacturers, or service providers to gain unauthorized access to and control devices by collecting and analyzing user's data. That may cause ethical and privacy problems. For example, Fitbits, a company that makes small devices to help people keep track of their fitness activities. It turns out that it broadcasts the intimate activity of quite a few of their users [2]. Fitbit, and many other companies, believed that all the data they gathered should be public by default. Unfortunately, when we share our information with third parties, we lose control and ownership straight away. What happens when we decide to stop using a service? What happens to all the data we previously shared with that service provider? From this moment, our data is forever lost and kept inside the databases we have no control over. Our new FairAccess framework breaks this custom and gives people what fairly belongs to them. Actually, we believe that IoT needs a new access control framework that satisfies its specific requirements and characteristics, where users are able to control and master their own privacy. This "shift" will require rethinking access control technologies and building new solution with IoT privacy and security requirement in mind. Hopefully, we stand at the edge of a new phase of decentralization. After over 20 years of scientific research, there have been significant advances in the fields of cryptography and decentralized computer networks, resulting in the emergence of a new technology —known as the blockchain—which has the potential to fundamentally shift our notions of centralized authority. In this paper, we leverage the consistency guarantees provided by this promising technology to build an access control framework for IoT called FairAccess

1.2. Contribution

This paper extends and improves our prior work in [3] with significant new materials. More specifically, our contributions are (i) proposition of a reference model for our proposed access control framework, based on the Objetives, Models, Architectures and mechanisms (OM-AM) way introduced in [4] (ii) conceiving and implementing a new distributed access control framework based on blockchain technology named FairAccess that meets IoT security and privacy arising needs. Actually, unlike financial bitcoin transactions, our framework introduces new types of transactions that are used to grant, get, delegate, and revoke access. It is based on the main following principles. User driven and transparency: User is the master of his own data; he has the full and granular access control over the data he shared in the network or in the cloud. Fairness: Using blockchain in our access control framework improves fairness because nobody could systematically be enforced to loss control over his own data. Distributed architecture and the lack of a central authority: Each node in the network shares his data with others nodes directly, without the intervention of any third or trusted entity. Fine-granularity: The use of smart contact enable user to implement an expressive and granular access control policies over our framework.

1.3. Related work

In one hand, numerous efforts have been emerged in adapting traditional access control models to meet new IoT security requirements. We cite as example the role-based access control (RBAC) model that was extended to a new model named context-based access control in IoT [5]. In this model, the permission is assigned to the role according to the characteristics and contextual information collected from the environment of the physical object. However, privacy and security objectives were not considered. The capability-based access control model was also chosen in [6] where it was directly implemented on resource-constrained devices, within a fully distributed security approach. Ye *et al.* have proposed in [7] an efficient authentication and access control scheme, based on attribute-based access control) for the perception layer of the IoT. Another generic authorization framework for the IoT is proposed in [8]. It supports fine-grained and flexible access control, based on current Internet standards and access control solutions such as eXtensible Access Control Markup Language and Security Assertion Markup Language, for any objects with low power and memory resources.

In other hand, across the industry, many companies implement their own proprietary authorization software based on the OAuth protocol [9], in which they serve as centralized trusted authorities. For instance, European Union project Connect All IP-Based Smart Objects [10] adopt a centralized approach where the authorization logic is outsourced from the smart and constrained device to a more powerful server called IoT-OAS. However, it has demonstrated in [11] the impossibility to run all OAuth logic in a constrained device due to its heavy communication and processing overheads. In this direction, the Internet Engineering Task Force (IETF) is making continuous efforts to adapt OAuth protocol and its profile User-Managed Access with lightweight protocols, such as Constrained Application Protocol [12] and Datagram Transport Layer Security (DTLS), in order to make OAuth fits IoT requirements [13–17].

Unfortunately, those typical security and access control standards today are built around the notion of trust where a centralized trusted entity is always introduced, which harm user transparency and privacy. In addition, they are built around a single logical server and multiple clients. As a consequence, access control is often carried out within the server side application, once the client has been authenticated. IoT reverses this paradigm by having many devices serving as servers and possibly many clients, taking part in the same application. More importantly, servers are significantly resource-constrained, which results in the minimization of the server side functionality. Subsequently, access control becomes a distributed problem. We therefore turn our attention to blockchain, the technology behind bitcoin protocol, to conceive our new FairAccess authorization framework as ultimate solution and equilibrium that solve all IoT authorization challenges extensively highlighted in [13] [18]. Actually, the blockchain is a technology breakthrough that has fundamentally changed our notions of centralized authority. It is a universal digital ledger that functions at the heart of decentralized financial systems such as bitcoin, and increasingly, many other decentralized systems such as Storj,[†] a decentralized

[†] www.storj.io

DOI: 10.1002/sec

peer-to-peer cloud storage network; Onename,[‡] a distributed and secured identity platform; and International Business Machines (IBM's) Adept, an IoT architecture [19]. However, to our knowledge, the use of a blockchain for the instantiation of authorization process has never been fully explored. Except for a recent project called Enigma [20] that proposes a similar objective but within different perspectives to our work.

1.4. Organization

The rest of this paper is organized as follows: Section 2 proposes a reference OM-AM model to our new blockchain-based access control framework. Section 3 introduces and presents an architecture overview of our proposed FairAccess framework. Therefore, we explain its main building blocks and functionalities within a static and dynamic description using Unified Modeling Language (UML) class and interactive overview diagrams followed by detailed technical implementation in Section 4. A proof of work of the model is provided with a real implementation using Raspberry PI in Section 5. We finally discuss the limitations and propose some possible future extensions to our framework in Section 6, while Section 7 concludes the paper.

2. FAIRACCESS'S OM-AM REFERENCE MODEL

We opt for the four layer OM-AM framework coined in [21], and already highlighted in our previous work in [4] to conceive the authorization process in FairAccess. OM-AM stands for Objective, Model, Architecture, and Mechanism. Actually, the objective and model (OM) layers articulate what the security objectives are and what should be achieved, while the architecture and mechanism (AM) layers address how to meet these requirements. The OM-AM framework is approximately analogous to Open Systems Interconnection 7 layer network protocol stack. Like Open Systems Interconnection 7 layers, each OM-AM framework layer's mapping to adjacent layers is many-to-many. In other words, security policy can be formalized with many access control models as they can support different security policies. Moreover, an access control model, in its turn, can be supported by multiple architectures, while a specific architecture can support multiple models and do not necessarily comply with the top-down waterfall-style software engineering process. We argue our choice to adopt the OM-AM framework for analysis because it allows us to (i) define in perspicuous way the boundaries and the relationship between each phase in the authorization process, because each phase matches a specific layer; (ii) discuss each phase independently from the other and as an example, discuss the security requirements separately from the mechanism required for its implementation; (iii) compare in a vertical way different access control policies that encapsulate the same security policy or different architectures that implement the same access control model and different mechanisms that enforce the access control architecture; (iv) compare in a horizontal way within one layer for example between different access controls models/mechanisms; and (v) design each layer separately, for example, design mechanisms that are able to enforce multiple policies [22]. This latter aspect is particularly crucial because it will give great flexibility and scalability to the whole access control system. In fact, if a tool in one layer is tied to a specific component in another layer, changing in the policy would require changing the whole access control system. Let us discuss in details each layer.

2.1. Objectives: security policy

In this phase, we study the characteristics and features of the system to be secured in order to extract its main privacy and security requirements, then fixes the objectives to meet those requirements. Based on those objectives, (high-level) rules, according to which access control must be regulated, are identified. This will help to act as a basis for building our access control framework.

2.1.1. Security and privacy preserving objectives

Unlike other networking systems, new issues are raised, in terms of security and privacy, in the area of IoT, caused by its specific characteristics. We propose the following Security and Privacy-preserving (S&PP) objectives for analyzing Privacy and Security in IoT. These criteria are based on extensive review of the literature and the recent published European Union regulation for electronic identification [23]:

- Privacy: is the ability of an entity to determine whether, when, and to whom personal information can be released or disclosed [24]. An access control system should preserve its users' privacy, which is one of the ways to gain user trust; hence, preserving privacy is a highly related crucial issue in emerging information technology areas, such as IoT. User privacy including user data and personal information should be flexibly preserved according to the policy and expectation of IoT users. We quantify this objective through the following key factors shaping privacy. (i) Transparency: consist in helping people to understand who knows what about them, how their data will be used, with whom it is shared with, and how long it is held. (ii) User-driven: Users are the master of their own data; they have the full and granular access control over the data they shared in the network or in the cloud. (iii) Anonymity: IoT applications are required to not disclose the identity of their users. (iv) Pseudonymity: is a trade-off anonymity with accountability. Actually, actions of a person are linked with a pseudonym, a random identifier, rather than an identity. Pseudonymity might be used to serve many purposes [25],

[‡] https://onename.com

especially resolving privacy and accountability concerns in the IoT. (v) Unlinkability: qualifies pseudonymity in the sense that specific actions of the same person must not be linked together. This requirement might serve as a protection from profiling in IoT. (vi) Unobservability: ensures that a user may use a resource or service without others, especially third parties, being able to observe that the resource or service is being used [26]. It requires that users and/or subjects cannot determine whether an operation is being performed. (vii) Decentralization: Each node in the network shares its data with others node directly, without intervention of any third or trusted entity.

- Technologies constraints: (i) flexibility: Access control should be flexible to be adapted to different contexts. Furthermore, it should support long-lived and planned patterns as well as causal spontaneous and short-lived interaction. Indeed, the collaboration between users is established in unscripted way, and the role of service provider or service consumer can no longer be static or identified a priori, because it can be played alternatively by the same entity. (ii) Scalability: IoT is a more demanding environment in terms of scalability and manageability. Because of the potentially unbounded number of things (resources and subjects), access control mechanism should be extensible in size, structure, and number of users and resources. (iii) Lightweight: Access control, designed for IoT, should support lightweight solutions and standards, because of the low capabilities of power, memory, and computing of IoT devices meaning that all introduced security mechanisms shall be designed such that the total overhead due to computation and communication is as low as possible on the device side. (iv) Heterogeneity: is a key challenge in IoT. Actually, a collaborative environment may combine several technologies and different devices and ecosystems. They may be conceived by diverse constructors and are thought up for diverse purposes and designed for different application domains, making it arduous to achieve a global agreement or adopt any specification
- Social and economic aspect of IoT: There are several social and economic challenges that need to be addressed: (i) interoperability/collaboration: IoT is an ecosystem that requires, in one hand, the collaboration between a plethora of untruthful stakeholders establishing loose relationships, such as individuals, public, and private establishment. In other hand, it establishes communication and information exchange across heterogeneous set of devices. Thus, access control model must be designed for multiple organizations. Each of them sets up its own policies and must respect other collaborating organization's policies. (ii) Context awareness: The context is particularly important in IoT. It provides services and applications that use knowledge and inspire its intelligence from its surrounding contextual information about user and his environment. Actually, in IoT, sensors continuously generate enormous amounts of raw data, and context-aware computing has proven its efficiency to understand this data and add value on it.
- Confidentiality and integrity: Confidentiality means no unauthorized disclosure of resources and integrity is no improper modifications of resources. Moreover, access control system should have the following features: (i) high granularity: means expressiveness of the grammar used to formulate access control rules: the more flexible the grammar is and the more information it can cater for, the more fine grained the resulting access control will be. (ii) Revocation: the ability to revoke permission to access resources and make sure the revoked user cannot access the associated resources anymore. (iii) Delegation: A subject can grant access rights or part of the granted rights to another subject. As many things are owned by their users (either permanently or temporarily) and may belong to a group, it is necessary to consider the design of delegation mechanisms.
- Reliability and availability: The continuity of service leads to reliability, while the readiness for usage leads to availability. In the IoT context, these properties may also be declined to (i) Offline mode: Decisions are made even if its maker (that could be resource owner (RO)) is absent or not connected. (ii) Short-term availability: The IoT resources must be available to authorized users with reasonable response-time as process or data may have timeliness constraints. (iii) Long time availability: Regulations impose that some data must be kept for a very long time (e.g., cancer records must be kept for the patient's lifetime, and records of genetic diseases have to be kept even longer). Only some well-identified users should have the ability to delete those data and only after the appropriate time period has expired.
- Usability: Access control should be easily managed, expressed, and modified. Indeed, with the omnipresence of IoT devices that encompasses every day and personal tools such as toothbrush, fridge, and wearable. Users, with different expertise, are more involved nowadays in authorization activities than in the past.

These dimensions will be used in considering the specific requirements of IoT domain applications.

2.1.2. Internet of Things domain application and their security requirements

In order to build a suitable security mechanism for IoT, we need to understand the nature of its applications and their security requirements properly. Then, it is mandatory, as a first step, to classify, on one hand, which domain application we are dealing with and the various devices that comprise the domain application, on the other hand. Indeed, each domain application has specific characteristics and, thus, special security requirements that have to be taken into consideration to conceive a security solution. This will help to act as a basis for building our access control framework to achieve the required security level for each domain application. To meet this end, we list different application fields of IoT and specify the characteristics and security requirements of each one. Moreover, we give an overview of different taxonomies and classification of constrained devices in IoT. We categorize these applications into three domains: (i) personal and home: at the scale of individual, home, and healthcare; (ii) government and utilities: at the scale of community nation and region; and (iii) enterprise and industry: at the scale of industries and big companies. Actually, many ways have been already introduced in the literature to classify IoT devices. We are interested in this one proposed in [27] that suggests a classification based on spatial closeness to human (intimate, personal, social, and public), using Edward Hall's theory of proxemics [28]. Indeed, from this classification, we can deduce that devices that are closer to human need an access control model that is more user-driven and that gives users full ability to control their intimate devices with their specific granularity.

Contrariwise, devices that are classified as public do not, necessarily, need a fine-grained access control model. In addition, the involvement of end user in access control decision for these type of devices is not required. In Figure 1, we combine the results issued from analyzing domain characteristics, devices, and security and privacy preserving objectives.

We can deduce from our previous analysis and as state in [4] that IoT applications will need to be built on principles of cooperation and collaboration openness/interoperability, high scalability, flexibility, distribution, context-awareness, etc. We can conclude that in personal and home category that encompasses smart home and healthcare applications, end user is considered as pivot element due to its high involvement. As a result, access control models targeting such domains are required to be more users driven. They should allow end user to have full control over his own resources. In the government utilities, enterprise and industry, end user involvement is less important than in the first category. It is unlikely that a "one-size-fits-all approach" in all IoT application but rather many and diverse approaches. A peruse case-specific vulnerability analysis will identify the level of security required for each device in certain applications and that access control solution achieves the required level. However, the common requirements for all IoT applications are high flexibility, scalability, heterogeneity, collaboration between different stockholders, and the need of lightweight security mechanisms due the omnipresence of constrained devices in all IoT application domains. Without appropriate security mechanisms, attackers might gain control over things relevant to our lives. Authentication and authorization mechanisms are therefore prerequisites for a secure IoT. A comprehensive and holistic access control framework for IoT requires that all given objectives can be well achieved.

To bridge the policies and the actual mechanisms to enforce them, an access model or more precisely an authorization model is needed.

2.2. Authorization model

An authorization model is a formalism (often mathematical) for representing in a clear and unambiguous way the security policy. It helps to abstract it (i.e., reduce its complexity) and to facilitate its understanding. It can be used to verify that the policy is complete and consistent. Popular authorization model include discretionary model DAC, mandatory model MAC, RBAC [29] and its extensions, and attribute-based access control model [30] usage control presented by some authors [31–33].

It is important to note that, thanks to the many-to-many relationship for mapping between adjacent layers in our OM-AM authorization reference model, our proposed FairAccess authorization framework is generic and not restricted to the use of any specific access control model. Actually, the access control policy could be expressed with any access control model. But the only condition the model has to meet in order to be integrated in our framework is to be transcoded to a script language and encapsulated inside FairAccess's transactions. Hence, in order to obtain access to a protected resource, an

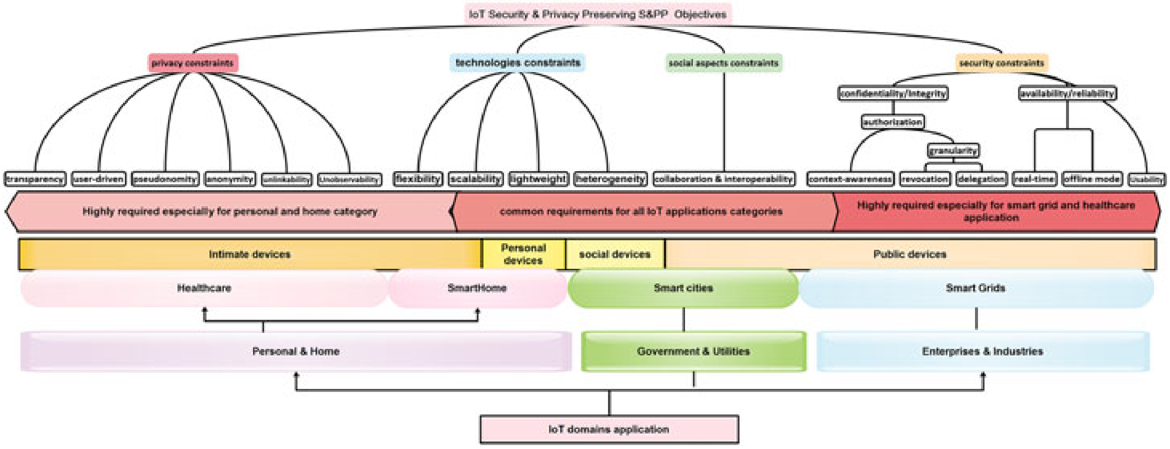


Figure 1. Internet of Things domains application and their security and privacy preserving requirements.

DOI: 10.1002/sec

access token is required. The access Token is not delivered by this transaction until the requester fulfills all access control conditions already described with the model and included in the transaction (more details about FairAccess transactions are given in the next section). As a proof of concept of this idea, two access control models (identity and permissioned access control models) have been used during our implementation in Section 5. For those reasons described previously, the access control model in FairAccess OM-AM reference model will be genereic and not, apriori, restrected to any specific model.

2.2.1. A representative example using Organization Based Access Control (OrBAC) model

The Organization Based Access Control (OrBAC) [34] model is conceived to address existing issues in extended RBAC models. It introduces the notion of "organization" and completely separates the abstract level (roles, views, and activities) of the security policy from the concrete level (user, object, and action). In our context, OrBAC presents several benefits: (i) granularity: OrBAC defines permissions, interdictions and obligations. (ii) Abstraction of the security policy: OrBAC has a structured and an abstracted expression of the policy; it also separates the specification from the implementation of the policy. (iii) Scalability: OrBAC has no limitation in size or capacity. It can define an extensible policy. It is then easily applicable to large-scale environments such as IoT. (iv) Loose coupling: Each organization is responsible for its assets and entities. Implementation details and private information are managed separately by each organization. (v) Flexibility: A policy in OrBAC is evolvable. It easily handles changes in organizations. (vi) User driven: specifying and updating an OrBAC security policy are rather intuitive. (vii) Context-aware: present the powerful point of OrBAC in IoT context, because it takes the context (e.g., specific situations, time, and location constraints) into account. (viii) Popularity: OrBAC has a growing community.

2.2.2. Projection of OrBAC model in Internet of Things [35]

We develop a simplified (but representative) example to project the crucial notions of OrBAC in a scenario adapted to IoT concept. The scenario intends to demonstrate the cross-application nature of objects in one smart city scenario that horizontally connects several application domains, and, more specifically, smart health, smart home, smart living, smart transport, and public safety (Figure 2). Smart cities are actually a real driver for connecting application domains.

Alice, a cardiac woman, has opted for an assisted living service that is provided by a medical center. A doctor, who monitors Alice's health remotely from the medical center, receives an alarm that Alice has fainted. An ambulance is instructed to go to assist Alice. A smart driving application is used by the ambulance to reach Alice's home as quickly as possible. The situation requires the interaction of the following stockholders: (i) smart home of Alice, which is equipped with actuators and sensors. The patient's movements are tracked by sensors, and wireless controllers send this information to the monitoring service, in the medical center, which oversees the patient's condition. (ii) The medical center for monitoring Alice's health and the environmental conditions in the smart home and initiates appropriate action, such as alerting emergency services and sending the smart ambulance. The ambulance requests information from traffic monitoring in the police department in order to find the best route to Alice's home and

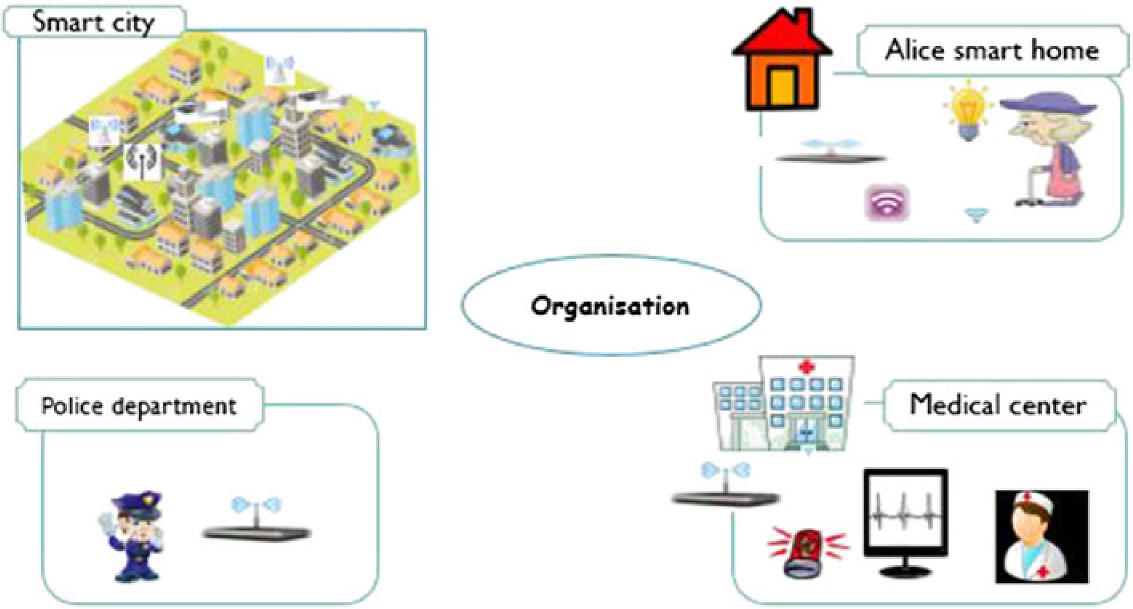


Figure 2. The notion of organization in OrBAC model adopted to Internet of Things environment.

DOI: 10.1002/sec

save valuable time. (iii) The police department for traffic jams monitoring: receives data from the distributed platforms sensors in order to infer the traffic status in the city's streets. Combines real-time data from different types of sensors (e.g., cameras, gas sensors, and magnetic field sensors) and infers the traffic status in the city's streets. (iv) The smart city: which include all the previous stakeholders as a sub-stakeholders and where are various types of sensors, which are connected through wireless sensor network platform using various access technologies and/or communication protocols (ZigBee, Bluetooth, Wi-Fi, etc.) sharing their data. In fact, each stakeholder can be considered as an organization, for example, "the medical center". This organization is structured by different roles, for example, "doctor in the monitoring service," several activities, for example, "consult," and several view, for example, "patient's medical history and received sensor's data from monitored patient," and finally, the context can be, for example, "a medical emergency such as Alice's faint."

According to the OrBAC model, the "smart home of Alice" can be specified as an organization with rules such as "*permission* (smart home of Alice, Bob, sensor's data, consulting, urgency)" and instances (of relations) such as the following:

- *Empower* (org: smart home of Alice, role: neighbor, and subject: Bob),
- *Consider* (org: smart home of Alice, activity: consulting, and action: get),
- *Use* (org: smart home of Alice, view: Alice' heartbeat, and object: sensor's data_xx).

The derivation of the permission (i.e., instantiation of security rules) mentioned previously can be formally expressed as follows (Figure 3).

However, even if OrBAC has several advantages, it is not completely adapted to IoT. In fact, OrBAC is not able to manage collaboration-related aspects. Actually, if we can assume that OrBAC is a model expressing the security policies of several organizations, it is unfortunately only adapted to centralized structures and does not cover the distribution, collaboration, and interoperability needs, while these aspects are very important in the IoT context. In order to overcome the limitations listed previously, we propose to integrate OrBAC, as model to express access control policies in our FairAccess Framework to release a fully distributed approach. The access control policy will be encapsulated in a FairAccess transactions. The development of this improvement is out of the scope of this paper and will be described in future works.

2.3. Authorization architecture

Internet of Things is considered as a collaborative environment. The authorization in IoT inherits, therefore, the characteristics of authorization in collaborative environments such those described in [36], also known as coordinate environments. Furthermore, the main and special feature of IoT is actually the limitation of constrained devices.

Within a collaborative environment, we basically distinguish two levels of the centralization or decentralization of access control: (i) The first level concerns the management of access policies over operations between cooperative organizations: whether a centralized entity is defined as a common authority. Hence, all decisions about whether a user in an organization A is able to access a given Resource in an organization B are taken and implemented at a central point which can be a super-organization which imposes its policy on all other security organizations and the decentralized or chain approach where each organization is responsible for defining its own access control decisions and their implementations. (ii) The second level concerns the management of access control within an organization. But in this case, the management and localization of access control logic could be centralized on a central and dedicated point. Or distributed and located in each protected resource.

In our framework, we propose a centralized and decentralized approach to manage access control policies in IoT. Indeed, we propose a decentralized approach at the first level that concerns the interactions between cooperatives organizations. At this level, we opt for a peer to peer approach (fully distributed) where each organization is responsible of defining and implementing its own security policies. Besides that, at the second level, because typical resources (*R*) in IoT environments are expected to be constrained devices unable to carry out all the authorization functions by themselves; then, we opt for a centralized approach where a centralized entity, called authorization manager point (AMP) is defined for each organization. This AMP manages and approves authentication and authorization data for a Resource. It can be responsible for

Permission(**Smart home of Alice**, user, sensor's data_xx , Consulting, Urgency)) ∧

Empower(**Smart home of Alice**, neighbor, Bob) ∧

Consider(**Smart home of Alice**, Consulting, GET) ∧

Use(**Smart home of Alice**, Alice'heartbeat, sensor's data_xx) ∧

Hold(**Smart home of Alice**, Bob, GET, Alice'heartbeat, Urgency)

⇨ *Is_Permitted* (Bob, GET, sensor's data_xx)

Figure 3. Derivation of permissions in OrBAC in Internet of Things scenario.

a single or multiple devices or even for a whole network and a resource may have multiple AMP. We note the AMP in organization *A* side: "A-AMP" and in the organization *B* side: "B-AMP" (Figure 4).

In order to comply with the privacy preserving objective, defined in Section 2, aiming to empower and emphasis user involvement in defining and managing access control process over its own resources, we introduce the resource owners (A-RO) and (B-RO) in organization A and B, respectively. Those later decide about the security policies and the level of authorization's granularity of their respective endpoints belonging to the same organization. They are the only responsible for controlling and managing their corresponding AMP. A-RO: is the person who is in charge of requested resource and controls its access permissions. B-RO: The person who is in charge of the requesting resource. He controls the requests a resource in organization B (B-R) makes. Our proposed architecture is depicted in Figure 5.

2.4. Authorization mechanism

A mechanism defines the low level functions to enforce policies and define how access requests are evaluated against those policies. In our proposed framework, in order to fulfill the P&SP objectives, we will use the following mechanisms:

In FairAccess, we use, **Bitcoin-like addresses**: to identify all interacting entities. **Smart contract/scripting language** to express fine-grained and contextual access control policies enveloped inside transactions. We opt for **authorization tokens** distributed by the blockchain. We use **blockchain** firstly to ensure evaluating and enforcing access policies and secondly to ensure token integrity and detect token double spending. The detail of each mechanism is described below:

- Bitcoin-like addresses to ensure unlikability and pseudonymity.
- **Blockchain**: FairAccess provides several useful mechanisms using the blockchain. In fact, in FairAccess, the blockchain is considered as a database or, a policy retrieval point, where all access control policies for each pair (resource, requester), are stored in form of transactions, it serves also as logging databases that ensures auditing functions. Furthermore, it prevents forgery of token through transactions integrity checks and detects token reuse through the double spending detection mechanism.
- SmartContract/ scripting language: to express access control policies. it is worth to note that in this article, we focus more on the deployement of FairAccess within the UTXO model of blockchain. Then, in this version, access control conditions are exppressed by scripting language. However, the feasibility of using Smartcontract in our framework will be explored in [37].
- Authorization token: In our FairAccess framework, we define an Authorization Token as a digital signature that represents the access right or the entitlement defined by the creator of the transaction to its receiver in order to access a specific resource identified by its address. We introduce this authorization token as a new field stored on the blockchain as data, encrypted using the in-built cryptocurrency public/private key mechanisms.

In fact, the token-based access control enforced by the blockchain technology provides many advantages in IoT context. Actually, having the token securely held on the blockchain means smart devices can easily verify the validity of the access token relieving IoT constrained devices from the burden of handling a vast amount of access control-related information and at the same time mitigates the need for outsourcing these functionalities to a trusted powerful entity that prevent

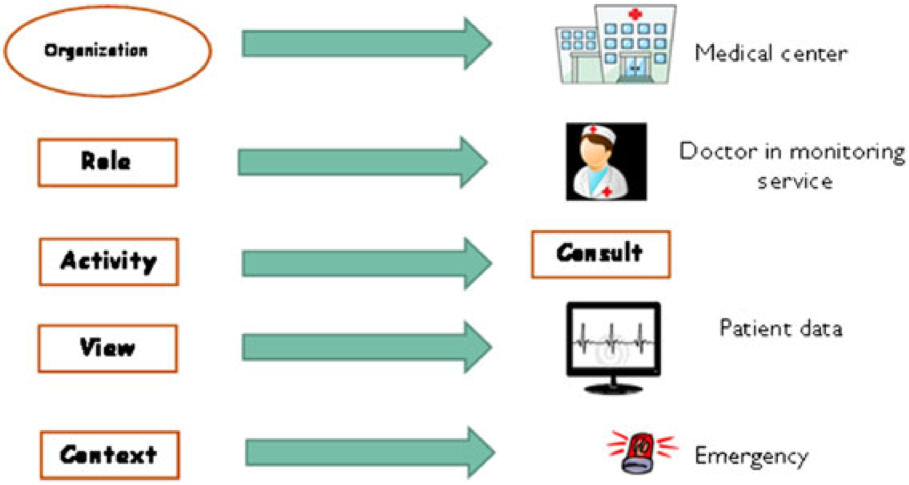


Figure 4. Projection of OrBAC in Internet of Things.

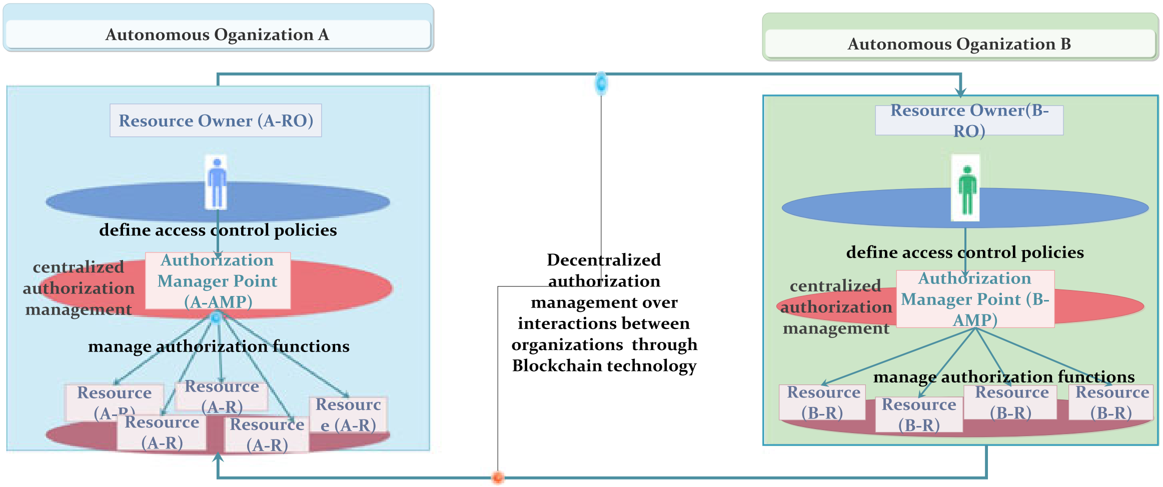

Autonomous Oganization A
Autonomous Oganization B
Resource Owner (A-RO)
define access control policies
centralized authorization management
Authorization Manager Point (A-AMP)
manage authorization functions
Resource (A-R)
Resource (A-R)
Resource (A-R)
Resource (A-R)
Resourc e (A-R)
Decentralized authorization management over interactions between organizations through Blockchain technology
Resource Owner(B-RO)
define access control policies
centralized authorization management
Authorization Manager Point (B-AMP)
manage authorization functions
Resource (B-R)
Resource (B-R)
Resource (B-R)
Resource (B-R)

Figure 5. Architecture layer in Objectives, Models, Architecture and Mechanism reference model. Note that this architecture is compatible with the one proposed in this work in progress by the IETF [38].

end-to-end security to be achieved. Moreover, it reduces communication cost, since no further authentication mechanisms are required to get the token. Since only signature is sufficient. In addition, the token could be used for many access control operations such as getting, delegating, revoking and even updating access in an easier and flexible way.

We describe the whole OM-AM reference model for our framework in Figure 6.

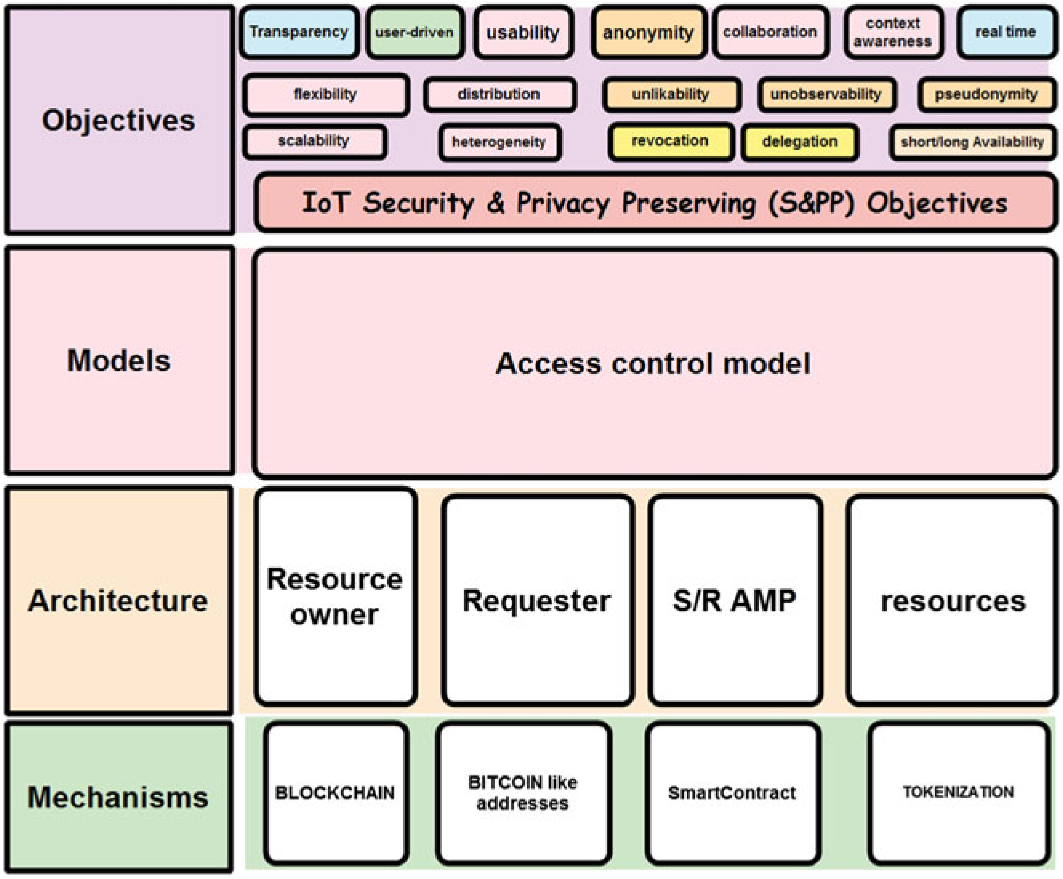


Figure 6. Objectives, Models, Architecture and Mechanism reference model.

3. PROPOSED FRAMEWORK

Referring to the OM-AM framework developed in Section 2 as reference model to build our FairAccess framework and based on the objectives and model layers that are extensively presented in that section, in this section, we introduce and describe our proposed framework and its whole functionalities.

3.1. Architecture overview

The vision of our decentralized access control framework is a system of autonomous organizations hinged around one or many ROs in possession of one or many resources identified with addresses and interacting between each other through transactions (requesting, granting, delegating, and revoking access) under the control of their RO. The blockchain is a ledger keeping track and ensure the validity of access transaction among interacting organization. Each manages its own access policy, under only the control of his RO, resulting in an "Internet of Decentralized, Autonomous Organization" as depicted in Figure 7 and thus the fairness of our access control framework

3.2. Our framework building blocks: a static description

The building blocks of our framework are described through UML class diagram in Figure 8.

3.2.1. User

It can be either an RO, belonging to supplier organization, who exposes his resources and control access over it, or a requester, belonging to requesting organization, who owns the resource aiming to access to supplier resources. Each of those entities: RO, a requester, and their resources are identified with addresses, interacting between each other through transactions.

3.2.2. Wallet

Every user has a wallet that stores his credentials, addresses and the transactions related to them. It contains all the keys needed to register and identify his resources, sign his transactions, and ask for access. In our framework, we consider a wallet as AMP, it could be a web or mobile application, through the wallet, the RO could register his resources need to be protected, define his access control policies. Then, the main functionalities of a wallet are (i) generating keys and addresses. (ii) Transform the access control policies to a transactions and broadcast those later to the network. (iii) validate transactions received from the network.

3.2.3. Addresses

In our framework, users either a RO or the requester and their resources can have a virtually unlimited amount of cryptographic identities, called addresses. Addresses are public and shared in the network. They are used to grant and ask for an access token. An address is basically the hash of an Elliptic Curve Digital Signature Algorithm (ECDSA) public key and a user in possession of the corresponding private key is said to own the address. We use address to identify all entities in our framework: RO, requester, and all types of their resources. Addresses enable many of the interesting properties, including decentralized trust and control, ownership attestation, and the cryptographic-proof security model.

3.2.4. Transactions

A transfer of bitcoins between addresses is called a transaction. Actually, the fundamental building block of a bitcoin transaction is an *unspent transaction output*, or UTXO. UTXO are a value of bitcoin currency locked to a specific owner, recorded on the blockchain, and recognized as currency units by the entire network. The UTXO consumed by a transaction are called transaction inputs, and the UTXO created by a transaction are called transaction outputs. In our framework, we do not transfer bitcoin but rather, we transfer access token, then the UTXO of our framework's transactions correspond to an access

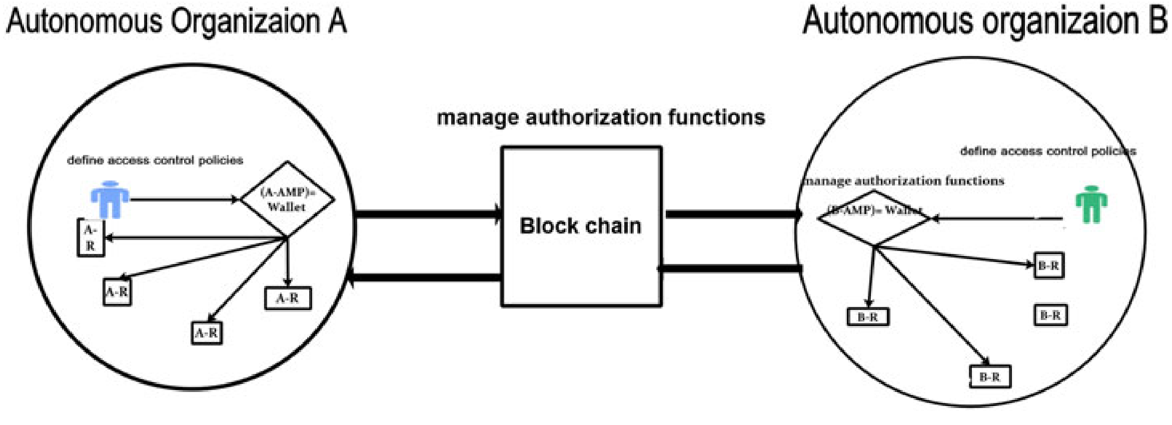

Autonomous Organizaion A
Autonomous organizaion B
manage authorization functions
define access control policies
(A-AMP)= Wallet
A-R
A-R
A-R
A-R
Block chain
define access control policies
manage authorization functions
(B-AMP)= Wallet
B-R
B-R
B-R
B-R

Figure 7. FairAccess architecture overview.

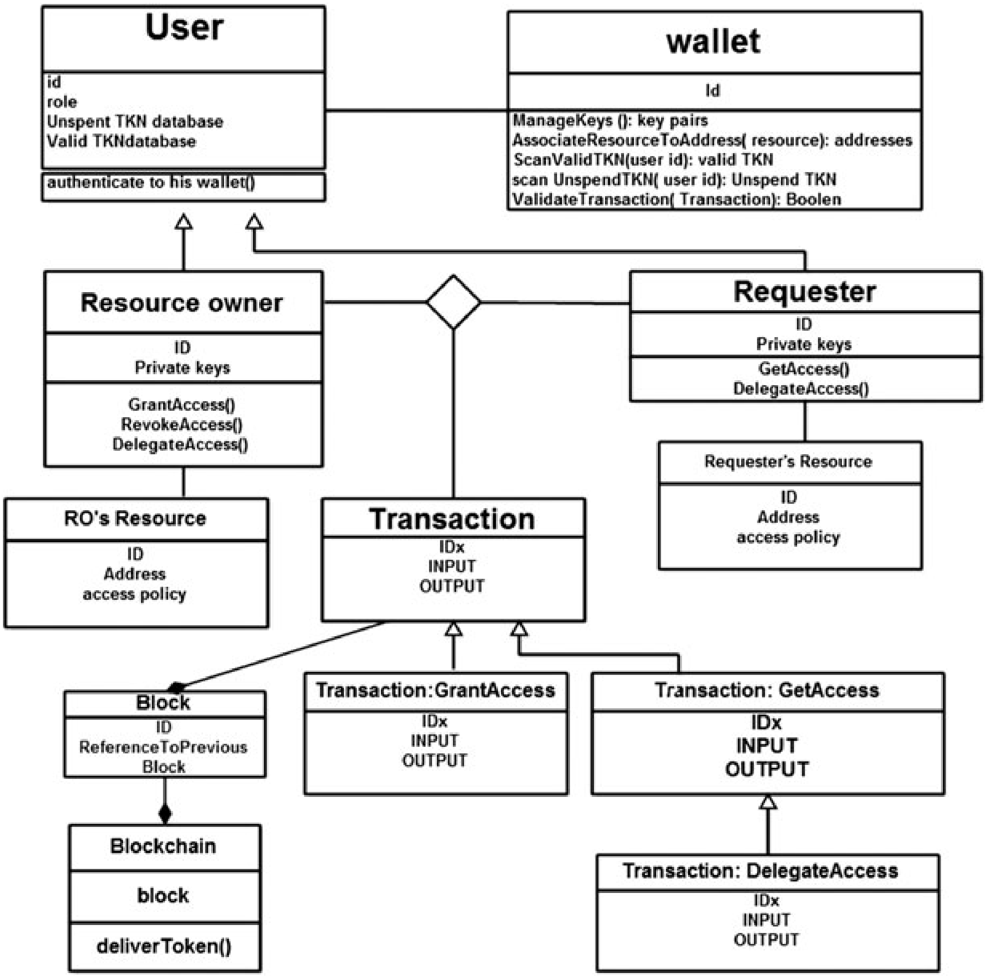


Figure 8. FairAccess UML class diagram.

token TKN, recognized by the entire network as an access rights defined by the creator of the transaction to the receiver of the transaction to access Resources identified by their addresses in the transaction. This way, we move forward from owner to either a requester or a new owner in a chain of transactions consuming and creating TKN. Then, we introduce two types of transaction: (i) a TokenBase transaction also named GrantAccess transaction where an RO defines an access policy and new TKN are created. And a regular transaction that could be either a GetAccess transaction: where a requester spends a *TKN* that he already obtains from a previous GrantAccess transaction, to access a resource identified with an address (by fulfilling access control conditions) or Delegate Access: where a requester delegates access, by transferring *TKN* that already owns, to another new owner under new conditions.

We note that every access token TKN is encrypted with the public key, extracted from the address of the requesting party to which the TKN is designated. Hence, we are sure that, only the requesting entity to which the TKN is created will be able to decrypt with his private key and obtain access to the TKN.

Any transaction is composed with an identifier *IDx*, an input vector *Vin []*, and an output vector *Vout [ ]*.

IDx is the transaction identifier. We uniquely and unambiguously identify a transaction with its cryptographic hash, which is a digital fingerprint, made by hashing the transaction. The result hash is called the transaction identifier *IDx*. This *IDx* can be independently derived by any node by simply hashing the transaction.

Vin []: Each input in the vector inputs has the following components. *Index*: the index of the input in the vector input. *Reference to previous output*: refers to the previous TKN by reference to the transaction identifier and sequence number where the TKN is recorded in the blockchain. The reference is uniquely identified by the tuple (*hash*, *index*). The hash field, referred to the *IDx* of previous transaction and the specific output within that transaction is identified by the output index. We note that for GrantAccess transaction, this file did not point to any previous transaction, because this type of transaction generate a new access token TKN. *Resource address*: the address of the requested resource. *Unlocking scripts*: a script that satisfy the spending conditions set by the RO.

In this field, the requester fulfills the access control policies, expressed in scripting language.

Vout [ ]: Each output in the output vector consists of two parts. *Index* is the index of the output in the vector. *Value*: fee of the transaction. TKN: an unspent access token, *requester address*: who wants to access the resource. *Locking script*: this script that "locks" this TKN by specifying the conditions (access control policies) that must be met to spend the TKN.

3.2.5. Peer-to-peer network

The peer-to-peer network is forming a loosely connected mesh without a fixed topology or any structure, making all nodes equal peers. Messages, including transactions and blocks, are propagated from each node to the peers to whom it is connected. Although nodes in the P2P network are equal, they may take on different roles depending on the functionality they are supporting. Especially in IoT scenario that encompass a large spectrum of devices capabilities. A full node in the network is a collection of functions: routing, the blockchain database, mining, and wallet services.

Then, the senders do not need to trust the nodes they use to broadcast the transaction, as long as they use more than one to ensure that it propagates. The nodes do not need to trust the sender or establish the sender's identity. Transactions can therefore be transmitted to the P2P network over insecure networks such as Wi-Fi, Bluetooth, Near Field Communication (NFC), and barcodes. This property enables all IoT devices, even those placed in secure location or connected to an insecure network like sensors or Radio Frequency IDentification (RFID) tag, to interact between each other through transactions.

3.2.6. Blockchain

The blockchain is a database that stores all processed transactions in chronological order shared by all users or nodes participating. We assume this memory to be tamper-proof under the same adversarial model used in bitcoin and other blockchains. The blockchain technology provides everyone with a working proof of a decentralized trust. All cryptocurrencies utilize what can best be described as a public ledger that is impossible to corrupt. Every user or node has the exact same ledger as all of the other users or nodes in the network. This ensures a complete consensus from all users or nodes in the corresponding currencies blockchain. In our framework, the blockchain is considered as a database that stores all access control policies for each pair (resource and requester) in form of transactions and serves also as logging databases that ensures auditing function.

3.3. Our framework functionalities: a dynamic description

Our authorization framework proposes the following authorization functionalities: (i) validate transaction; (ii) registering a new resource with a corresponding address; (iii) GrantAccess; (iv) RequestAccess; (v) DelegateAccess; and (vi) RevokeAccess as illustrated in the interactive overview diagram depicted in Figure 9. We describe each function in this paragraph.

We will denote key pairs using the capital letters (e.g., A) and refer to the private key and the public key of A by $A.sk$ and $A.pk$, respectively (then $A = (A.sk, A.pk)$). In addition, we will a use the following convention: if $A = (A.sk, A.pk)$ then let $sig_A(m)$ denote a signature on a message m computed with $A.sk$ and let $check_A(m, \sigma)$ denote the result (*true or false*) of the verification of the signature σ on the message m with respect to the public key $A.pk_,$ and finally, we note $\mathscr{H}$ as a hash instantiated by a SHA-256 [11] implementation.

(1) Transaction validation protocol

Here, we provide a detailed description of the core protocols executed on the blockchain. Transaction validation protocol is executed by nodes in the network when a transaction is received. Every full node, independently, validates every transaction before propagating it further. A malformed transaction will not get beyond one node. The validity of the transaction is checked against the following verifications described in Figure 10.

(2) Registering a new resource

In our framework, identity establishment and proof of ownership are established through digital keys, addresses, and digital signatures. Actually, keys come in pairs consisting of a private (secret) key and a public key. From the private key, we use elliptic curve multiplication, a one-way cryptographic function, to generate a public key $A.pk$. From the public key $A.pk$, we use a one-way cryptographic hash function $\mathscr{H}$ to generate an address. A private key $A.sk$ is a number, picked at random: The private key is used to create signatures that are required to (i) prove ownership of resources and control access (define access control policy) over resources identified with addresses extracted from its corresponding public keys in GrantAccess transaction type, and (ii) prove the possession of the Access *TKN* transferred in delegate access transaction. More precisely, the public key of a public/private key pair is used to identify a particular recipient, whereas the private key is used to create a signature for both transaction authentication and integrity.

- Key generation: We opt for the following hierarchical and deterministic method to generate addresses identifying our framework entities: keys are derived in a tree structure, such that a parent key can derive a sequence of children keys, each of which can derive a sequence of grandchildren keys, and so on, to an infinite depth. Resource's addresses have to be extracted from a child public key of the seed corresponding to its RO. Meaning that from one seed

DOI: 10.1002/sec

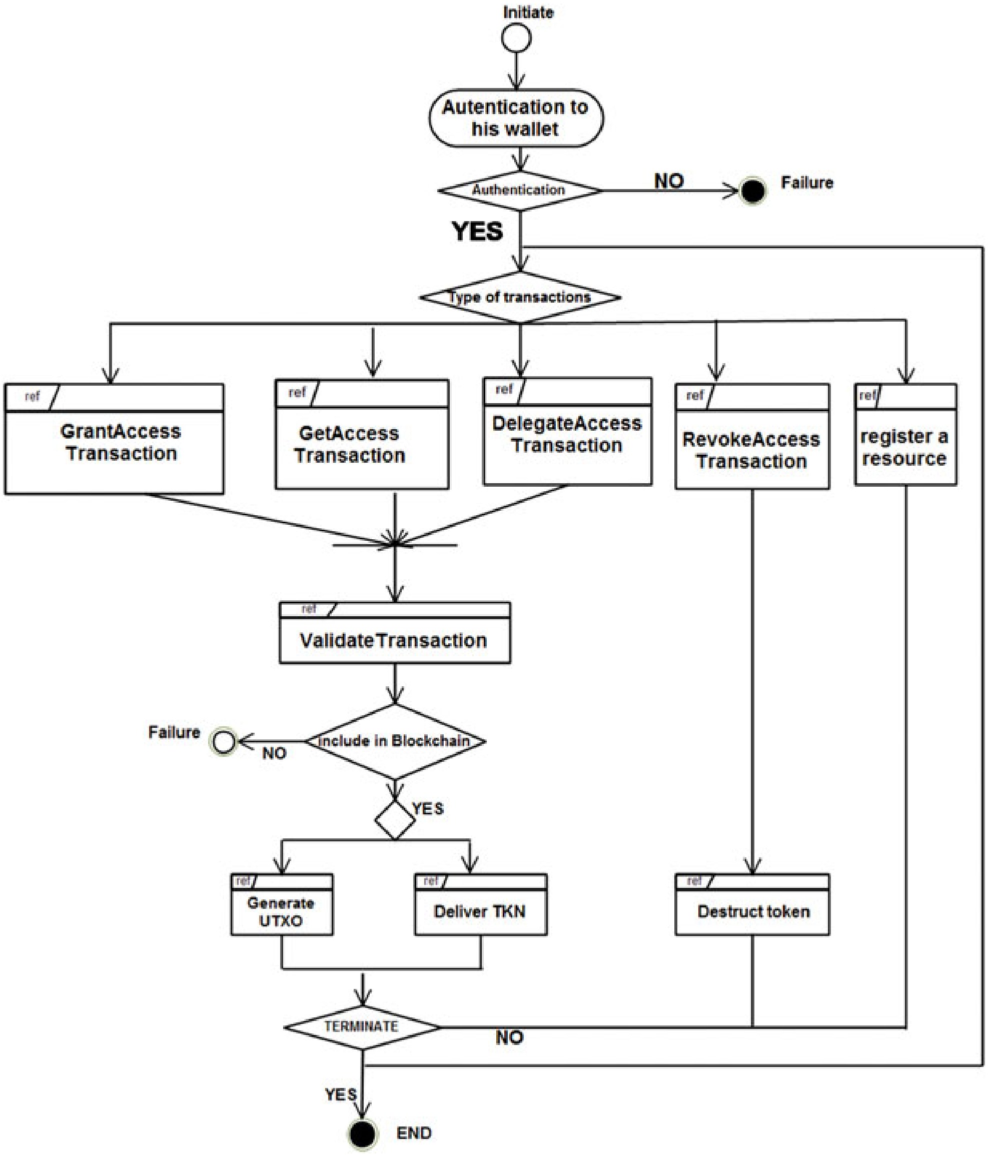


Figure 9. FairAccess interactive overview diagram.

key, the RO is able to identify unlimited number of resources, by extracting address from the generated keys. Furthermore, he can generate different addresses to the same resources from another seed key, for each transaction to enhance pseudonymity and unlinkability. Actually, this method offers our framework the following characteristics as depicted in Figure 11: (i) enabling a very high involvement and transparency of RO: the ability to derive public child keys from public parent keys, *without* having the private keys. Such a kind of deployment can produce an infinite number of public keys and addresses but cannot give the possibility to produce valid transaction without the permission of RO who generates addresses. That means the RO who generated the seed is the only person who can define and manage access control policies for all his registered resources by deriving the corresponding private keys to sign transactions. That could be very useful for resources like IoT devices with constrained capabilities or placed in insecure locations.

Those resources can use public key derivation function to create a new address for every transaction (e.g., in a elderly healthcare scenario when a monitoring service requests access to data of medical sensors that are widespread in the city); those interacting entities (monitoring sensors) could obtain a public key to obtain an address to receive access request without knowing any private keys that would be vulnerable to theft. But they cannot decide either to grant or deny access without the permission of their RO. Because he is the only entity

Transaction Validation protocol:

- A Node receives a signed transaction in the following form:

$$(T_x, sig_A(\ T_x)\ )where \quad T_x = (ID_x, Vin\ [input1\ (\ ref, A.pk, \psi)], Vout[output1\big(value, B.pk, \pi_x,\ TKN_{A.pk,B.pk}\big)]\ )$$

IDx: is the identifier of the current transaction T_x where $x\ =\ \mathcal{H}\ (T_x)$.

ref= point to the previous transaction output/in GrantAccess transaction this field is "Tokenbase"

$A.pk$: the address of requested resource.
ψ: unlocking script./ do not exist in a GrantAccess transaction
$TKN_{A.pk,B.pk}$:access token. *value*: fee of the transaction.
π_x: *locking script.*

$B.pk$: *i*s the address of the requester(either the Requester or his resource)who is the receiver of the current transaction

- Then the node executes the following functions:

✓ CheckIdentity: checking the signature of the owner prove the following properties:

1) Authenticate the owner. 2) prove his ownership to the resource 3) prove his non repudiation

$$Check_A(T_x, \sigma) =\ True$$

✓ CheckIntegriy (T_x) : Hash the transaction and compare to its ID_x to ensure that the transaction has not been altered during its propagation in the network

Figure 10. Transaction validation protocol.

Registering a new resource protocol:

✓ Address generations

The wallet generate, in offline mode, a key pair:

- Root seed derivation and generation of the first key pair A:

$$\mathcal{R}andomGenerator\ (k) \rightarrow\ Rs\ where\ Rs\ is\ the\ \ Root\ seed$$

$$\mathcal{H}(Rs) \xrightarrow{HMAC-SHA512} (\underbrace{code}_{256bits}\ , \underbrace{First\ private\ key}_{256bits})$$

$$First\ private\ key \rightarrow A.sk$$

$$g(A.sk) \xrightarrow{elliptic\ curve\ multiplication} A.pk$$

- Child key derivation function :

$$\mathcal{H}(parent\ private\ key, parent\ code, index) \xrightarrow{HMAC-SHA512} (\ R\ Right\ side/256bits\ , L\ Left\ side/256bits)$$

$$R\ \ \rightarrow Child\ code$$

$$\mathcal{H}\left(parent \begin{cases} public \\ private \end{cases} key, L, index\right) \xrightarrow{HMAC-SHA256} child \begin{cases} public \\ private \end{cases} key$$

✓ Address association

For each resource a corresponding address $ResourceAdress$ is generated from the public key

$$\mathcal{H}(\ pk) \xrightarrow{HMAC-SHA256} hashpk$$

$\mathcal{E}(hashpk) \rightarrow ResourceAdress$

Where: $\mathcal{E}$ is encode function for human readability

Figure 11. Registering a new resource protocol.

that controls the private key and that can sign transactions, in this way, our framework preserves in a very succinct way the user privacy. (ii) Great scalability: Each parent extended key can have many billion children. Each of those children can have another billion of children, and so on. The tree can be as deep as user wants, with an infinite number of generations. (iii) Flexibility: The R/S RO can prove his ownership and control the access to all his resources associated to generated addresses by just one private key which is the seed key. (iv) The seed is sufficient to recover all the derived keys, and therefore, a single backup at creation time is sufficient.

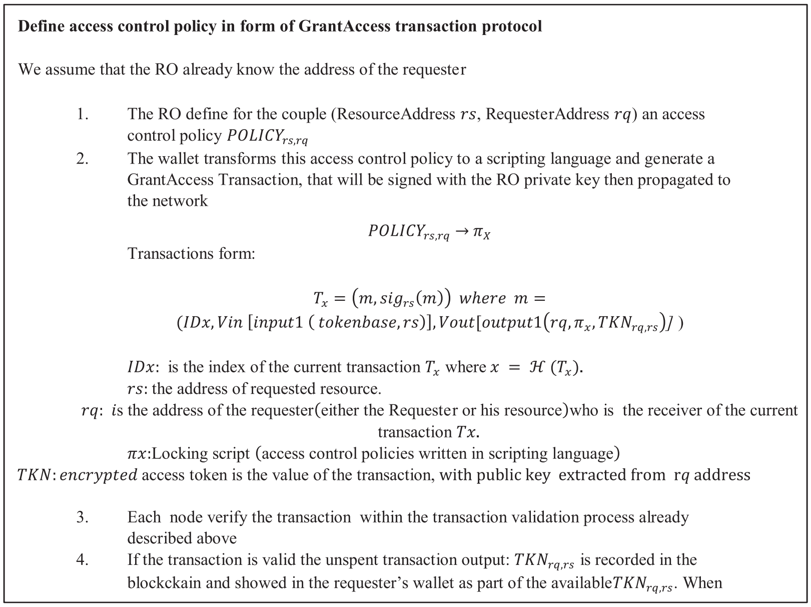
(3) Grant access to a requester through a GrantAccess transaction type

This type of transaction is frequently used in our framework. As demonstrated in Figure 12, through this transaction, the RO defines access control policy in form of conditions (unlocking script) to grant access over one or several resources identified by their addresses, to a requester, represented also by his address, who is the receiver of this transaction. When we generalize the blockchain's computation to arbitrary Turing complete logic, transactions can contain an infinite number of conditions that we can refer to expressive "smart contract" system such as Ethereum [30]. The use of smart contract will enable our framework to express fine-grained context-aware access control policies. A policy is a set of rules and conditions (based on a specific context or attribute, etc) that a requester entity has to fulfill in order to obtain the access token and get access to the specific resource. These rules could be expressed by any access control model but must be transformed to a script language considered as locking script placed on an output in the GrantAccess transaction. Then, the wallet broadcasts the transaction to the network, the network verifies, validates the transaction and includes it into the blockchain if it was valid else it will be rejected and a notification will be sent to the owner. The transaction describe the fact that an access token *TKN* (the value of a transaction) is delivered to address $B.pk$ to access a resource, identified by $A.pk$ address, under the πy condition defined by $A.pk$ owner.

At the end of this phase, if the transaction appears in the blockchain, it means that a new *encrypted* $TKN_{rq,rs}$ is added to the requester *Available TKN* database. That means the network witnesses that the RO had defined an access right to that requester to access to that resource but the requester has not access yet till he unlocks the access condition and decrypt the $TKN_{rq,rs}$ with his private key than spend it. To do so, the requester have to prove to the network that he fulfills really the access condition in a new transaction called GetAccess transaction which is the objective of the second phase explained in Figure 13.

(4) Access to resource through a GetAccess transaction type

In order to the requester solves the unlocking script (fulfill access control policies) and spends the value of the TokenBase transaction which is an access token $TKN_{rq,rs}$. The requester uses this type of transaction to prove to the network his fulfillment of access condition.



Define access control policy in form of GrantAccess transaction protocol

We assume that the RO already know the address of the requester

1. The RO define for the couple (ResourceAddress rs, RequesterAddress rq) an access control policy $POLICY_{rs,rq}$
2. The wallet transforms this access control policy to a scripting language and generate a GrantAccess Transaction, that will be signed with the RO private key then propagated to the network

$$POLICY_{rs,rq} \rightarrow \pi_X$$

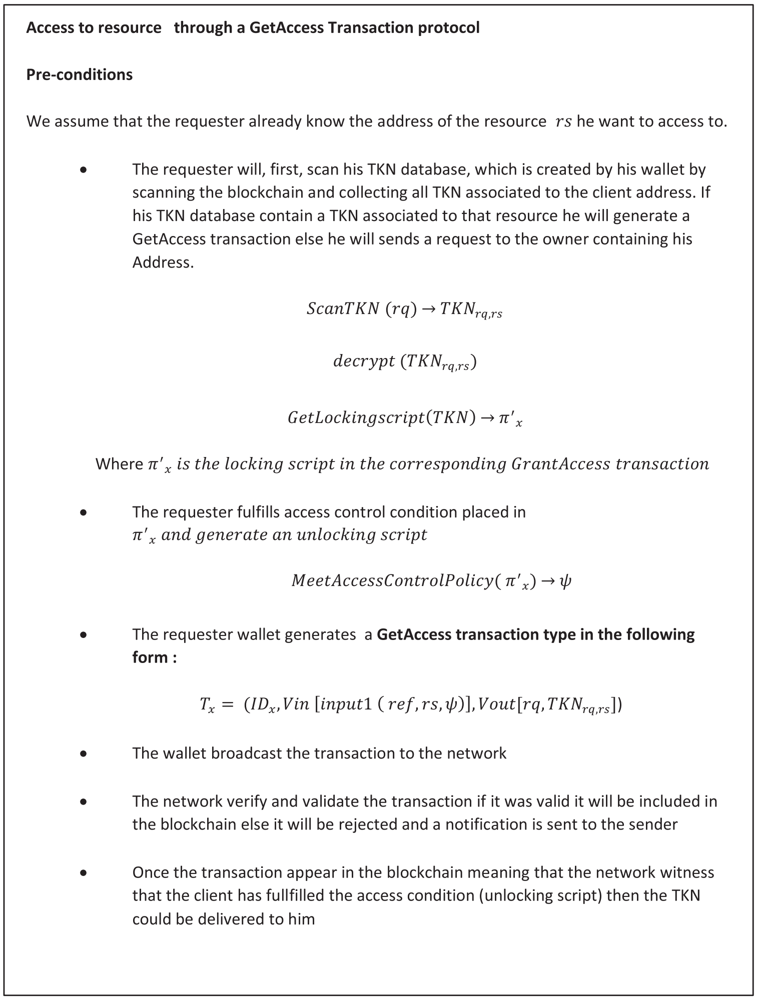
Transactions form:

$$T_x = \big(m, sig_{rs}(m)\big)\ where\ m = (IDx, Vin\ [input1\ (\ tokenbase, rs)], Vout[output1\big(rq, \pi_x, TKN_{rq,rs}\big)]\ )$$

IDx: is the index of the current transaction T_x where $x = \mathcal{H}\ (T_x)$.
rs: the address of requested resource.
rq: is the address of the requester(either the Requester or his resource)who is the receiver of the current transaction Tx.
πx:Locking script (access control policies written in scripting language)
TKN: $encrypted$ access token is the value of the transaction, with public key extracted from rq address

3. Each node verify the transaction within the transaction validation process already described above
4. If the transaction is valid the unspent transaction output: $TKN_{rq,rs}$ is recorded in the blockckain and showed in the requester's wallet as part of the available$TKN_{rq,rs}$. When

Figure 12. GrantAccess transaction protocol.



Access to resource through a GetAccess Transaction protocol

Pre-conditions

We assume that the requester already know the address of the resource rs he want to access to.

- The requester will, first, scan his TKN database, which is created by his wallet by scanning the blockchain and collecting all TKN associated to the client address. If his TKN database contain a TKN associated to that resource he will generate a GetAccess transaction else he will sends a request to the owner containing his Address.

$$ScanTKN\ (rq) \rightarrow TKN_{rq,rs}$$

$$decrypt\ (TKN_{rq,rs})$$

$$GetLockingscript(TKN) \rightarrow \pi'_x$$

Where π'_x *is the locking script in the corresponding GrantAccess transaction*

- The requester fulfills access control condition placed in π'_x *and generate an unlocking script*

$$MeetAccessControlPolicy(\ \pi'_x) \rightarrow \psi$$

- The requester wallet generates a **GetAccess transaction type in the following form :**

$$T_x =\ (ID_x, Vin\ [input1\ (\ ref, rs, \psi)], Vout[rq, TKN_{rq,rs}])$$

- The wallet broadcast the transaction to the network
- The network verify and validate the transaction if it was valid it will be included in the blockchain else it will be rejected and a notification is sent to the sender
- Once the transaction appear in the blockchain meaning that the network witness that the client has fullfilled the access condition (unlocking script) then the TKN could be delivered to him

Figure 13. GetAccess transaction protocol.

We note that the requester can spend this $TKN_{rq,rs}$ either by access to resource rs in this case the $TKN_{rq,rs}$ will be spend to him in the output of the transaction or by delegating access to another one in a delegating access transaction (Figure 14).

The inclusion of this transaction into the blockchain enables the delivery of the encrypted access token $TKN_{rq,rs}$ to the requester.

(5) Delegate access through a DelegateAccess transaction

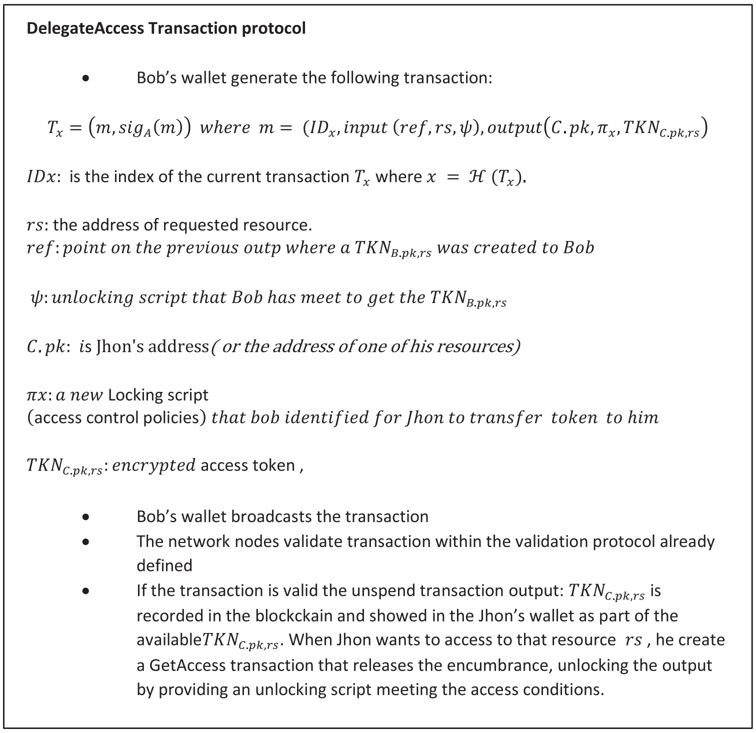
The requester Bob identified with $B.pk$ can grant access rights or part of the granted rights to another requester Jhon identified with $C.pk$ to access resource rs through this transaction.

(6) Revoke access through a RevokeAccess transaction type

The RO could revoke access by introducing the time-validity of the authorization token TKN, as a filed, to be checked in the token. This and this later method will be discussed in future work.

4. IMPLEMENTATION AND TYPICAL USE CASE

As a "proof of concept," we have established an initial implementation and execution to the presented protocol as demonstrated in figure 17. Actually, our proposed authorization framework FairAccess can be used for a variety of IoT application systems such as transportation, smart home, and healthcare. To demonstrate the usefulness of the proposed framework and to illustrate the user experience, we consider as a typical use case the following scenario. In this scenario, the target system is a smart security camera with a remote-control function that can take care of babies. It is built using a Raspberry Pi 2 board with its dedicated camera. The Raspberry Pi is connected to a WLAN providing a remote access. The camera represents the resource to control access to, in



DelegateAccess Transaction protocol

- Bob's wallet generate the following transaction:

$$T_x = (m, sig_A(m))\ where\ \ m = \ (ID_x, input\ (ref, rs, \psi), output(C.pk, \pi_x, TKN_{C.pk,rs})$$

IDx: is the index of the current transaction T_x where $x\ =\ \mathcal{H}\ (T_x)$.

rs: the address of requested resource.
$ref: point\ on\ the\ previous\ outp\ where\ a\ TKN_{B.pk,rs}\ was\ created\ to\ Bob$

$\psi: unlocking\ script\ that\ Bob\ has\ meet\ to\ get\ the\ TKN_{B.pk,rs}$

$C.pk$: is Jhon's address*(or the address of one of his resources)*

πx: $a\ new$ Locking script
(access control policies) $that\ bob\ identified\ for\ Jhon\ to\ transfer\ \ token\ \ to\ him$

$TKN_{C.pk,rs}$: $encrypted$ access token ,

- Bob's wallet broadcasts the transaction
- The network nodes validate transaction within the validation protocol already defined
- If the transaction is valid the unspend transaction output: $TKN_{C.pk,rs}$ is recorded in the blockckain and showed in the Jhon's wallet as part of the available$TKN_{C.pk,rs}$. When Jhon wants to access to that resource rs , he create a GetAccess transaction that releases the encumbrance, unlocking the output by providing an unlocking script meeting the access conditions.

Figure 14. DelegateAccess transaction protocol.

consequence the authorized requester, depending on their rights, defined by the camera owner, could perform multiple actions (video recording, live streaming, time-lapse photography, etc.). As a proof of concept, we will take a snapshot and save it on the Raspberry Pi SD card, define our control policy, and give the requester a remote access using a token distributed by our local blockchain. Hence, the overall system consists of four major components:

RO the owner of the camera

Requester it could be a service provider or any user who wants to access to the camera

Resource is a Raspberry Pi camera module that is used to take high-definition video. It has a five megapixel fixed-focus camera that supports 1080p30, 720p60, and VGA90 video modes, as well as stills capture. It attaches via a 15-cm ribbon cable to the CSI port on the Raspberry Pi. In order to use the camera module, there are three applications provided, raspistill, raspivid, and raspistillyuv. Raspistill and raspistillyuv are very similar and are intended for capturing images; raspivid is for capturing video. In this scenario, we will use raspistill -v -o Token1.jpg. The camera will take a picture within 5 s, save it to the file Token1.jpg. By giving the -v option, we will obtain various informational messages.

Blockchain and bitcoin network For a testing environment, there is bitcoin's test network (testnet) or regression test mode (regtest). In this scenario, we are using a regtest, where all types of bitcoin network nodes are running on the local machine. We have chosen bitcoinj as implementation of the bitcoin protocol. This library provide us a wallet and perform the required operation: send/receive transactions without needing a local copy of bitcoin core and has many other advanced features.

RO wallet (AMP) The Raspberry Pi 2 is a low cost, credit card-sized computer that includes quad-core ARM Cortex-A7 CPU and 1 GB of RAM. It is considered in this scenario as the RO AMP and node in bitcoin network. So we have installed the bitcoin core 0.11.× on Raspberry Pi using Berkeley DB version 4.8. After Setting up the wallet, we run our bitcoin node using bitcoind regtest to join the local blockchain. And the bitcoin client bitcoin-cli regtest to send and receive transactions. The regtest option helps us to obtain a self-contained testing environment: Everything the Peers included could be running our raspi. We note that, Because of the technical constraints of the IoT objects, in this scenario, we use lightweight wallets. These kinds of wallets use simplified payment verification mode that only requires them to download part of the blockchain.

Requester wallet (AMP) We opt for a non-IoT device, a laptop running Windows 7 and installing a bitcoin core.

Figure 15 shows the major components of the system and the major interactions between them.

We provide more details about our implementation steps below:

Step 1 add camera as a resource to be protected (assign an address to the camera).

Step 2 define an access control policy to the requester.

In this scenario, the RO defines two access control policies:

- Identity-based access control policy:

Access over the camera is granted to the requester after being successfully authenticated and proved his identity. The wallet transforms this access control policy to script using pay-to-public key hash script.

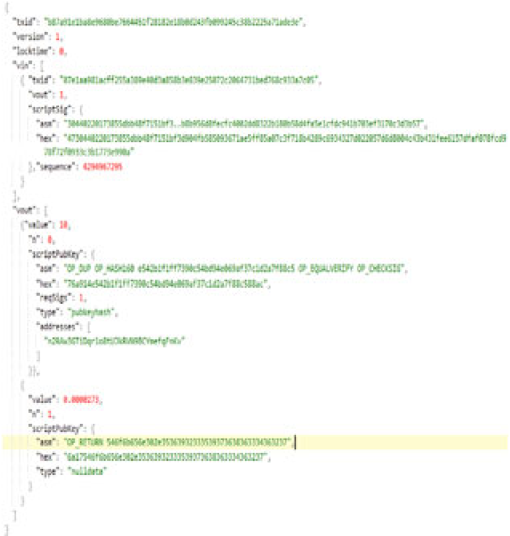
- Permissioned-based access control policy:

Access over the camera is granted to the requester after being successfully authenticated and proved his identity. In addition, the RO wants to keep his control over the token for each use. Meaning that, the requester could not use the token without the permission of his owner whenever he intend to use or delegate this token for example. The wallet transforms this access control policy to script using a multisig public key script.

Step 3 create GrantAccess transaction

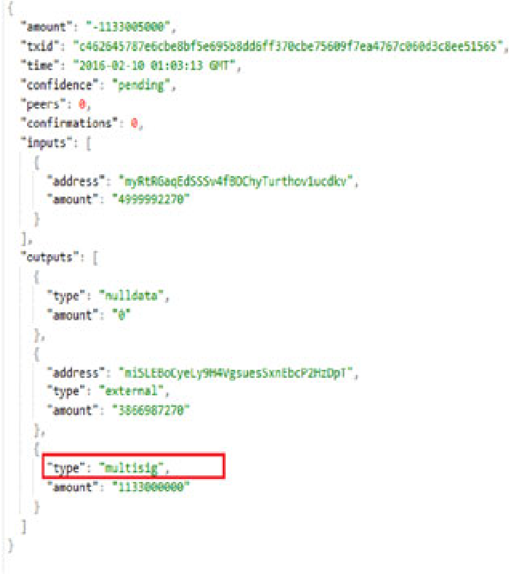
To transfer the token encrypted in the transaction, we had used OP_RETURN. Actually, Bitcoin provide us the ability to store within OP_RETURN, 80 bytes of arbitrary data in the blockchain. Enough for our token that is an SHA256 hash concatenated to an 8-byte custom header as a prefix, the needed size is 40 bytes. So the access token is encapsulated inside the output script using the OP_RETURN operation.

- In the first case where access control policy is based on identity, the wallet creates the following transaction:



```
{
  "txid": "b87a01e1ba8e9680be7664461f28182e18b0d243fb099245c38b2225a71ade3e",
  "version": 1,
  "locktime": 0,
  "vin": [
   { "txid": "87e1aa981acff255a308e48d3a858b1e639e25872c2064731bed768c933a7e05",
    "vout": 1,
    "scriptSig": {
     "asm": "304402201573855dbb48f7151bf3..b8b95648fecfc4082dd8122b180b54d4fe5e1cfdc941b705ef3170c3d7b57",
     "hex": "47304402201573855dbb48f7151bf3d904fb585093671ae5ff85a07c3f718b4289c6934327d022057d6d8004c43b431fee0157dfaf078fcd9
      70f72f0933c5b1779e998a"
    },"sequence": 4294967295
   }
  ],
  "vout": [
   {"value": 10,
    "n": 0,
    "scriptPubKey": {
     "asm": "OP_DUP OP_HASH160 e542b1f1ff7390c54bd94e069af37c1d2a7f88c5 OP_EQUALVERIFY OP_CHECKSIG",
     "hex": "76a914e542b1f1ff7390c54bd94e069af37c1d2a7f88c588ac",
     "reqSigs": 1,
     "type": "pubkeyhash",
     "addresses": [
      "n2Wu5GT1Dqr1o8tUJkRU6NBCVeefqFnKx"
     ]
    }},
   {
    "value": 0.0000273,
    "n": 1,
    "scriptPubKey": {
     "asm": "OP_RETURN 546f6b656e382e35363932333539373638363334363237",
     "hex": "6a17546f6b656e382e35363932333539373638363334363237",
     "type": "nulldata"
    }
   }
  ]
}
```

- In the second case where access control policy is permissioned, the wallet creates this following transaction:



```
{
  "amount": "-1133005000",
  "txid": "c462645787e6cbe8bf5e695b8dd6ff370cbe75609f7ea4767c060d3c8ee51565",
  "time": "2016-02-10 01:03:13 GMT",
  "confidence": "pending",
  "peers": 0,
  "confirmations": 0,
  "inputs": [
    {
      "address": "myRtRGaqEdSSSv4fBDChyTurthov1ucdkv",
      "amount": "4999992270"
    }
  ],
  "outputs": [
    {
      "type": "nulldata",
      "amount": "0"
    },
    {
      "address": "miSLEBoCyeLy9H4VgsuesSxnEbcP2HzDpT",
      "type": "external",
      "amount": "3866987270"
    },
    {
      "type": "multisig",
      "amount": "1133000000"
    }
  ]
}
```

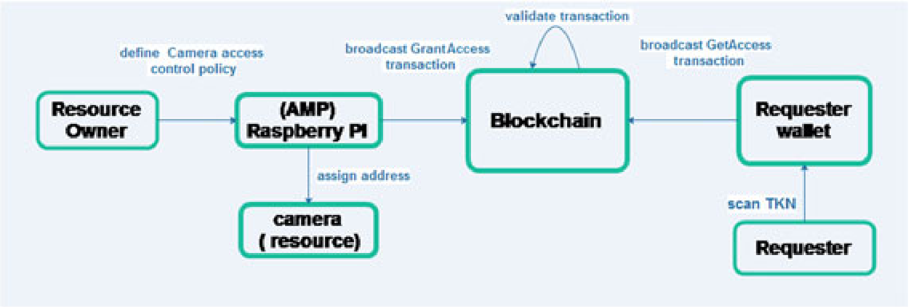


Figure 15. The major components of the implementation and the major interactions between them.

- A picture of the implemented architecture is shown in Figure 16, where camera module is connected to a Raspberry Pi hardware running a bitcoin client node.

5. RESULTS

Overall, the system worked as intended and showed the following aspects:

- Both IoT and non-IoT resources were identified with bitcoin addresses.
- The RO was able to define two access control policies over the camera through his (AMP) running on the Raspberry Pi. The first policy is an identity-based access control, and the second is a permissioned access control.
- The wallet transforms these policies to a script language using pay-to-public key hash script and multisig public key script.
- The token was encapsulated in the transactions using OP_RETURN.
- The wallet sends the GrantAccess transaction.
- The network validates the transactions.
- The requester scans with his wallet access tokens delivered to him.
- The requester fulfills access control condition to obtain the TKN through GetAccess transaction
- The requester obtains the TKN.

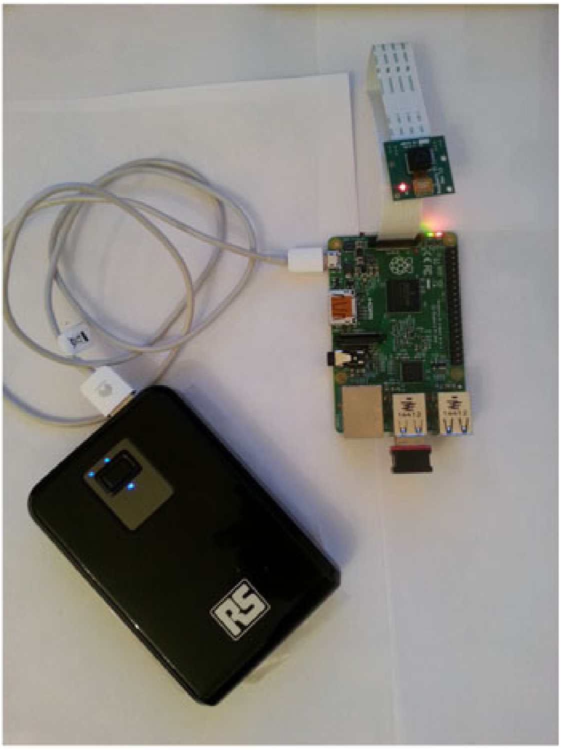


Figure 16. Picture of the implementation.

However, we did find several issues during the implementation, which we will discuss next.

- Real-time: The design of the decentralized, proof-of-work consensus method used by bitcoin is adjusted to maintain roughly 10-min confirmation times. For applications that require high integrity, multiple confirmations may be required. A common requirement is to wait for six confirmations, which can lead to wait times over an hour.
- Blockchain bloat: with the bitcoin blockchain size limit of 1 MB per block, transaction throughput is capped at seven transactions per second. In addition, OP_RETURN is opening a debate inside the bitcoin community about storing nonfinancial data into the blockchain is acceptable or not. To avoid such limitation, building a custom blockchain reserved to record tokens would be a worthy project.
- Granularity: The scripting language used in Bitcoin allows us to transcode two types of access control policies: identity-based access control and permissioned access. We think that using a Turing-complete scripting language such the one used in Ethereum will enable our framework to transcode more complex and granular access control model. The use of SmartContract instead of scrinpting language in FairAccess will be explored in [37].

6. FUTURE EXTENSIONS

In this section, we slightly give hints of possible future extensions to our system. These could play a noteworthy role in craving more mature confidence-preserving storing system.

6.1. Storage layer using Distributed Hash Table (DHT) and secure multiparty computation

Our described system allows users to have a transparent and granular access control over their registered resources, but in the case where data are stored in a cloud or a storage network, the storing system may obtain access to the data. So to ensure that only user who is able to control his data even if it was stored either by centralized system (like Dropbox or Google Drive) or decentralized architecture such as bittorrent, We propose an additional secure, decentralized, and off-chain storage layer for storing files such as an electronic medical record (EMR) or sensor's data, or even any simple Microsoft Word document. The functionality of this off-chain layer consists in encrypting data delivered by registered resources with an encryption key generated by the associated wallet, in a way that ensures its decryption with a hidden key. If the access time determined to the requesting party runs out, this later will be unable to obtain access to encrypted data because he will never know

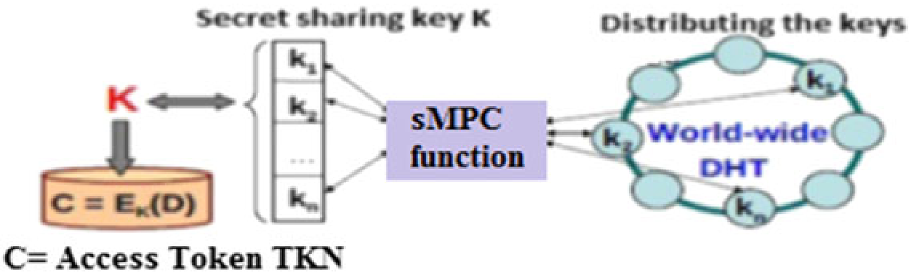


Figure 17. Encrypting token with hidden key.

the encryption key. To do so, we propose using secure Multi Party Computation (sMPC) protocol originating from the seminal works of Yao [39] to obtain to the decryption key through TKN without knowing the key. We propose to split the encryption key into shares (e.g., using Shamir's secret sharing [40]) and use the shares as input in sMPC function and distributed the shares in a distributed hash table DHT, then only pointer to key will be stoked in the blockchain included in the encrypted TKN as depicted in Figure 17. Hence, a storage provider cannot learn anything about the raw data, as it is encrypted with keys that only the user owns and controls.

6.2. Billing model

The majority of IoT application is based on big data analytics and the collection of consumer information, so we think that it is fair to reward the user for sharing his information or allocating his resources since those resources constitute the fuel of those emerging services. With FairAccess, not only users will have a transparent and granular access control over their resource but also they can earn money by allocating or sharing them. By leveraging the economic aspect of the blockchain, this later will not only be considered as an access control layer but also as an economic overlay to IoT application.

7. CONCLUSION

Recently, many efforts have overcome many of the technological requirements for the integration of smart objects into the current Internet. However, IoT paradigm has still to face hard challenges related to the application of security and access control mechanisms over constrained environments. In this paper, we have inaugurated a new applicability domain of blockchain that is access control through our access control framework FairAccess. Our framework leverages the consistency offered by blockchain-based cryptocurrencies such as bitcoin to provide a stronger and transparent access control tool. We have explained, within a static and dynamic description using UML class and interactive overview diagrams, the main building blocks and functionalities of our framework. Moreover, we have proposed a detailed technical implementation of our framework protocol, and as a "proof of concept," we have established an initial implementation and execution to the presented protocol using a typical IoT use case with a Raspberry Pi and its camera module for babies monitoring. During the implementation, the RO was able to define an identity-based access control and permissioned access policy. Afterwards, we have discussed the limitations that we have faced during our implementation using the UTXO model of blockchain mainly the real time and bloat blockchain issues and propose some possible solutions. Finally, we have listed some possible future extensions to our framework.

REFERENCES

1. Mousannif H, Khalil I. The human face of mobile. In *Information and Communication Technology, Lecture Notes in Computer Sciences*, Vol. **8407**: 2014. Springer: Bali, Indonesia, 2014; 1–20. doi:10.1007/978-3-642-55032-4_1.
2. The dark side of wearables: How they're secretly jeopardizing your security and privacy. Online available: http://www.techrepublic.com/article/the-dark-side-of-wearables-how-theyre-secretly-jeopardizing-your-security-and-privacyn.d.
3. Ouaddah A, Elkalam AA, Ouahman AAIT. Towards a novel privacy-preserving access control model based on blockchain technology in IoT. In *Europe and MENA Cooperation Advances in Information and Communication Technologies*. Springer International Publishing, 2017; 523–533.
4. Ouaddah A, Mousannif H, Abou Elkalam A, Ouahman AAIT. Access control in The Internet of Things: Big challenges and new opportunities, Computer Networks (2016), doi: 10.1016/j.comnet.2016.11.007
5. Zhang G, Tian J. An extended role based access control model for the Internet of Things. In: Information Networking and Automation (ICINA), 2010 International Conference on. IEEE, 2010. p. V1-319-V1-323.
6. Hernández-Ramos JL, Jara AJ, Marín L, *et al.* Dcapbac: embedding authorization logic into smart things through ECC optimizations. *International*

Journal of Computer Mathematics 2014, no ahead-of-print:1–22.

7. Ye N, Zhu Y, Wang R-c, *et al.* An efficient authentication and access control scheme for perception layer of Internet of Things. *An International Journal Applied Mathematics & Information Sciences* 2014; **8**:1617–1624.
8. Seitz L, Selander G, Gehrmann C. Authorization Framework for the Internet-of-Things. In Proc. of the 14th IEEE International Symposium and Workshops on a World of Wireless, Mobile and Multimedia Networks (WoWMoM'13), Madrid, Spain, pages 1–6. IEEE, June 2013.
9. Hardt D. (ed), "The OAuth 2.0 Authorization Framework," IETF, RFC6749, October 2012, available at http://www.rfc-editor.org/rfc/rfc6749.txt
10. Connect All IP-Based Smart Objects (CALIPSO)—FP7 EU Project.[Online]. Available: http://www.ict-calipso.eu/, accessed Oct. 15, 2014.
11. Cirani S, Picone M, Gonizzi P, *et al.* IoT-OAS: an OAuth-based authorization service architecture for secure services in IoT scenarios. *Sensors Journal, IEEE* 2015; **15**(2):1224–1234.
12. Shelby Z, Hartke K, Bormann C. "The constrained application protocol (coap)," IETF RFC 7252, vol. 10, June 2014.
13. Yao AC-C. How to generate and exchange secrets (extended abstract). In 27th Annual Symposium on Foundations of Computer Science, pages 162–167. IEEE Computer Society Press, October 1986.
14. Tschofenig H. "The OAuth 2.0 Bearer Token Usage over the Constrained Application Protocol (CoAP)" IETF Internet Draft, draft-tschofenig-ace-oauth-bt-01.txt 2015
15. Tschofenig H. "The OAuth 2.0 Internet of Things (IoT) Client Credentials Grant" IETF Internet Draft, draft-tschofenig-ace-oauth-iot-00.txt 2014.
16. Wahlstroem E. "OAuth 2.0 Introspection over the Constrained Application Protocol (CoAP)" IETF Internet Draft, draft-wahlstroem-ace-oauth-introspection-01.txt 2015.
17. Tschofenig H, Maler E, Wahlstroe E, Erdtman S. "Authentication and Authorization for Constrained Environments Using OAuth and UMA" IETF Internet Draft, draft-maler-ace-oauth-uma-00.txt 2015.
18. Ouaddah A, Mousanif H, *et al*. access control model in the Internet of Things: the road ahead. In the proceeding of the Proceeding of the 12th ACS/IEEE International Conference on Computer Systems and Applications (AICCSA).
19. Panikkar S, Nair S, Brody P, Pureswaran V. ADEPT: An IoT Practitioner Perspective, DRAFT COPY FOR ADVANCE REVIEW, IBM (2015).
20. Zyskind G, Nathan O, Pentland A. Decentralizing privacy: using blockchain to protect personal data. In *Security and Privacy Workshops (SPW), 2015 IEEE*. IEEE, 2015; 180–184.
21. Sandhu R. Engineering authority and trust in cyberspace: The OM-AM and RBAC way. In Proceedings of the fifth ACM workshop on Role-based access control. ACM, 2000; 111–119.
22. Di Vimercati SDC, Foresti S, Jajodia S, *et al.* Access control policies and languages in open environments. In *Secure Data Management in Decentralized Systems*. Springer: US, 2007; 21–58.
23. Regulation (EU) No 910/2014 of the European Parliament and of the Council of 23 July 2014 on electronic identification and trust services for electronic transactions in the internal market and repealing Directive 1999/93/EC OJ L 257, 28.8.2014, p. 73–114 (BG, ES, CS, DA, DE, ET, EL, EN, FR, GA, HR, IT, LV, LT, HU, MT, NL, PL, PT, RO, SK, SL, FI, SV).
24. Yan Z, Holtmanns S. Trust modeling and management: from social trust to digital trust. IGI Global, 2008; 290–323.
25. Pfitzmann A, Köhntopp M. Anonymity, unobservability, and pseudonymity—a proposal for terminology. In *Designing privacy enhancing technologies*. Springer: Berlin Heidelberg, 2001; 1–9.
26. ISO IS 15408, 1999, http://www.commoncriteria.org/
27. Jincy VJ, Sundararajan S. Classification mechanism for IoT devices towards creating a security framework. In *In Intelligent Distributed Computing*. Springer International Publishing, 2015; 265–277. doi:10.1007/978-3-319-11227-5_23.
28. Marquardt N, Greenberg S. Informing the Design of Proxemic Interactions. *IEEE Pervasive Computing* 2012; **11**(2): 14–23.
29. "Role Based access control" NIST.gov - Computer Security Division - Computer Security Resource Center.n.d
30. Yuan E, Tong J. Attributed Based Access Control (ABAC) for Web Services. In Proceedings of ICWS'05: IEEE International Conference on Web Services. IEEE Press: Orlando, FL, USA, 2005; 569–578.
31. Park J, Sandhu R. Towards usage control models: Beyond traditional access control. In SACMAT'02: Proceedings of the Seventh ACM Symposium on Access Control Models and Technologies, ACM, New York, NY, USA, 2002; 57–64.
32. Park J. Usage control: A unified framework for next generation access control, Ph.D. Thesis, George Mason University, Fairfax, VA, USA, 2003.
33. Zhang X. Formal model and analysis of usage control, Ph.D.Thesis, George Mason University, Fairfax, VA, USA, 2006.
34. Miège. A, Definition of a Formal Framework for Specifying Security Policies: The Or-BAC Model

and Extensions, Ph.D. Computer Security, ENST - INFRES Computers and Networks, ENST, 2005.

35. Ouaddah, A, Bouij-Pasquier, I, Elkalam, AA, *et al*. Security analysis and proposal of new access control model in the Internet of Thing. In Electrical and Information Technologies (ICEIT), 2015 International Conference on. IEEE, 2015; 30–35.
36. Sujansky WV, Faus SA, Stone E, Flatley Brennan P. A method to implement fine-grained access control for personal health records through standard database queries. *Journal of Biomedical Informatics* nd:S46–S50.
37. Ouaddah A, Elkalam AA, Ouahman AAIT. Harnessing the power of blockchain technology to solve IoT security & privacy issues. In *Second Int. Conf. Internet Things, Data Cloud Comput. (ICC 2017)*. ACM - International Conference Proceedings Series (ICPS): Cambridge City, United Kingdom; 2017.
38. Gerdes S, Seitz L, Selander G, Bormann C. (ed). “An architecture for authorization in constrained environments ”, IETF Internet Draft, draft-gerdes-ace-actors-05 -04-2015.
39. Federal Information and Processing Standards. FIPS PUB 180-4 Secure Hash Standard (SHS). (March), 2012.
40. Shamir A. How to share a secret. Communications of the ACM, 22(11):612–613, 1979 Adi Shamir. How to share a secret. Communications of the ACM, 22(11):612–613, 1979.

DOI: 10.1002/sec