The Security Framework of Fed4IoT

Pedro Gonzalez-Gil
pedrog@um.es
Dept. Ingeniería de la Información y las Comunicaciones
Murcia, Spain

Antonio F. Skarmeta
skarmeta@um.es
Dept. Ingeniería de la Información y las Comunicaciones
Murcia, Spain

Juan Antonio Martinez
jamartinez@odins.es
Odin Solutions
Alcantarilla (Murcia), Spain

ABSTRACT

The high number of IoT devices, and also their availability through the Internet, has made the topic of IoT virtualisation an emerging topic, which has gained a lot of interest from both academia and industry points of view. Fed4IoT is an H2020 EU-JPN Research Project, whose aim is precisely this one. Nevertheless, security, and more specifically authorisation or access control is a fundamental aspect that must be addressed, motivated by the increase of threats and attacks that the IoT domain has suffered. In this paper we propose the use of a distributed authorisation mechanism based on DCapBAC technology, to specifically deal with the IoT virtualisation aspect, in the scope of this project. This technology has proven its validity in the IoT domain because of its distributed nature and the flexibility of their authorisation policies.

CCS CONCEPTS

• **Security and privacy → Domain-specific security and privacy architectures**.

KEYWORDS

iot, security, privacy, aaa

ACM Reference Format:
Pedro Gonzalez-Gil, Antonio F. Skarmeta, and Juan Antonio Martinez. 2020. The Security Framework of Fed4IoT. In *CCIoT' 20: Workshop on Cloud Continuum Services for Smart IoT Systems, June 03–05, 2018, Waseda University, Tokyo, JP*. ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

The goal of Fed4IoT is to decouple developers from providers in the IoT realm. It does so by virtualizing physical things, sensors, actuators and other data sources by means of its virtualizing architecture: VirIoT[5]. It allows owners of heterogeneous IoT infrastructures to share them with many IoT application developers, which can simply rent the Virtual Things and the IoT Brokers their applications need.

In such a scenario, with many more actors and with the mind of further monetizing existing architectures by renting virtualized parts of it, the case of security becomes even more important. ABAC strategies have proved to be good candidates to express the finely grained policies required for similar scenarios.

In this paper, we present the security architecture of VirIoT. It is based on the DCapBAC [6] access control framework, a flexible and performant verson of the Attribute-Based Access Control (ABAC) strategy, that relies on XACML for policy definition and evaluation, that decouples the authorization request from the access to the resource. It does so by utilizing an intermediary element; the Capability Token, which enables the enforcement of policies in the Policy Enforcement Point, without the need of access to the Policy Decission Point, making the system more reliable, performant and scalable; attributes especially well suited in the IoT Scenario.

2 BACKGROUND

The Cloud business has led in the past few years, to the massive outsourcing of the IT infrastructures of many big capital companies. This rising trend and increase in both demand and available services, has lead to another interesting market opportunity: IoT Cloud Services. This way, many internet-capable small devices, can be easily deployed everywhere for tasks such as equipment control and resource tracking, without having to face the initial investment in data handling infrastructure and communications.

2.1 IoT Cloud Services

IoT applications have sophisticated requirements in terms of coordination across connected objects and cross-compatibility against multiple clouds and networks. It is needless to say that these are complex matters, that are ideal candidates to be addressed by specialized IoT platforms offered by a plethora of vendors and ICT giants. From Intel IoT platform or Bosch IoT Cloud, passing through systems integrators like IBM Watson IoT, to Google Cloud IoT, AWS IoT and Microsoft Azure IoT, companies and individuals searching for an IoT solution, have plenty to choose from.

All of the aforementioned IoT Cloud Services have similar architectures. Usually, the first layer begins with the phisical "Thing" layer, followed for an "Edge" layer (being more or less explicity depending on the specific technology), followed by a "Cloud" layer and ending with a "Data" layer.

Again, in all of those IoT Cloud Services, the user is invited to bring his/her own set of sensors and actuators to the vendor architecture, where they are offered with many functions to analyse data, integrate devices and inspect the resulting data via automated dashboards. Together with those, usually come improved security, scalability to the edge of billios of sensors and data messages, flexible deployment strategies and tools and SDKs for application development.

CCIoT' 20, November 16, 2020, Waseda University, Tokyo, JP

ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/10.1145/1122445.1122456

2.1.1 Differences with Fed4IoT. The previously mentioned platforms, mainly offer Cloud Services for the customer's IoT devices, extending their potential at a relatively low cost. Fed4IoT VirIoT, instead, focuses on offering *things as-a-service*. With that, the desired outcome is to acquire control of the ever-growing number of devices already deployed in the field and virtualising them. In doing so, an scalable layer of horizontally share-able IoT resources can be supplied to customers.

Another difference brought by our focus on **sharing** and **reusing**, is the adoption of the flexible and standardized NGSI-LD data model, based on property graphs, as the internal format for all IoT data. This allows us to integrate information from an object or different linked objects, coming from data sources owned and maintained by heterogeneous and unrelated parties. Put in simple words, Thing virtualization allows us to personalize, transform and translate data from real Things, into objects that can be shared, used in novel applications and rented by customers, creating a new, extended market for those devices.

2.2 IoT Platforms and Brokers

IoT systems can be generally described as a set of interconnected devices, handled by an IoT software platform. At the core, such platforms usually rely on IoT *Brokers*: components exposing IoT data and services of the conected devices though unified APIs and data models. The usual flow of information in such Brokering scenarios is that devices publish their data to the Broker, which in turn sends it to the interested subscribers.

Two different IoT platforms have attracted much attention by industry and academia alike, thanks to the endorsement received so far by standardization bodies and industry. These two platforms are oneM2M [4, 7], and FIWARE [2]. Actually, a third IoT platform is currently emerging, stemming from the FIWARE NGSI standard [3] , thanks to the effort made by the ETSI ISG CIM workgroup,: NGSI-LD [1]. This new standard is more powerful and flexible, allowing not only the description of context entities but also to define relationships between them, by leveraging the JSON-LD (for Linked Data) ability to express semantic information together with data.

2.2.1 NGSI-LD. Of special interest for this work, NGSI-LD [1] is both an information model and an API. Currently being standardized by the ETSI Industry Specification Group on cross-cutting Context Information Management (ETSI ISG CIM), the Fed4IoT project is actively contributing to its development.

NGSI-LD information model is a meta-model with three main concepts: *entities*, *properties* and *relationships*. The assumption is that the world consists of entities, which can be physical objects like a car or a building, but can also be more abstract things like a company or the coverage area of an WLAN access point. Entities are identified via URI and type (e.g. a car of type `Vehicle`, with identifier `urn:ngsi-ld:Vehicle:A4567`).

Entities have properties, e.g. a `location` or a `speed`, as well as relationships to other entities (e.g. `isOwnedBy` or `isParkedAt`). Furthermore, properties and relationships can be annotated by properties and relationships themselves, making for a powerful and expressive tool capable of capturing a *Property Graph*, whose concepts and relations are represented by the UML diagram in Figure 1.

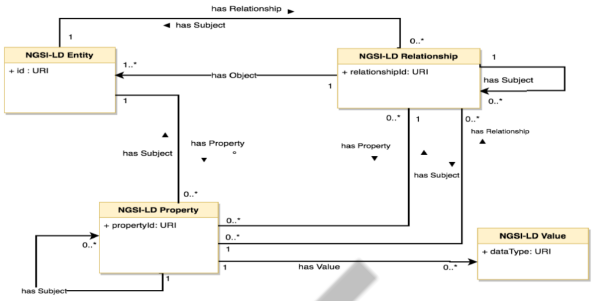

Figure 1: NGSI-LD concepts and relations.

Finally, in addition to the meta model, NGSI-LD defines a cross-domain model, defining some geographic and temporal properties (among other things) and providing the anchor for other ontologies to be linked from/to NGSI-LD.

3 FED4IOT

Cloud Computing decouples infrastructure providers from application providers. The huge proliferation of web, mobile and machine-to-machine applications is undeniably an effect of this decoupling. Small/medium sized application providers can give vent to the invention of new applications by being free from the burden of maintaining and implementing complex ICT infrastructures; thanks to Cloud Computing they can rent them.

While current IoT platforms focus on driving the data from the origin to the destination, current Cloud Computing decouples infrastructure providers from application providers, by using a pool of real ICT devices (servers, network switches, storage, etc.) to offer virtual machines with configurable virtual hardware and operating system (OS). Fed4IoT wants to bring a similar infrastructure/application providers decoupling to the world of IoT, by introducing "Virtual Things". This parallelism is better captured in Figure 2, depicting the **IoT virtualization paradigm** that we are promoting.

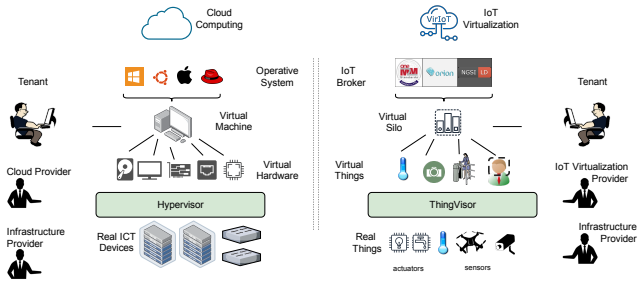

Figure 2: IoT virtualization paradigm.

3.1 VirIoT

Fed4IoT system architecture, named the **VirIoT** platform, uses a federated pool of **Real Things** (i.e. sensors and actuators), offering **Virtual Silos** (similar to a virtual-machine), containing a set of configurable **Virtual Things** (similar to virtual hardware) and an IoT Broker, exposing their data to the user. Virtual things emulate

Real Things, processing data exchanged with real IoT devices, made available through different infrastructure providers. The **ThingVisor** is the piece of software responsible for the processing/virtualization performed, and can connect with Real Things via different technologies, such as: **OneM2M**, **NGSI** and **NGSI-LD**, as well as simple MQTT.

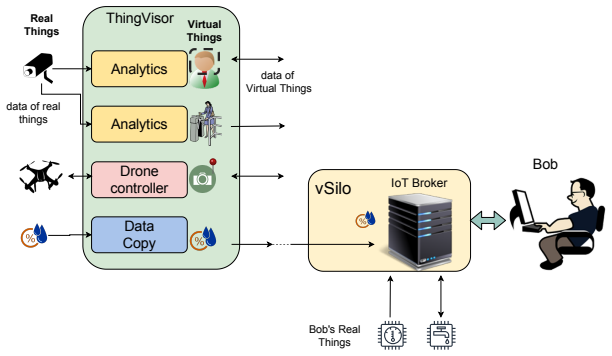

Figure 3: vThings and vSilos.

Figure 3 shows an example of Virtual Things and Virtual Silos, in which face detection and person counting applications (among others) are featured. Virtual Silos are isolated environments, dedicated to running applications for a specific tenant. Tenants can add data coming from the platform's Virtual Things, to his Virtual Silo. Collectively, this data is exposed to the external world through a broker technology of choice (an example of such could be MQTT and FIWARE Orion Broker).

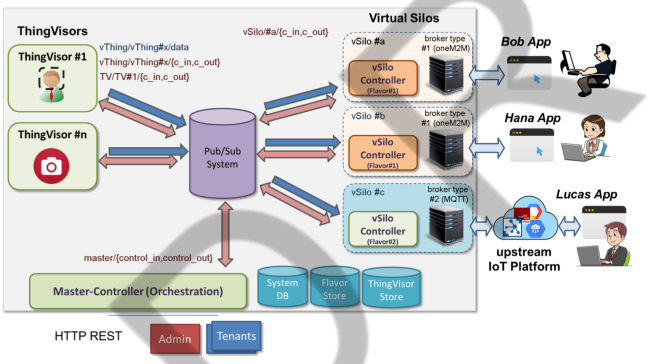

Figure 4: VirIoT system Architecture.

Figure 4 shows a broader view of the full VirIoT system architecture, in which the two remaining central components of the system are depicted: the **Master Controller** and the internal communications **Pub/Sub System**.

The Master-Controller is the main element of the VirIoT control plane: manages deployment, configuration and delivery of new components in the system, following requests from administrator and tenants. To that end, it uses an underlying cloud/edge platform, offering containers-as-a-service, such as Kubernetes. All system state information, about Virtual Silos, Virtual Things, ThingVisors, etc. is stored in a System Database, making the Master Controller a stateless component.

```
{
 "meta": {
  vThing info, such as vThing identifier,
  wether it is an actuator, etc...
 }
 "data": {
  [
   NGSI-LD Entity 1,
   NGSI-LD Entity 2,
   ..
  ]
 }
}
```

Figure 5: Structure of the neutral-format messages

Internal communications in VirIoT, are split into control plane and data planes, both of which are based on MQTT protocol. The reasons behind this architectural decision are manifold, one of the most important being that there are implementations of MQTT that support clustering (e.g. VerneMQ), a feature that suits the distributed cloud/edge infrastructure of VirIoT. The data format of choice for the internal communications in VirIoT, coined "**Neutral Format**" in the project, is based on the NGSI-LD data-model, making it easy to translate to/from the different formats used by IoT Brokers and Sensors respectively. Figure 5 represents the basic structure of a neutral-format message.

Lastly, the *System vSilo* (depicted in Figure 4)is a special kind of Virtual Silo, owned by the administrator of the VirIoT platform. It contains all of the data produced by the platform and its mission is to make it accessible to external systems that may want to connect with VirIoT, effectively enabling the federation of VirIoT systems.

4 FED4IOT SECURITY

Security is a dimension of paramount importance, and as such has been considered in the Fed4IoT project. As a start point, we need to take into consideration that we are dealing with an architecture which has many external interaction points, being:

- `ThingVisors`: receives data from providers, transform and push them into the federated architecture. They can send commands to data root domain too.
- `Virtual Silo`: receives the updated information from ThingVisors and transform it again according to the end-user needs. It can also send commands back to the ThingVisors, upon tenant request.
- `Master-Controller`: offers an API to deploy and configure the system.
- `System Silo`: supports external communications (e.g. with other platforms, for federation).

4.1 Distributed Capability-Based Access Control (DCapBAC)

DCapBAC is the access control mechanism proposed in our architecture. It supports the management of certificates, authentication and

authorization processes. It also offers benefits in terms of distributed management and suports delegation and traceability, among others.

DCapBAC integrates both an Identity Manager (**IdM**), and the eXtensible Access Control Markup Language (**XACML**) framework. The IdM is responsible for storing the identities that can interact with the system as well as its related attributes; which define that identity, such as name, email, organisation or roles (to name but a few). Later, XACML, thanks to its finely-grained defined policies and its processing model (which describes how to evaluate access request according to the previously defined policies) will use those attributes during the authorization evaluation process in order to emit a verdict.

The main advantage presented by DCapBAC, is that it decouples the authorisation process into two different phases: one for requesting access control to a specific resource, and another for accessing it. In this manner, after the requesting phase, the user or service receives authorisation in the form of a Capability Token (**CT**). This CT, which has a limited life-span, is used during the access phase.

4.1.1 Identity Manager. The Identity Manager (IdM) is implemented by FIWARE's Keyrock General Enabler. It offers a complete API for authentication, based on the OAuth 2.0 protocol (described in the RFC6749 standard). This protocol supports several methods for a client application to acquire an access token, including Authorization Codes and user/password schemes among others. An access token can later be used to authenticate requests to API endpoints. An example of such, could be the specific API endpoint of the VirIoT Master-Controller. Following, a list of the different methods of authentication offered is included for completeness:

- `Authorization Code`, for apps running on a web server. The client will redirect the user to the authorization server, and then the user will be asked to login to the authorization server for client approval.
- `Password`, for logging in with the traditional username and password scheme.
- `Client credential`, the simplest of all of the OAuth 2.0 grants, this grant is suitable for machine-to-machine authentication where a specific user's permission to access data is not required.
- `Refresh token`, the access token eventually expires; however, some grants respond with a refresh token which enables the client to get a new access token without requiring the user to be redirected.

The IdM provides functionalities to achieve access within the system, and to manage access privileges. Specifically, the Keyrock Identity Manager implementation defines the following actors and roles:

- `User`. Any signed-up user who is able to identify itself with an email and password. Users rights can be assigned individually, or by group.
- `Application`. Any application consisting of a set of microservices.
- `Organization`. A group of users who can be assigned a series of rights. Altering the rights of the organization affects the access of all users of that organization. Users within an organization can either be members or admins. Admins are able to add and remove users from their organization, while members merely inherit the roles and permissions of an organization. This allows each organization to be responsible for their members and removes the need for a super-admin to administer all rights.
- `Role`. A role is a descriptive bucket for a set of permissions. A role can be assigned to either a single user or an organization. A signed-in user gains all of the permissions from all of her own roles plus all of the roles associated with her organization.
- `Permission`. An ability to do something on a resource (e.g. API endpoint) within the system.
- `X-Subject-Token`. After a user has supplied her username and password, the server returns an access-token in `X-Subject-Token` header.

As authentication's example, Figure 6 shows a *curl* request for obtaining an access-token via the `X-Subject-Token` header, by performing a username and password authentication. This request returns a response with status 201 and the `X-Subject-Token` header containing the string representing the access-token.

```
curl -iX POST 'https://<IDM>:443/v1/auth/tokens' \
  -H 'Content-Type: application/json' \
  -d '{
    "name": "user1@fed4iot.org",
    "password": "fed4iot"
}'
```

Figure 6: IDM authentication request

4.1.2 Authorisation. A classical DCapBAC scenario like that of Figure 7, showcases an entity (subject) willing to access a resource (e.g. an API endpoint) of another entity (target). Usually, a third party (issuer) generates a token for the subject specifying the privileges that it has been given. Then, when the subject attempts to access a resource hosted in the target, it attaches the token which was generated by the issuer. Following, the target evaluates the token, and grants access to the resource. Finally, the subject which wishes to access certain information from the target, sends the token together with the request and the target device that receives such token evaluates the validity of such request based on the token and user information, acting as a Policy Enforcement Point (**PEP**). This simplifies the access control mechanism, and it is a relevant feature in IoT scenarios, since complex access control policies are not required to be deployed on end devices, greatly improving system performance and reliability.

4.1.3 VirIoT integration. Figure 8 shows the integration of this access control technology to the VirIoT platform.

In a schematic way DCapBAC's components are as follows:

- `Security enablers` based on XACML framework: Policy Administration Point (PAP) and Policy Decission Point (PDP). They manage access control policies, and decide who can access a resource and what actions can be performed over it. Policies are represented as triplets (subject, resource, action).

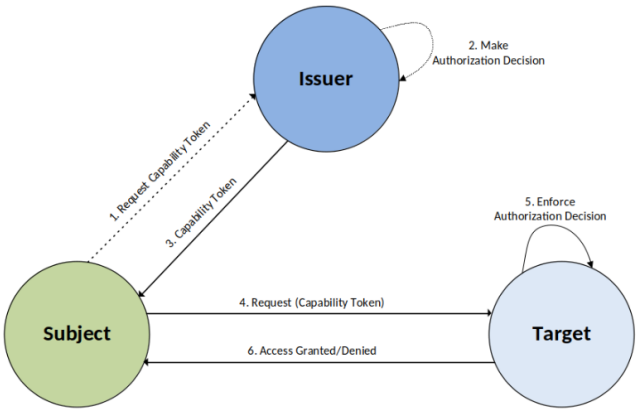

Figure 7: DCapBAC scenario.

- `Capability Manager` (and `Capability Token`). It takes care of:
  - Access to authentication component to validate authentication token (returned by IdM).
  - Access to XACML framework for validating authorization requests.

  This component generates an authorization token, named Capability Token, which is then required to access the resource.
- `PEP-Proxy`, which validates the authorisation. Before performing any action the PEP-Proxy carries out validation, and, if validation is passed, it then forwards the command/message to the Master-Controller.

Under this scheme, tenants (or even administrators), in order to interact with VirIoT for configuration (e.g. create a vSilo, add a vThing, etc.) or maintenance purposes, go through the Identity Manager (IdM) and DCapBAC components, rather than directly accessing the Master-Controller API. The PEP_proxy has full system administration rights, and directly interacts with the Master-Controller, based on the request characteristics and whether they are allowed according to the CT.

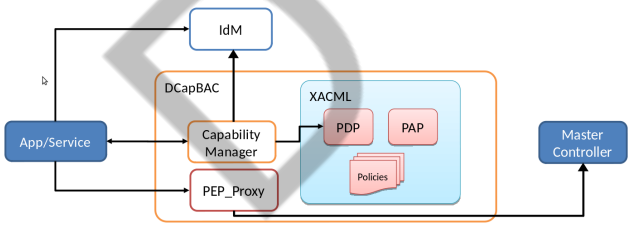

Figure 8: Security Relationships Components.

The sequence diagrams that follow, show the whole interaction that a user/service must perform, in order to be granted access to a specific REST resource of the Master-Controller API. After an authentication process (Figure 9), the User receives an authentication token. This must be later introduced in the authorisation query, addressed to the Capability Manager (Figure 10). This component is responsible for translating this request to XACML, and route it to the XACML-PDP component. It validates the XACML authorisation query by checking its stored XACML policy files. If a matching policy is found, then the XACML-PDP will issue a positive verdict, which is received by the Capability Manager. This, in turn, generates an authorisation token, called Capability Token which is sent to the User.

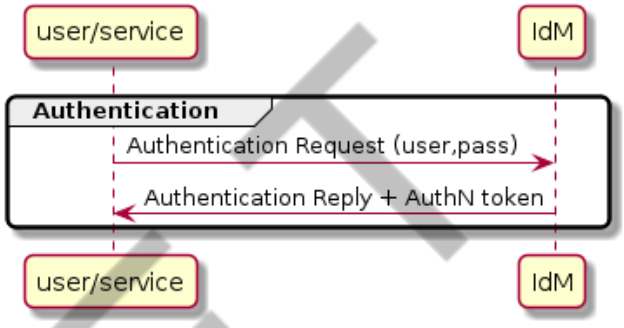

Figure 9: Authentication.

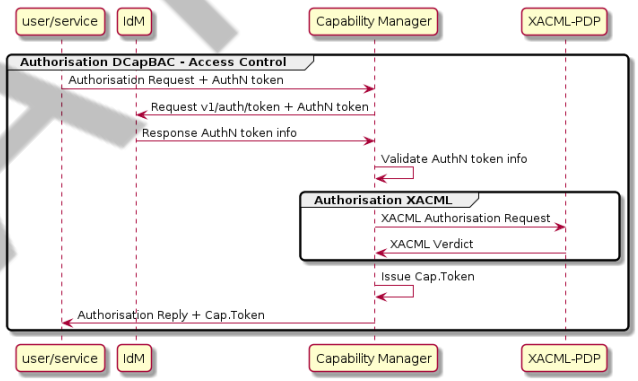

Figure 10: Authorisation.

This Capability Token must accompany the query (Figure 11) issued to the system, so that the PEP_Proxy can verify that the query issued by the User is granted. This process is done without querying any other third party, so it is a straightforward task. After validating the query, the PEP_Proxy will forward the query to the Master-Controller and later will forward back the response to the User.

As PEP_Proxy request example, Figure 12 shows how to access the resource, using a curl request, through the `Cap.Token` obtained through the Capability Manager request. This request will return the response of the specific Master-Controller API.

5 CONCLUSION

The use of Attribute-Based Access Control has proven a reliable and expressive way of defining policies access control in information systems. This is also the case for the VirIoT architecutre of the Fed4IoT Project, where access to the Master Controller can be

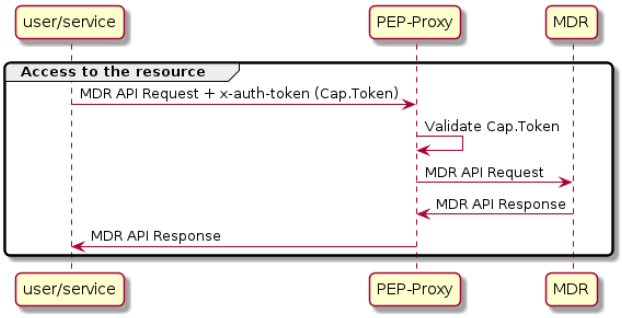

Figure 11: Data access.

```
curl -iX <action> <'pepproxy endpoint><resource>'
    \
-H 'x_auth_token: <capability_token>' \
-d '...'
```

Figure 12: PEP_Proxy request - accessing to specific API

successfully controlled by the application of DCapBAC, successuflly decoupling the evaluation of the access request to a resource, from the actual enforcement, by means of a Capability Token, that can be evaluated in the Policy Enforcement Point without the need of any further contact with the Policy Decision Point.

This approach, not only offers the added expressiveness of ABAC for defining policies, but it also improves on performance and scalability, becoming a great solution for highly dynamic and quickly growing IoT scenarios. In this paper we have shown the different steps required for the authentication, authorization and access to resources of the Master Controller (main control point of the ViRIot architecture) required by DCapBAC, and the integration of the different elements of DCapBAC in the Fed4IoT architecture.

ACKNOWLEDGMENTS

This work is supported in part by H2020 EU-JP Fed4IoT project(www.fed4iot.org, EU contract 814918)

REFERENCES

[1] [n.d.]. *ETSI GS CIM 009. Context Information Management (CIM): NGSI-LD API.* https://docbox.etsi.org/ISG/CIM/Open
[2] [n.d.]. *FIWARE home page.* https://www.fiware.org/
[3] [n.d.]. *OMA, Open Mobile AllianceTM.* http://www.openmobilealliance.org
[4] Soumya Kanti Datta, Amelie Gyrard, Christian Bonnet, and Karima Boudaoud. 2015. oneM2M architecture based user centric IoT application development. In *Future Internet of Things and Cloud (FiCloud), 2015 3rd International Conference on.* IEEE.
[5] Andrea Detti, Giuseppe Tropea, Giulio Rossi, Juan A. Martinez, Antonio F. Skarmeta, and Hidenori Nakazato. 2019. Virtual IoT systems: Boosting iot innovation by decoupling things providers and applications developers. *Global IoT Summit, GIoTS 2019 - Proceedings* (2019), 1–6. https://doi.org/10.1109/GIOTS.2019.8766422
[6] José L Hernández-Ramos, Antonio J Jara, Leandro Marín, and Antonio F Skarmeta. 2013. Distributed Capability-based Access Control for the Internet of Things. *Journal of Internet Services and Information Security* (2013). https://doi.org/10.22667/JISIS.2013.11.31.001
[7] Hyuncheol Park, Hoichang Kim, Hotaek Joo, and JaeSeung Song. 2016. Recent advancements in the Internet-of-Things related standards: A oneM2M perspective. *ICT Express* (2016).