

Ivana Podnar Žarko · Arne Broering  
Sergios Soursos · Martin Serrano (Eds.)

LNCs 10218

# Interoperability and Open-Source Solutions for the Internet of Things

Second International Workshop, InterOSS-IoT 2016  
Held in Conjunction with IoT 2016  
Stuttgart, Germany, November 7, 2016, Invited Papers



# Attribute-Based Access Control Scheme in Federated IoT Platforms

Savio Sciancalepore<sup>1,4</sup>(✉), Michał Pilc<sup>2</sup>, Svenja Schröder<sup>3</sup>, Giuseppe Bianchi<sup>1,5</sup>,  
Gennaro Boggia<sup>1,4</sup>, Marek Pawłowski<sup>2</sup>, Giuseppe Piro<sup>1,4</sup>,  
Marcin Płóciennik<sup>2</sup>, and Hannes Weisgrab<sup>3</sup>

<sup>1</sup> CNIT, Consorzio Nazionale Interuniversitario per le Telecomunicazioni,  
Parma, Italy

<sup>2</sup> Poznań Supercomputing and Networking Center, IBCh PAS, Poznań, Poland  
{michal.pilc,marek.pawlowski,marcin.plociennik}@man.poznan.pl

<sup>3</sup> Cooperative Systems Research Group, University of Vienna, Vienna, Austria

<sup>4</sup> Department of Electrical and Information Engineering (DEI),  
Politecnico di Bari, Bari, Italy

{savio.sciancalepore,gennaro.boggia,giuseppe.piro}@poliba.it

<sup>5</sup> Department of Electronic Engineering,  
University of Rome Tor Vergata, Rome, Italy

giuseppe.bianchi@uniroma2.it

**Abstract.** The Internet of Things (IoT) introduced the possibility to connect electronic things from everyday life to the Internet, while making them ubiquitously available. With advanced IoT services, based on a trusted federation among heterogeneous IoT platforms, new security problems (including authentication and authorization) emerge. This contribution aims at describing the main facets of the preliminary security architecture envisaged in the context of the symbIoTe project, recently launched by European Commission under the Horizon 2020 EU program. Our approach features distributed and decoupled mechanisms for authentication and authorization services in complex scenarios embracing heterogeneous and federated IoT platforms, by leveraging Attribute Based Access Control and token-based authorization techniques.

**Keywords:** Internet of Things · Security mechanisms · Attribute-Based Access Control · Interoperability framework · Macaroons · JSON Web Token

## 1 Introduction

The term Internet of Things (IoT) was first used by Kevin Ashton in 1999 as a name of a network of RFID devices used to monitor corporate supply chains while simultaneously being connected to the Internet [1]. By 2004 the term had been adopted by most scientific and technological journals like Scientific American [2]. Many proposals of IoT platforms like smart housing, smart stadium or even a smart city have been presented since then [3,4]. In parallel, consortia of

enterprises and international institutions started to develop protocols and communication standards more suitable for different deployment cases (including machine-to-machine, body area network, industrial telemetric network, and so on). Now, with the incumbent explosion of the IoT in everyday life, heterogeneous application-specific platforms are emerging, often designed as standalone solutions that hardly communicate with each other.

Unfortunately, the fragmentation of the IoT ecosystem resulted in poor cooperation between different IoT platforms in terms of resource sharing and reusability of applications. In the face of the demand for interoperability between different IoT platforms, several international projects like symbIoTe<sup>1</sup>, INTER-IoT<sup>2</sup> and bIoTope<sup>3</sup> were launched by European Commission under the Horizon 2020 EU program.

Security is an important cornerstone of all those projects which will impact their success or failure in two aspects: usability and technical implementation. Every solution therefore needs to protect the privacy of users and its resources against unauthorized access and must still provide full functionality.

In distributed (but interoperable) IoT networks, for instance, the protection of resources against unauthorized accesses and the authentication of users requires more sophisticated methods. Conventional computer networks adopt the Role-Based Access Control (RBAC) paradigm. In RBAC a user is assigned a role such as “*administrator*” or “*ordinary user*” that predetermines access rights policies. Unlike RBAC, the Attribute-Based Access Control (ABAC) method of authorization derived from distributed computing relies on the assignment of so-called “*attributes*” to each entity in the system. An attribute may refer either to a user or to a particular resource or to the surrounding environment. An “attribute” is defined as a particular property, role or permission associated to a component in the system. It is assigned after an authentication procedure by the system administrator [5].

In this paper we present a security architecture that enables an ABAC-based controlled access to IoT resources and easily supports a trusted resource sharing among different IoT platforms. The proposed security architecture was developed in the symbIoTe project, which aims at a symbiosis of smart objects across IoT environments. The main contributions of the work are summarized below:

- we describe general requirements for a secure and standardized interoperability framework;
- we identify components to be deployed in the system to manage security issues;
- we provide a baseline architecture for the authentication and authorization among federated IoT platforms;
- we identify different scenarios that require customized security functionalities;
- we design interfaces and interactions among components in the aforementioned architecture;

---

<sup>1</sup> <https://www.symbiote-h2020.eu>.

<sup>2</sup> <http://www.inter-iot-project.eu>.

<sup>3</sup> <http://biotope.cs.hut.fi>.

- we propose two possible technical solutions for the token format, that are Macaroons and JSON Web Tokens (JWTs).

The rest of the paper is organized as in the following.

Section 2 presents security requirements for the IoT interoperability framework and Sect. 3 outlines the state-of-the-art in distributed systems and IoT. Section 4 describes the architecture of the proposed system with focus on security aspects. Following the requirements and architecture, two possible token solutions are presented and compared in Sect. 5. Our efforts, contributions and future work are summarized in Sect. 6.

## 2 Requirements

In a network of federated IoT platforms it is necessary to provide security of applications, components, platforms and resources at high level. Such complex system must be protected against many security threats: opening doors by an illegitimate user in a Smart Home environment, reading confidential data from remote sensors connected to an industrial sensor network or launching evacuation mechanism in Smart Stadium environment. Due to a huge number of low-power devices, IoT networks are vulnerable to many types of attacks that can cause substantial damages. First of all, every device that is in the radio range can overhear the transmission between sensors, actuators and other devices, thus each message must be protected by cryptographic protocols. Secondly, many IoT devices are not susceptible against malicious updates of their firmware, which can cause damages in a computer network the IoT platform is connected to for instance, letting hackers break into e-mail account<sup>4</sup>. Finally, IoT platforms and networks of IoT platforms are vulnerable to distributed denial-of-service (DDoS) and man-in-the-middle (MITM) attacks. With nearly 20 billion devices that will have been connected to the Internet by 2020 it is crucial to mitigate security threats.

After a detailed analysis of use case sample scenarios, like Smart Home, we defined security requirements of the system. At application layer it is necessary for all devices to adhere to the rules defined within OWASP Internet of Things Project<sup>5</sup>. One of the most important requirement concerning the system is the provision of mutual authentication: this means that not only a user must authenticate with an IoT platform but also an IoT Platform must authenticate with a user. This can help to protect the network against impersonating whole services or servers together with two factor authentication (password and PIN). Authentication and authorization are implemented with cryptographic protocols and primitives. Communication between all system entities must be encrypted to prevent illegitimate access to resources and tampering the data. Another important security mechanism is the validation of input data which is based on

---

<sup>4</sup> <http://iotsecurityconnection.com/posts/security-is-a-must-in-everyiot-device>.

<sup>5</sup> [https://www.owasp.org/index.php/OWASP\\_Internet\\_of\\_Things\\_Project](https://www.owasp.org/index.php/OWASP_Internet_of_Things_Project).

sanitization mechanisms. The countermeasure consists in eliminating potentially harmful characters from user input by means of removal, replacement and encoding the characters. Another important security requirement in such a framework is the implementation of access control through access policies. The users and entities must be able to define constraints that limit the users that are allowed to get access to resources, time constraints and other attribute. The major difficulty here is the capability of granting access to resources from one platform while being a user logged in another IoT platform. To solve this problem, the system must offer identification mechanisms for instance through tokens that store attributes. Another feature that must be implemented in application layer are mechanisms of establishing trust relationships and thus implicitly trust levels, prior to applying security mechanisms for the first time. Information about this must be stored in a secure data store i.e. by Public Key Infrastructure (PKI). Finally, privacy at user level must be preserved. All data about sensors, entities and other resources that are exposed by IoT platform must be anonymized, i.e. devices must not expose details on manufacturer, firmware version and other sensitive data. The details about sensor location must not be exposed unless its owner agrees on it.

### 3 State of the Art

Guaranteeing user authentication and authorization in distributed computing systems has been always regarded as a concern.

Authorization methods that rely on attributes were widely applied in cloud computing systems, where security policies are supported by the authorization mechanisms of the cloud [6]. More recently a commercial solution of security architecture specifically designed for Supervisory Control and Data Acquisition (SCADA) systems was presented in [7].

A decentralized network of federated IoT platforms like our approach resembles the aforementioned scenarios. The core part of our design is a cloud responsible for seamless connection between sensors, actuators and user applications placed in different IoT platforms. Trust management concerns about the Internet-of-Things were summarized in [8].

A related work showing security threats in IoT was published in 2015 by Sicari et al. [9].

In 2014, the concept of macaroon tokens for decentralized authorization in the cloud was presented [10]. A different authorization method, widely adopted in online purchasing, that is JWT, was described in [11]. Recently, a recommendation for authentication and authorization in IoT was issued in the Authentication and Authorization for Constrained Environments (ACE) IETF Working Group [12]. Scalability and adaptation of access control policies to the environment conditions were proposed as a solution for the Internet of Things [13].

Security concerns were also addressed in EU-funded projects under the 7th Framework Programme (FP7) like OpenIoT<sup>6</sup>, SMARTIE<sup>7</sup>, RERUM<sup>8</sup>, COMPOSE<sup>9</sup> and FI-WARE<sup>10</sup>.

To enable a baseline comparison, COMPOSE and OpenIoT are two FP7 projects in which authentication and authorization issues have been tackled by using a centralized solution. In COMPOSE, the middleware is the owner of all registered resources, and manages centrally the authentication of client applications and the issuing of tokens. In OpenIoT, instead, same mechanisms are offered by a Central Authorization and Authentication Service (CAS). This architecture property allows those projects to apply the well-known OAuth 2.0 authorization framework [14]. In our case, instead, the ownership over resources is left to each IoT platform, thus making impossible to apply the OAuth 2.0 paradigm. However, its decoupled logic have been used as a useful starting point for the development of our solution. Finally, while most of the aforementioned projects use ABAC for access control (SMARTIE, RERUM, FI-WARE), COMPOSE uses RBAC and openIoT Lattice-Based Access Control (LBAC). While JSON is being used for issuing tokens in some of the projects, none of them so far considered using macaroons for tokens. However, none of them included attributes directly in the token.

## 4 Architecture

The reference architecture considered in this contribution is depicted in Fig. 1. It integrates many independent IoT platforms exposing heterogeneous resources. Each IoT platform (thus, each available resource) is registered with a trusted mediator (i.e., the interoperability framework's *core*) which offers advanced mechanisms for enabling platform interoperability and distributed resource access. Moreover, there are applications willing to access the available resources.

To maximize interoperability among platforms our security framework has to deal with different scenarios: applications can be registered to only the trusted mediator, to only one IoT platform, or two (or more) IoT platforms federated with the mediator entity. Therefore, the following target scenarios can be identified:

- **Scenario #1:** an application is registered with an IoT platform and it would like to access resources exposed by the IoT platform where it is registered to. This is the case of a typical cloud system, where applications, services and resources are controlled by the same administrator, without the need to interface with other platforms.

---

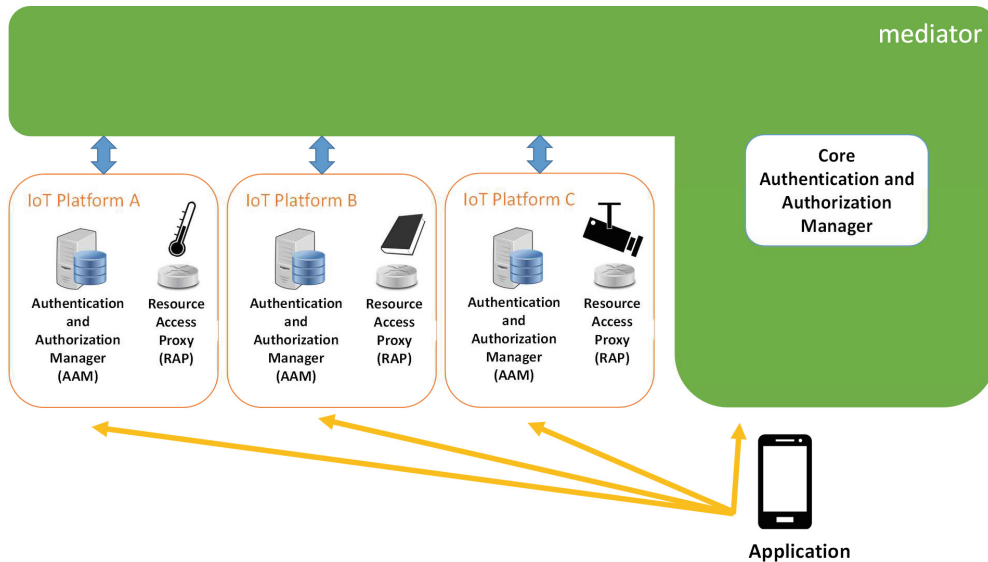
<sup>6</sup> <http://www.openiot.eu>.

<sup>7</sup> <http://www.smartie-project.eu>.

<sup>8</sup> <https://www.ict-rerum.eu>.

<sup>9</sup> <http://www.compose-project.eu>.

<sup>10</sup> <https://www.fiware.org>.



**Fig. 1.** System architecture.

- **Scenario #2:** an application is registered with the trusted mediator and it would like to access resources exposed by a federated IoT platform. This is the case of a third-party application developer, that implements special applications to access services and resources exposed by a given IoT platform controlled by a different administrator.
- **Scenario #3:** an application is registered with one or more IoT platforms federated with the mediator and it would like to access resources exposed elsewhere in the considered architecture. This is the case of a current sensor in a smart home. To access the data, the system could require an application to register both with Smart-Home and the Service Provider IoT platforms. We refer to this scenario as the *multi-domain access rights composition* paradigm.

#### 4.1 Main Security Rationale

The resource access is handled through the ABAC logic. ABAC is a well-known technique for dealing with access control in distributed environments, which is able to protect sensitive data, applications or services from unauthorized operations by means of efficient, simple and flexible access rules. It is based on *attributes* and *access policy* concepts.

An attribute encodes a specific property, role or permission assigned to an application. Attributes are stored within a digital object, namely a *token*, that certifies the authenticity of both the issuer (i.e., a dedicated component of mediator or IoT platform) and the owner (i.e., the application), additionally to its time validity. Both symmetric or asymmetric cryptography techniques can be used to ensure authenticity and integrity of those tokens. To provide few examples, Macaroons use symmetric keys to generate an Hash-based Message Authentication Code (HMAC) that assures integrity and authenticity of the token, while

JWTs can be created both by using symmetric or asymmetric keys. A thorough description of both solutions will be provided in Sect. 5.

An access policy, instead, enables a fine-grained access control mechanism. In fact, it describes the combination of attributes needed to obtain the access to a given resource. For each resource, a dedicated access policy can be defined. Then, an application in possession of tokens storing a set of attributes matching the aforementioned access policy can successfully obtain the access to the resource. Otherwise, its access request will be denied.

It emerges that the token represents a key element in the resource access mechanism. From the security perspective, it is generated during the authentication procedure and inspected and validated during the authorization procedure. The solution described in this contribution natively offers the decoupling between authentication and authorization processes. This means that authentication and authorization involve different components and are independently executed at different times. An application uses the authentication procedure to authenticate itself within a given domain (like the trusted mediator or an IoT platform federated with the mediator). In case of a successful authentication it obtains a set of tokens storing its own attributes. Then, the collected attributes can be used during the authorization procedure to obtain access to resources. Since an application should not perform the whole authentication process for each resource access, the designed approach allows also for enhanced flexibility and scalability benefits for the whole system.

Note that when an application or component registered in a given IoT platform or in the mediator would like to access resources exposed elsewhere, it could be possible that the attributes that are assigned to it are not valid also in the new domain. Therefore, an *Attributes Mapping Function* is needed to manage the translation between attributes in different platforms. Thanks to the described functionalities, at the same time the interoperability framework works on top and extends the existing security architecture of a given IoT platform, providing procedures for secure communications with foreign IoT platforms and third-party applications.

## 4.2 Component Description

To practically implement the aforementioned security rationale in each target scenario, the following logical components are introduced:

**Platform Authentication and Authorization Manager (AAM):** With reference to **Scenario #1** and **Scenario #3**, it handles the authentication procedure for applications registered with the IoT platform. Therefore, it releases *home* tokens storing attributes that describe properties, roles and/or permissions assigned to the application within the platform where it is registered to. It also manages a Token Revocation List (TRL) storing the list of tokens that have been revoked before their expiration. For this reason it may be contacted by any component in the architecture during the *check revocation procedure*.

With reference to **Scenario #2** and **Scenario #3**, the Platform AAM is in charge of (i) verifying the validity and the possible revocation of tokens generated by the AAM component of the mediator or the AAM component of another IoT platform, (ii) performing the attributes mapping functionality and (iii) releasing a new set of tokens, namely *foreign tokens*, usable in the local IoT platform.

**Core AAM:** With reference to **Scenario #2**, it handles the authentication procedure for applications registered with the mediator. Therefore, it releases *core* tokens storing attributes that describe properties, roles and/or permissions assigned to the application at the mediator side. Moreover, similarly to the Platform AAM, it manages the TRL and can be contacted by any component in the architecture during the *check revocation procedure*.

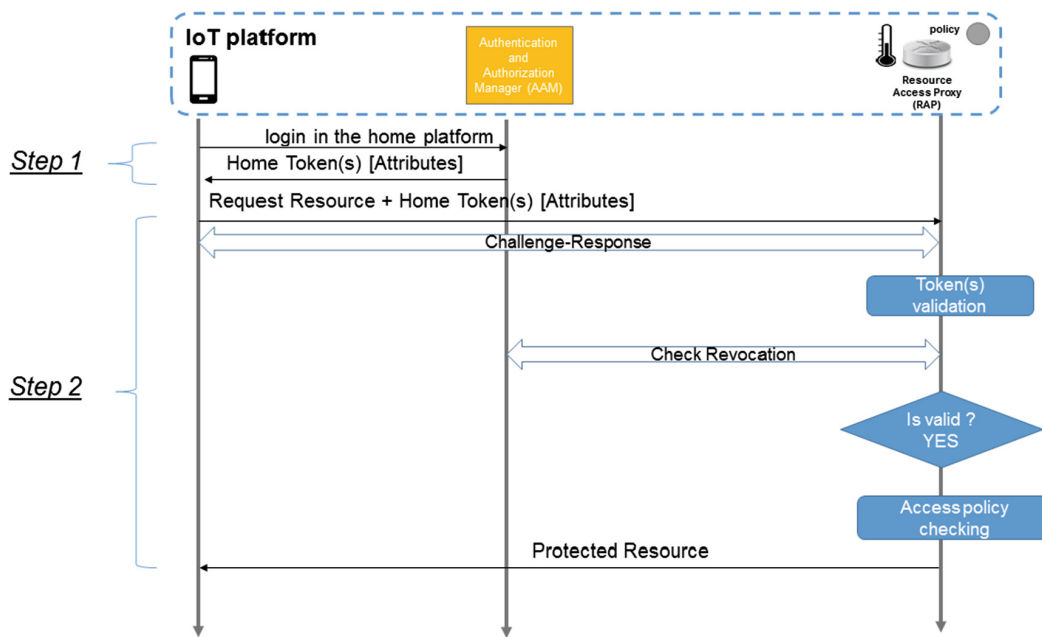
**Resource Access Proxy (RAP):** With reference to **Scenario #1**, **Scenario #2** and **Scenario #3**, it holds the URI for obtaining all the resources available in the specific IoT platform, along with the access control policy associated to each of them. Therefore, it receives all the requests for accessing these resources along with the tokens containing the attributes of the requester. Finally, it enforces the access control by checking if the provided attributes satisfies the policy associated with the resource. In the positive case it provides access to the resource itself, otherwise the access is denied.

Note that each AAM component manages authentication and authorization functionalities only for users and applications that are registered within its domain. In this sense, third-party applications that are not registered with any reference IoT platform registers within the core AAM. This design choice provides enhanced flexibility to the interoperability framework, because each AAM does not need to store data related to all possible components in the system. Instead each AAM stores only a limited amount of data. When foreign components tries to access to local services, the reference AAM can interact with the remote AAM to obtain the required information about the application. Finally, we highlight that specific modules devoted to the detection and prevention of system anomalies could be envisaged to work in strict connection with the described modules. However, their design is left for future work.

### 4.3 Sequence Diagrams

The reference sequence diagrams for the scenarios introduced at the beginning of the section will be provided in what follows. We suppose a previous registration phase, in which each component in the system receives an asymmetric key pair (private/public keys) and a public-key certificate in a trusted way. The public key of the application is also included in the token, to guarantee authenticity of the token itself and to provide the cryptographic material to be used in the challenge-response procedure. Finally, end-to-end security in the communication between described components is guaranteed thanks to the mandatory use of the Transport Layer Security (TLS) protocol [15].

The sequence diagram describing the resource access in **Scenario #1** is depicted in Fig. 2. It is composed by two main steps:



**Fig. 2.** An application registered in a given IoT platform wants to access resources produced within its home IoT platform.

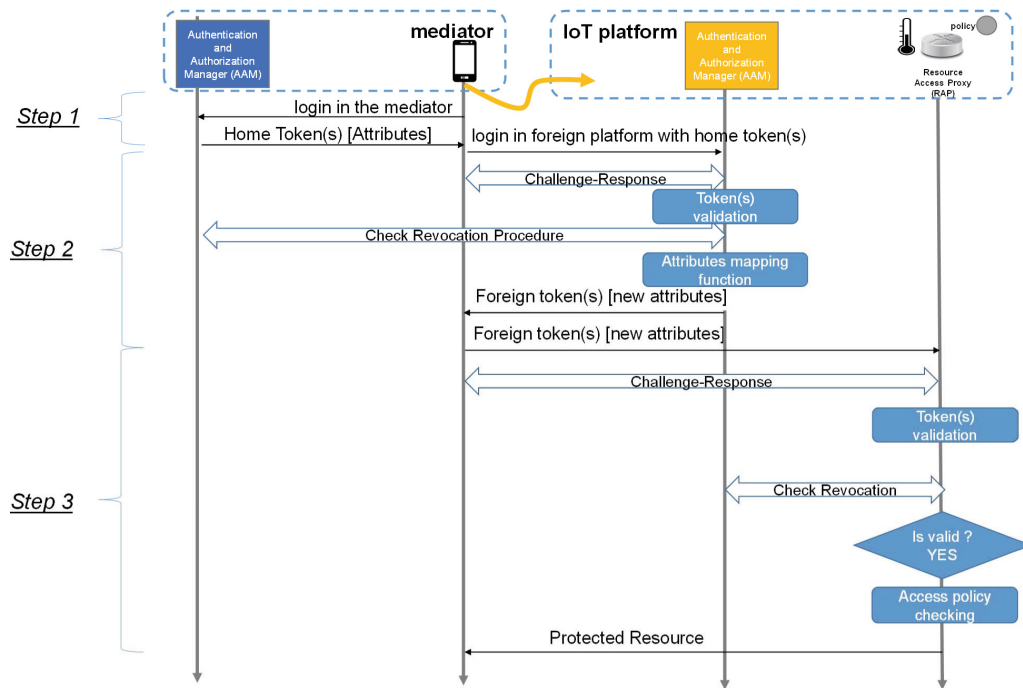
**Step 1: Home authentication.** At the beginning, the application performs the login in its home platform by contacting the home AAM and receiving home tokens.

**Step 2: Resource access authorization.** The application contacts the RAP and delivers the home tokens retrieved in the previous step. The RAP initiates the challenge-response mechanism to verify that the application is the real owner of the tokens. In the case the challenge-response mechanism is successfully completed, the RAP verifies that tokens are valid and that they have not been revoked by contacting its reference platform AAM component. Then it checks the provided attributes against the access policy associated with the requested resource: if the attributes supplied by the applications are enough to satisfy the access policy associated with the resource (according to the ABAC logic) the RAP grants the access to the resource. Otherwise access is denied.

The sequence diagram describing the resource access in **Scenario #2** is depicted in Fig. 3. It is composed by three main steps:

**Step 1: Core authentication.** At the beginning, the application performs the login with the mediator by contacting the core AAM and receiving core tokens.

**Step 2: Foreign authentication.** The application forwards core tokens to the AAM component of the foreign platform. The AAM component of the foreign platform initiates the challenge-response mechanism to verify that the application is the real owner of the tokens, thus preventing both replay and impersonation attacks. In the case the challenge-response mechanism is successfully completed, the AAM component of the foreign platform validates the



**Fig. 3.** An application registered in the mediator space wants to access resources in an IoT platform.

tokens, verifies that they have not been revoked by contacting the AAM component of the mediator and performs the attribute mapping function. Then, it generates a new set of foreign tokens and sends them to the application.

**Step 3: Resource access authorization.** The application contacts the RAP and delivers it the foreign tokens retrieved in the previous step. The RAP initiates the challenge-response mechanism to verify that the application is the real owner of the tokens. In the case the challenge-response mechanism is successfully completed, the RAP verifies that tokens are valid and that they have not been revoked by contacting its reference platform AAM component. Then it checks the provided attributes against the access policy associated to the requested resource: if the attributes supplied by the applications are enough to satisfy the access policy associated to the resource (according to the ABAC logic) the RAP grants the access to the resource. Otherwise the access is denied.

The sequence diagrams describing the resource access in **Scenario #3**, referring to as *multi-domain access rights composition*, are depicted in Fig. 4. Without loss of generality, it is assumed that the application is registered in platforms *IoT\_A* and *IoT\_B* and would like to gain access to a resource available in the platform *IoT\_C*. Also in this case, the procedure has three main steps:

**Step 1: Home authentications.** At the beginning the application performs the login in the IoT platforms where it is registered to (i.e., *IoT\_A* and *IoT\_B*). To this end it contacts the AAM component of each platforms for retrieving home tokens.

**Step 2: Foreign authentication.** The application combines the home tokens and forwards them to the AAM component of the foreign platform *IoT\_C*. The AAM component of the foreign platform initiates the challenge-response mechanism to verify that the application is the real owner of the tokens, thus preventing both replay and impersonation attacks. In the case the challenge-response mechanism is successfully completed, the AAM component of the foreign platform validates the tokens, verifies that they have not been revoked by contacting AAM components of the home platforms (i.e., *IoT\_A* and *IoT\_B*) and performs the attribute mapping function. Then, it generates a new set of foreign tokens and sends them to the application.

**Step 3: Resource access authorization.** The application contacts the RAP and delivers the foreign tokens retrieved at the previous step. The RAP initiates the challenge-response mechanism to verify that the application is the real owner of the tokens. In the case the challenge-response mechanism is successfully completed, the RAP verifies that the tokens are valid and that they have not been revoked by contacting its reference platform AAM component. Then it checks the provided attributes against the access policy associated with the requested resource: if the attributes supplied by the applications are sufficient to satisfy the access policy associated to the resource (according to the ABAC logic) the RAP grants access to the resource. Otherwise, the access is denied.

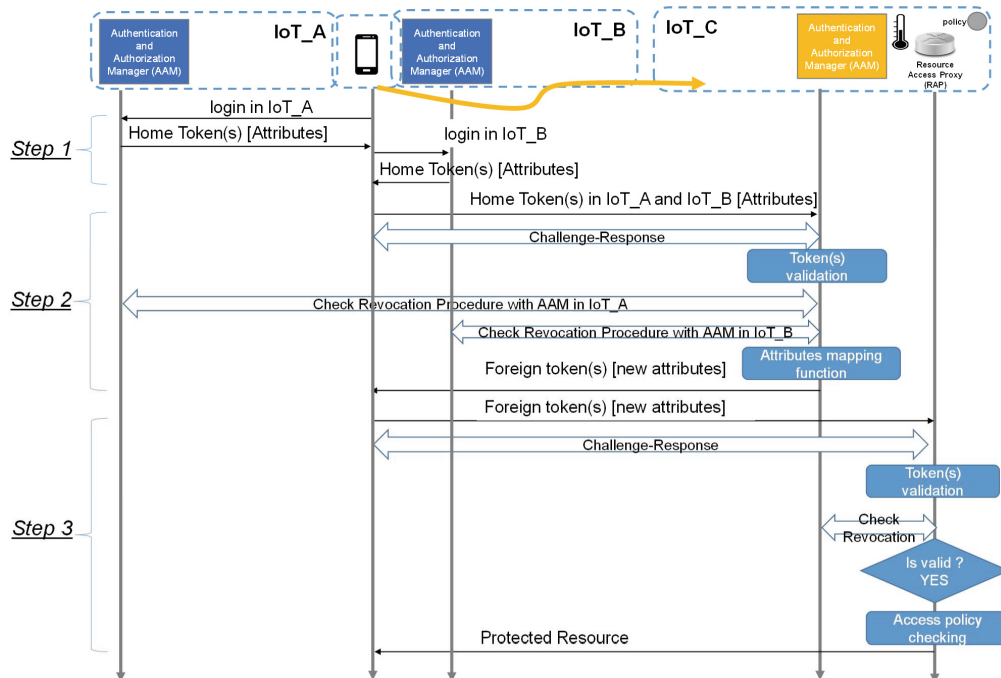


Fig. 4. Multi-domain access rights composition.

## 5 Planned Implementation

The implementation of the security architecture described in Sect. 4 requires the selection of a suitable token format. Without loss of generality a token is a digital object used as a container for security-related information. It serves for authentication and/or authorization purposes and generally appears as a list of elements. Each element contains an assertion that further specifies properties assigned to the owner of the token. Each token must contain an explicit expiration date, indicating the date until the token can be considered valid. Moreover, the token also contains at the end an element that certifies its authenticity and integrity. Depending on the chosen solution, validating a token could require different procedures.

The following discussion describes two promising technical solutions, candidates for the implementation of tokens in our approach. It does not provide only a comparison between the two approaches, but it illustrates also, specifically in Sect. 5.3, how they can be modified in order to fit within the proposed architecture.

During implementation of the framework itself one or a combination of the following technologies will be used. We aim at a flexible solution where platforms, applications and other use cases can decide which of the following technologies they want to use.

### 5.1 Macaroons

Macaroons are a new kind of authorization credential developed by Google [10]. As bearer credentials, they serve a similar purpose as cookies in World Wide Web, but they are more flexible and provide better security. Macaroons are based on a construction that uses nested, chained MACs (e.g., HMAC) in a manner that is highly efficient, easy to deploy and widely applicable. They allow authority delegation between bearers with attenuation and contextual confinement. Each field embedded within macaroons structure, i.e. the caveat, restricts both the macaroons' authority and the context in which it may be used (e.g. by limiting the permitted actions and requiring the bearer to connect from a certain IP address and to present additional evidence such as a third-party signature). Macaroon caveats are plain-text readable. Macaroons also contain a list of *AND* conditions. Its bearer is authorized to perform an operation *AS LONG AS condition1, condition2, ..., conditionN* hold true. The main (root) macaroon which allows for everything gets successively attenuated with those conditions. Each of them is signed with an HMAC function.

By considering the reference architecture shown in Fig. 1, three possible tokens can be introduced: *root macaroons*, *platform macaroons*, and *application macaroons*. The mediator creates a root macaroon by calculating the HMAC function of a random nonce and its secret key. Note that the output of the HMAC function must be shared among AAMs of federated platforms for verifying the authenticity of tokens received during the resource access procedure. Starting from the root macaroon the mediator also generates platforms macaroons. The

platform macaroons are signed by the HMAC function with the key being the previously calculated HMAC value. Then, each platform can autonomously generate application macaroons by following the same process.

An application, after obtaining the macaroon from the AAM component of its platform, may further attenuate it and therefore authorize others to perform actions in its name. For example a Smart Home System mints its owner a full access token. The owner can go on vacation and want the neighbor next door to be able to operate the windows and doors but not the garage. The owner can do so by attenuating his/her own token and handing it to the neighbor.

## 5.2 JSON Web Tokens

JWT is an open industry standard widely used in today's Internet to deal with authentication and authorization issues [11]. It contains a set of *claims*. A claim is a specific certified statements related both to the token itself or to the entity that is using it. Typically, these claims are encoded in the JavaScript Object Notation (JSON) format, thus easily allowing system interoperability. A claim is identified with a specific name: it is possible to distinguish between *Registered Claim Names*, that are names defined and standardized in the reference document and *Private Claim Names*, that represent extensions that a developer could choose for his/her own system.

The cryptographic force of the JWT resides in the sign field, stored at the end of the token. It can be generated through symmetric or asymmetric cryptography techniques and allows to verify the authenticity of the token, i.e. generation by a trusted entity, as well as integrity, in the sense that no one could modify its content without invalidating it.

Each JWT contains a header that provides information about the type of the token and the algorithm used to build the sign of the token. It contains also a body, encoding a set of claims for this token, and finally - a sign containing the cryptographic validation of the token and generated as stated in the header.

Registered Claim Names carried in the body of the JWT include *iss*, that uniquely identifies the entity that issued the token, *sub*, which uniquely identifies the entity for which this token has been released (it is a key field when a token needs to be used also for authentication purposes), *exp*, indicating the expiration time, after which this token should not be used and processed by any entity in the system; *nbf*, that identifies the time in which this token becomes effectively valid and can be processed by any entity in the system, *iat*, identifying uniquely the time in which this token has been created and, finally, *jti*, that is the unique identifier of the token.

JWT fits perfectly within the reference architecture described in Sect. 4. From a cryptographic perspective, the only requirement for its adoption is the deployment of a public-key infrastructure, which issues a private/public key pair to each entity in the system.

AAMs uses their private keys to generate and sign tokens. Any entity in the system that receives a token could easily verify its authenticity by gathering the public key of the issuer of the token (specified in the token itself).

Important features such as the support for an expiration date are integrated by JWT thanks to the definition of the *exp* claim.

Also, each token can be easily associated to a given entity in the system through the *sub* claim. More in detail, the public key of the owner of the token can be embedded in this claim. This can be used in the challenge-response procedure described in Sect. 4 to prove the possession of the respective private key and verify that the application using the token is effectively the entity for which the token has been generated. This procedure avoids replay attacks.

Finally, the JWT can be easily extended to support the carrying of attributes associated to the ABAC logic, thanks to the possibility to integrate customized Private Claims.

### 5.3 Usage of Tokens in Our Proposal

To conclude, the comparison between the user macaroon and the user JWT, as they can be modified to be included in the described architecture, is reported in Fig. 5.

nonce
access all
<b>time constraints</b> for platform-token
<b>sign_aam_id</b> = $S(PV_{root}, iot-a)$
<b>time constraints</b> for user-token
<b>list of attributes</b> $[A_1, A_N]$
<b>user certificate</b> (user id)
<b>sign_user_id</b> = $S(PV_{iot-a}, user\_id)$
<b>sign_user</b> = $HMAC_{platform-token}(token)$

(a)

<b>Header: Alg: RSA/ECDSA Typ: JWT</b>
<b>jti</b> = id
<b>iss</b> = iot-a
<b>sign_iss</b> = $S(PV_{root}, iot-a)$
<b>iat</b> [value]
<b>nbf</b> [value]
<b>exp</b> [value]
<b>sub</b> = user certificate
<b>att</b> = list of attributes
<b>token_sign</b> = $S(PV_{iot-a}, token)$

(b)

**Fig. 5.** Details of the content of (a) Macaroons and (b) JWT tokens for the designed architecture.

The application macaroon is shown in Fig. 5(a). It has a hierarchical structure. The nonce is used as a unique identifier of the token. The second, third and fourth caveats are related to the AAM of the platform in which the application is registered to. In particular, the ID of the AAM is signed through the private key of the mediator entity ( $PV_{root}$ ). The remaining lines are dedicated to the application. They contain the list of attributes assigned to the application, its public key certificate and, finally, the sign on the user ID by the AAM that issued the token, through its private key ( $PV_{iot-a}$ ). The last line is needed to

assure that the AAM is the unique component able to sign the token. Finally, the last caveat is the chained HMAC of all the caveats in the token, performed starting from the output of the HMAC function of the root macaroon.

The application JWT is shown in Fig. 5(b). Also in this case we can identify the part related to the issuer of the token, certified through the sign with the private key ( $PV_{root}$ ), and the part related to the owner of the token. The private claim *att* is introduced to encode the information about the list of attributes possessed by the application. Finally, the sign of the whole token is performed through the private key of the issuer ( $PV_{iot-a}$ ) without the need of a symmetric shared secret.

Note that both token types have a limited time validity. After the expiration of that date, the token must be renewed through a new authentication procedure.

The evaluation of pros and cons of macaroons tokens and JWTs as well as their suitability for the proposed scenario is left for future work.

## 6 Conclusions and Future Activities

In this paper we presented the baseline security architecture for an interoperability framework among IoT platforms, developed within the H2020 EU project symbIoTe. First, we described our general system requirements derived from use cases and requirements. We illustrated our approach for a standardized IoT architecture with the focus on security. Current plans for implementation foresee the usage of the ABAC paradigm, through Macaroons or JWT tokens. However, this paper so far provides a basic architecture and a proposal for some technical solutions to realize a security framework. Future work will aim first at implementing the proposed security architecture within the H2020 symbIoTe project, and conducting a deep performance evaluation of the two approaches described for tokens. Also, a solution for detecting anomalies in the system will be developed and device-level security will be carefully considered. Our main action point for future work, however, will be on the implementation of the aforementioned secure interoperability framework (and thus validation of the architecture concept).

**Acknowledgments.** This work is supported by the H2020 symbIoTe project, which has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No. 688156. The authors would like to cordially thank the entire symbIoTe consortium for their valuable comments and discussions.

## References

1. Ashton, K.: That Internet of Things thing. *RFID J.* **22**, 97–114 (2009)
2. Gershenfeld, N., Krikorian, R., Cohen, D.: The Internet-of-Things. Technical report, *Scientific American* (2004)
3. Gross, M.: Smart house and home automation technologies. Technical report, *Encyclopedia of Housing* (1998)

4. Mohanty, S.P., Choppali, U., Kougianos, E.: Everything you wanted to know about smart cities. *IEEE Consum. Electron. Mag.* **5**(3), 60–70 (2016)
5. Hu, V., Ferraiolo, D., Kuhn, R., Schnitzer, A., Sandlin, K., Miller, R., Scarfone, K.: Guide to Attribute Based Access Control (ABAC) definition and considerations. NIST special publication 800-162. NIST, January 2014
6. Khan, A.: Access control in cloud computing environment. *ARPN J. Eng. Appl. Sci.* **7**(5), 613–615 (2012)
7. Juniper-Networks: Architecture for secure SCADA and distributed control system networks. Juniper Networks White Paper (2010)
8. Yan, Z., Zhang, P., Vasilakos, A.: A survey on trust management for Internet of Things. *J. Netw. Comput. Appl.* **42**, 120–134 (2014)
9. Sicari, S., Rizzardi, A., Grieco, L., Coen-Porisini, A.: Security, privacy and trust in Internet of Things: the road ahead. *Comput. Netw.* **76**, 146–164 (2015)
10. Birgisson, A., Gibbs Politz, J., Erlingsson, U., Lentzner, M.: Macarons: cookies with contextual caveats for decentralized authorization in the cloud. In: *Proceedings of the Conference on Network and Distributed System Security Symposium* (2014)
11. Jones, M., Bradley, J., Sakimura, N.: JSON Web Token (JWT). RFC 5719, IETF, May 2015
12. Seitz, L., Selander, G., Wahlstroem, E., Erdtman, S., Tschofenig, H.: Authorization for the Internet of Things for constrained environments draft-ietf-ace-oauth-authorization-04. Internet draft, IETF (2016)
13. Hennebert, C., et al.: IoT governance. privacy and security issues. Technical report, European Research Cluster on the Internet of Things, January 2015
14. Hardt, D.: The OAuth 2.0 authorization framework. RFC 6749, IETF, October 2012
15. Dierks, T., Rescorla, E.: The transport layer security protocol Version 1.1. IETF, April 2006