Attribute-Based Access Control scheme in federated IoT platforms

Savio Sciancalepore[1,4], Michał Pilc[2], Svenja Schröder[3], Giuseppe Bianchi[1,5], Gennaro Boggia[1,4], Marek Pawłowski[2], Giuseppe Piro[1,4], Marcin Płóciennik[2], Hannes Weisgrab[3]

[1]CNIT, Consorzio Nazionale Interuniversitario per le Telecomunicazioni
[2]Poznań Supercomputing and Networking Center, IBCh PAS, Poznań, Poland; e-mail: {name.surname}@man.poznan.pl
[3] University of Vienna, Cooperative Systems Research Group, Vienna, Austria
[4] Department of Electrical and Information Engineering (DEI), Politecnico di Bari, Bari, Italy; e-mail: {name.surname}@poliba.it.
[5] Department of Electronic Engineering, University of Rome Tor Vergata, Rome, Italy; e-mail: giuseppe.bianchi@uniroma2.it

Abstract. The Internet of Things (IoT) paradigm rapidly changed how people think about and experience the Internet. During the inevitable evolution of the Internet landscape, in fact, IoT introduced the possibility to connect electronic things from everyday life to the Internet, while making them ubiquitously available. Now, the new research trend is following the development of advanced services, based on a trusted federation among heterogeneous IoT platforms. In this context, however, a new set of security problems (including authentication and authorization) emerges. The aim of this contribution is to describe the main facets of the preliminary security architecture envisaged in the context of the symbIoTe project, recently lunched by European Commission under the Horizon 2020 EU program. At the time of this writing, the developed solution already offers distributed and decoupled mechanisms for the provisioning of authentication and authorization services in complex scenarios embracing many, heterogeneous, and federated IoT platforms. Specifically, they leverage Attribute Based Access Control and token-based authorization techniques. Thus, the work will provide a step-by-step description of security functionalities in three different target scenarios and will suggest, at the end, some promising technical implementations that can be taken into account in the hereafter of the symbIoTe project.

Keywords: Internet of Things, Security mechanisms, Attribute-Based Access Control, interoperability framework, Macaroons, JSON Web Token, symbIoTe

1 Introduction

Since its invention Internet changed many times. Originally, in the 1970s, it was invented to connect mainframe computers. In the middle of 1980s, the invention

of the IP protocol allowed thousands of desktop computers to join the Internet. The idea of a ubiquitous network, where all devices are connected each other, world-spanning network and can be controlled remotely was presented in 1991 by Mark Weiser in Scientific American [1], and led to the development of the World Wide Web.

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 [2]. By 2004 the term had been adopted by most scientific and technological journals like Scientific American [3]. Many proposals of IoT platforms like smart housing, smart stadium or even a smart city have been presented since then [4][5]. 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. They leverage different communication technologies (i.e. RFID, Bluetooth, ZigBee, BACnet, IEEE 802.15.4), lightweight protocols provided by Internet Engineering Task Force (IETF) standardization efforts [6], and new application protocols, including Message Queuing Telemetry Transport (MQTT) and Constrained Application Protocol (CoAP).

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[1], INTER-IoT[2] and bIoTope[3] 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 require 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

[1] `https://www.symbiote-h2020.eu`
[2] `http://www.inter-iot-project.eu`
[3] `http://biotope.cs.hut.fi`

a component in the system. It is assigned after an authentication procedure by the system administrator [7].

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;
- 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 Section 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 Section 5. Our efforts, contributions and future work are summarized in Section 6.

2 Requirements

Security system mechanisms in software are a crucial part of large and complex systems, where attackers can get critical information about users or have the possibility to remotely control critical functions, like opening doors in a smart home environment or launch the evacuation mechanism in a smart stadium, to name a few. The following requirements have been derived from a subset of IoT use cases (smart home, smart ecological routing, smart stadium, etc.), from existing IoT platform solutions and finally from knowledge about security and privacy in software projects we obtained from the OWASP Secure Coding Practices [8].

The most fundamental and the most important security mechanisms are authentication (proving the identity of a user) and authorization (granting a user access rights to resources). Another important security mechanism is the validation of input data. This validation is based on sanitization mechanisms which for example eliminate potentially harmful characters from user input by means of: removal, replacement, encoding and escaping the characters.

Based on those findings, we derived the first and most important security requirements for a federated IoT network, which are mechanisms for the authentication and authorization of entities/actors i.e., users/application developers, IoT platforms, developed applications and clients. Another important security requirement in such a framework is the implementation of access control in the system, defined through access policies. The major function here is the capability of getting access to resources in one platform while being a registered user in a different one. To solve this problem the framework must offer a user identification methodology, for example through tokens (this paper contains the description of two different technology approaches which would be suitable).

Another feature which must be implemented in a federated IoT network in the application layer are mechanisms of establishing trust relationships - and thus implicitly trust levels - prior to applying security mechanisms for the first time. Information about thais must be stored in a secure data store, e.g. by Public Key Infrastructure (PKI). Two-factor authentication (if the underlying platform supports it, e.g. authenticate by using password and PIN) is one of the solutions considered to increase the level of security at the application level. Impersonating users or devices in the ecosystem is not the only potential attack against infrastructure/system: impersonating whole services or servers, is more problematic. To prevent this type of attacks mutual authentication must be supported by all security mechanisms. This means that not only the user/application/software/... must be authenticated against the platform but also vice versa, in order to facilitate malicious platform detection.

From a privacy perspective, one of the the most important requirements is the possibility to let users/entities choose where their data is being used and processed. Those users/entities must be able to modify the privacy parameters regarding their data. Another crucial security mechanism is the support of encrypted data communication between all involved entities in the application and core level (e.g. applications, platforms).

Apparently, large and complex systems like federated IoT networks pose even more demands in terms of security requirements. Unfortunately, listing them all would extend the scope of this publication. Thus, in the rest of this contribution we focused on the most important ones, that were identified while analyzing IoT platforms requirements.

3 State of the Art

Guaranteeing user authentication and authorization in distributed computing systems has been always regarded as a concern. In 1993 Woo and Lam proposed a logical approach for representing and evaluating authorization, independently of implementation [9]. In this seminal paper they postulated separating security requirements of a distributed computer network from the implementation and defined subject attributes referring to the location, in addition to other attributes. Following their ideas Foster et al. defined in [10] a security policy for large-scale distributed computing environments and presented the correspond-

ing security architecture, where they included a user proxy to avoid transferring user credentials. Moreover, they mapped attributes between a local subnet and the global network [10].

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 [11]. More recently a commercial solution of security architecture specifically designed for Supervisory Control and Data Acquisition (SCADA) systems was presented in [12].

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 [13].

A related work showing security threats in IoT was published in 2015 by Sicari et al. [14].

In 2014, the concept of macaroon tokens for decentralized authorization in the cloud was presented [15]. A different authorization method, widely adopted in online purchasing, that is JWT, was described in [16]. Recently, a recommendation for authentication and authorization in IoT was issued in the Authentication and Authorization for Constrained Environments (ACE) IETF Working Group [17].

Security concerns were also addressed in EU-funded projects under the 7th Framework Programme (FP7) like SMARTIE[4], RERUM[5], ARMOUR[6] and COSMOS[7]. Scalability and adaptation of access control policies to the environment conditions were proposed as a solution for the Internet of Things [18].

4 Architecture

The reference architecture considered in this contribution is depicted in Figure 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 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 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 the trusted mediator and it would like to access resources exposed by an IoT platform federated with the mediator;

[4] `http://www.smartie-project.eu`
[5] `https://www.ict-rerum.eu`
[6] `https://team.inria.fr/eva/h2020-armour`
[7] `http://iot-cosmos.eu`

– **Scenario #2**: 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;
– **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 scenario also refers to the *multi-domain access rights composition* paradigm.

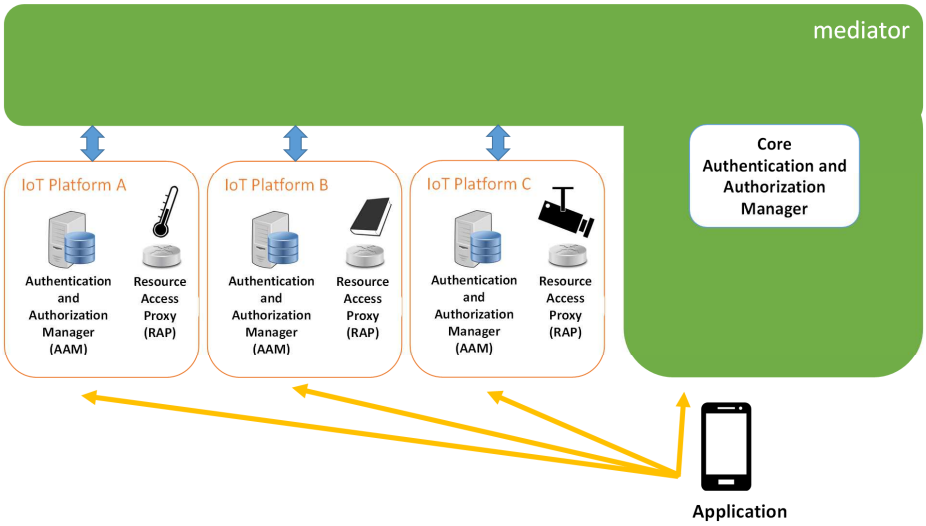


Fig. 1. System Architecture.

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.

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.

4.2 Component description

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

Core Authentication and Authorization Manager (AAM): With reference to **Scenario #1**, 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. 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*.

Platform AAM: With reference to **Scenario #2** 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. Moreover, similarly to the Core AAM, it manages the TRL and can be contacted by any component in the architecture during the *check revocation procedure*.

With reference to **Scenario #1** and **Scenario #3**, it 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.

Resource Access Proxy (RAP): With reference to **Scenario #1**, **Scenario #2** and **Scenario #3**, it holds the URL 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.

4.3 Sequence Diagrams

The sequence diagram describing the resource access in **Scenario #1** is depicted in Figure 2. 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 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 diagram describing the resource access in **Scenario #2** is depicted in Figure 3. It is composed by two main steps:

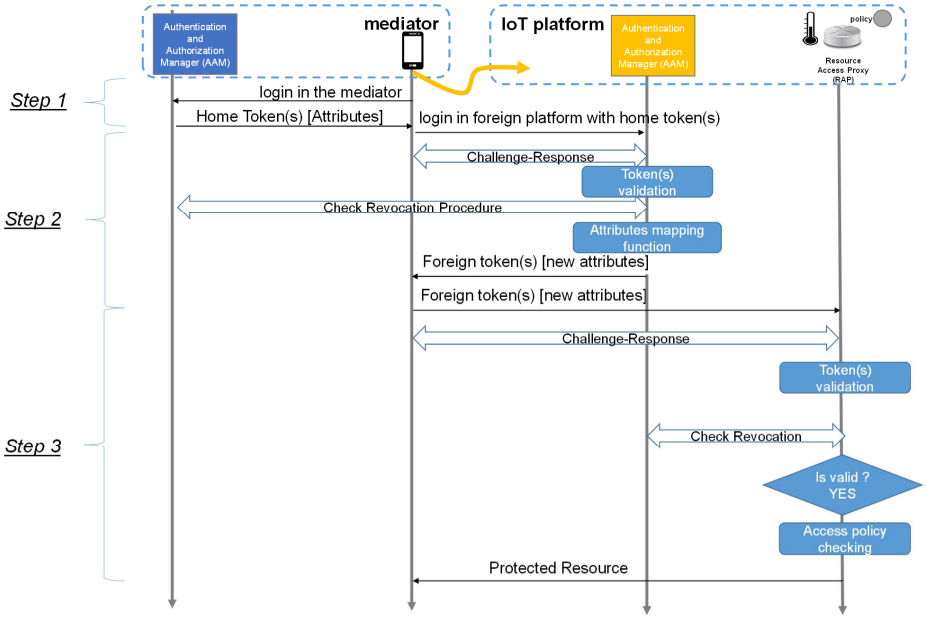


Fig. 2. An application registered in the mediator space would like to access resources in an IoT platform.

Step 1: Home authentication. At the beginning, the application performs the login in it's 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 diagrams describing the resource access in **Scenario #3**, referring to as *multi-domain access rights composition*, are depicted in Figure 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

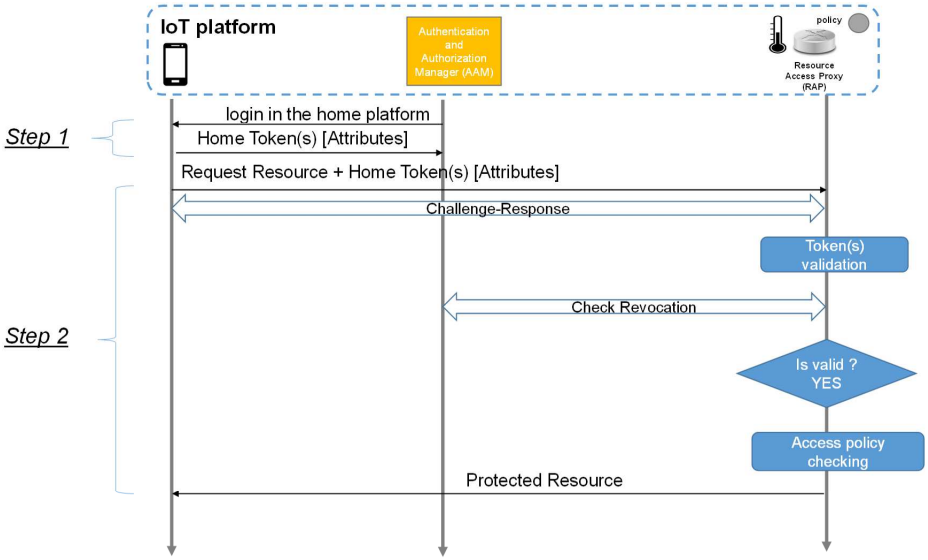


Fig. 3. An application registered in a given IoT platform wants to access resources produced within its home IoT platform.

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.

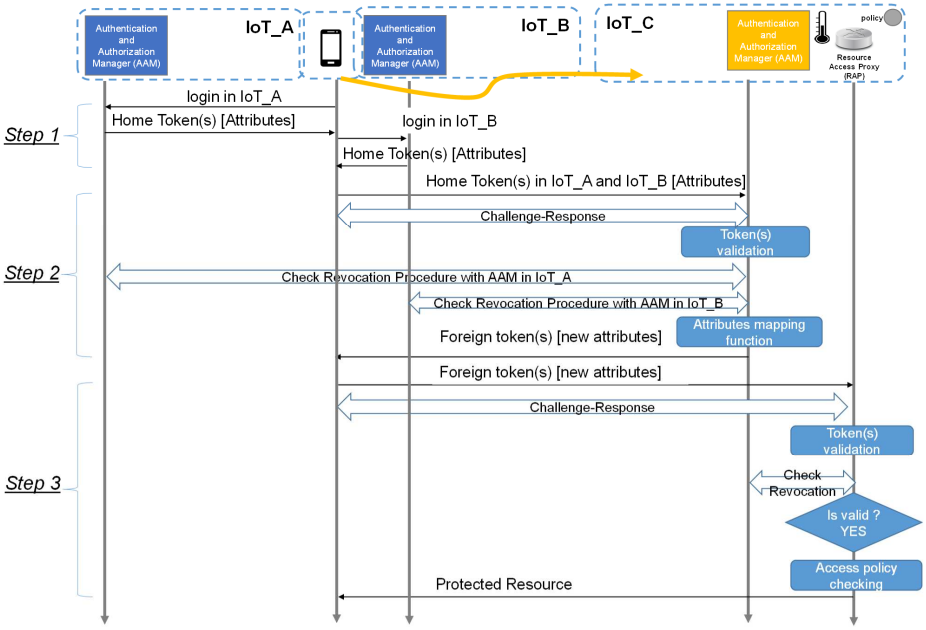


Fig. 4. Multi-Domain Access Rights Composition.

5 Planned implementation

The implementation of the security architecture described in Section 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.

In what follows we describe two promising technical solutions, candidates for the implementation of tokens in our approach. 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 [15]. 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., Hash-based Message Authentication Code (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 Figure 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 [16]. 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 Section 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.

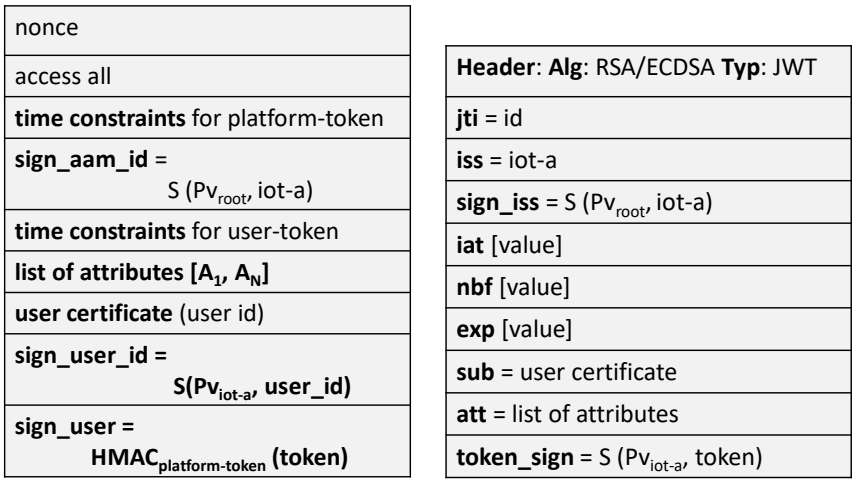
Also, each token can be easily associated to a given entity in the system through the *sub* claim. More in detail, the public key of the owner of the token can be embedded in this claim. This can be used in the challenge-response procedure described in Section 4 to prove the possession of the respective 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 Comparison

To conclude, the comparison between the user macaroon and the user JWT is reported in Fig. 5.

The application macaroon is showed in Figure 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



(a) (b)

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

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 showed in Figure 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 similar projects. Based on those 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.

Future work will aim at designing the system to be interoperable with a broad variety of heterogeneous IoT platforms. So far our project has focused on a certain range of IoT platforms as reference use cases, but we aim at developing interoperability IoT security architecture, which can also be applied to further use cases.

Additionally, in the context of the developing IoT market, usability increasingly becomes a sales argument. Especially with secure systems interfaces need to have a high usability to avoid human error during usage. Future challenges include thorough usability testing of the developed solutions and improvements of the security mechanisms to achieve a proper usable security experience.

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), the review of the architecture and the requirements.

7 Acknowledgements

This work was framed in the context of the project SymbIoTe, which receives funding from the European Union's Horizon 2020 research and innovation programme under grant agreement 688156.

References

1. Weiser, M.: The computer of the 21st century. Technical report, Scientific American (1991)
2. Ashton, K.: That Internet of Things Thing. RFID Journal (2009)
3. Gershenfeld, N., Krikorian, R., Cohen, D.: The Internet-of-Things. Technical report, Scientific American (2004)
4. Gross, M.: Smart House and Home Automation Technologies. Technical report, Encyclopedia of Housing (1998)
5. Mohanty, S.P., Choppali, U., Kougianos, E.: Everything you wanted to know about smart cities. IEEE Consumer Electronics Magazine **5**(3) (Jul. 2016) 60–70
6. Palattella, M., Accettura, N., Vilajosana, X., Watteyne, T., Grieco, L., Boggia, G., Dohler, M.: Standardized Protocol Stack for the Internet of (Important) Things. IEEE Commun. Surveys Tuts (2012)
7. 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 (Jan. 2014)
8. OWASP: Secure Coding Practices. . Quick reference guide, ver 2.0. (Nov. 2010)
9. Woo, T.Y.C., S.Lam: Authorization in Distributed Systems: a new approach. Journal of Computer Security (1993)
10. Foster, I., Kesselman, C., Tsudik, G., Tuecke, S.: A Security Architecture for Computational Grids. In: Proc. of the 5th ACM Conf. on Comp. and Commun. Security. (1998) 83–92
11. Khan, A.: Access Control in Cloud Computing Environment. ARPN Journal of Engineering and Applied Sciences **7**(5) (2012)

12. Networks, J.: Architecture for secure SCADA and distributed control system networks. White paper (2010)
13. Yan, Z., P.Zhang, A.Vasilakos: A survey on trust management for Internet of Things. Journal of Network and Computer Applications **42** (2014) 120–134
14. Sicari, S., Rizzardi, A., Grieco, L., Coen-Porisini, A.: Security, privacy and trust in internet of things: The road ahead. Computer Networks **76** (2015) 146 – 164
15. Birgisson, A., Gibbs Politz, J., Erlingisson, U., Lentczner, M.: Macaroons: Cookies with Contextual Caveats for Decentralized Authorization in the Cloud. In: Proc. of the Conf. on Network and Distributed System Security Symposium. (2014)
16. Jones, M., Bradley, J., Sakimura, N.: JSON Web Token (JWT). RFC 5719, IETF (May. 2015)
17. Seitz, L., Selander, G., Wahlstroem, E., Erdtman, S., Tschofenig, H.: Authorization for the Internet of Things for Constrained Environments draft-ietf-ace-oauth-authz-02. Internet draft, IETF (Jun. 2016)
18. : IoT Governance, Privacy and Security Issues. Tech. rep., European Research Cluster on the Internet of Things (Jan. 2015)