Hindawi
Wireless Communications and Mobile Computing
Volume 2017, Article ID 3859836, 13 pages
https://doi.org/10.1155/2017/3859836

Research Article

An ARM-Compliant Architecture for User Privacy in Smart Cities: SMARTIE—Quality by Design in the IoT

V. Beltran, A. F. Skarmeta, and P. M. Ruiz

Department of Information and Communication Engineering, University of Murcia, Murcia, Spain

Correspondence should be addressed to V. Beltran; vicbelma@gmail.com

Received 26 April 2017; Accepted 6 August 2017; Published 10 September 2017

Academic Editor: Lampros Lambrinos

Copyright © 2017 V. Beltran et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Much has been said about the benefits that the Internet of Things (IoT) will bring to citizens' life. Countless smart objects will be soon offering autonomous behavior in smart environments by sensing the physical world around us, collecting information about us, and taking proactive actions (many times without our consent) with the ultimate goal of improving our wellness. Without a strong guarantee on user privacy, the IoT may sound scary for many citizens. Indeed, the IoT-Architecture Reference Model (IoT-ARM) is a European effort for promoting IoT quality aspects such as security and privacy. This paper paves the way to the adoption of reference architectures by describing the application of the IoT-ARM within a European-funded project, SMARTIE. The SMARTIE architecture has been designed to empower citizens to take control of their IoT devices and privacy, while guaranteeing scalability for large deployments in smart cities.

1. Introduction

Millions of smart objects will be around us soon in what we call smart homes, smart buildings, and smart cities [1]. For citizens, smart environments will bring ubiquitous innovative services that will make their everyday life easier and improve their wellness and even their health. However, the ubiquitous and autonomous nature of Internet of Things (IoT) devices has made the debate on user privacy hotter than ever. These devices many times do not expose user interfaces for privacy configuration and collect and share user data without users being aware of this. The benefits of the IoT will not be maximized if citizens perceive their privacy in peril and hence neglect to take part of IoT services. Nevertheless, the risk of losing citizens' trust on the IoT is not seen by IoT verticals that focus on accomplishing their application-specific goals, leaving important quality aspects such as security undefined or poorly applied. Video cameras that are left open to online viewing, Internet-connected automobiles that are hacked on highways, and automatic unlock mechanisms for homes that grant unauthorized access [2] are just some examples. Having witnessed the harmful consequences of cyberattacks in the Internet, one can easily imagine the serious threats of smart cities with millions of connected IoT nodes; some of them controlling critical infrastructures for transport, security, and health. To meet the expectations on the IoT, it is imperative to address its challenges and maximize its benefits while reducing its risks. Common consensus on security and other quality aspects is needed in heterogenous and interconnected smart cities. In this regard, in order to promote quality aspects of IoT platforms, the European Union (EU) has invested efforts on several FP7-programme-funded projects in the last few years [3]. Notably, the IoT-Architecture (IoT-A) project started in 2010 to develop a Reference Architecture and finally released the IoT-Architectural Reference Model (IoT-ARM) [4] in 2012. The ultimate goal of IoT reference architectures is to consolidate an IoT ecosystem for engineers to work under the frame of well-stated quality-of-service aspects. Likewise, stakeholders can rely on reference architectures as a framework that guarantees the quality of compliant IoT platforms and enables their comparison.

This paper introduces the IoT-ARM to readers through a real use case, the generation of the platform developed by the SMARTIE EU-funded project. Given the large and interconnected documentation on the IoT-ARM, this paper is intended to help readers to understand the benefits of this Reference Architecture and how they can generate their IoT platforms based on its specifications. SMARTIE is an

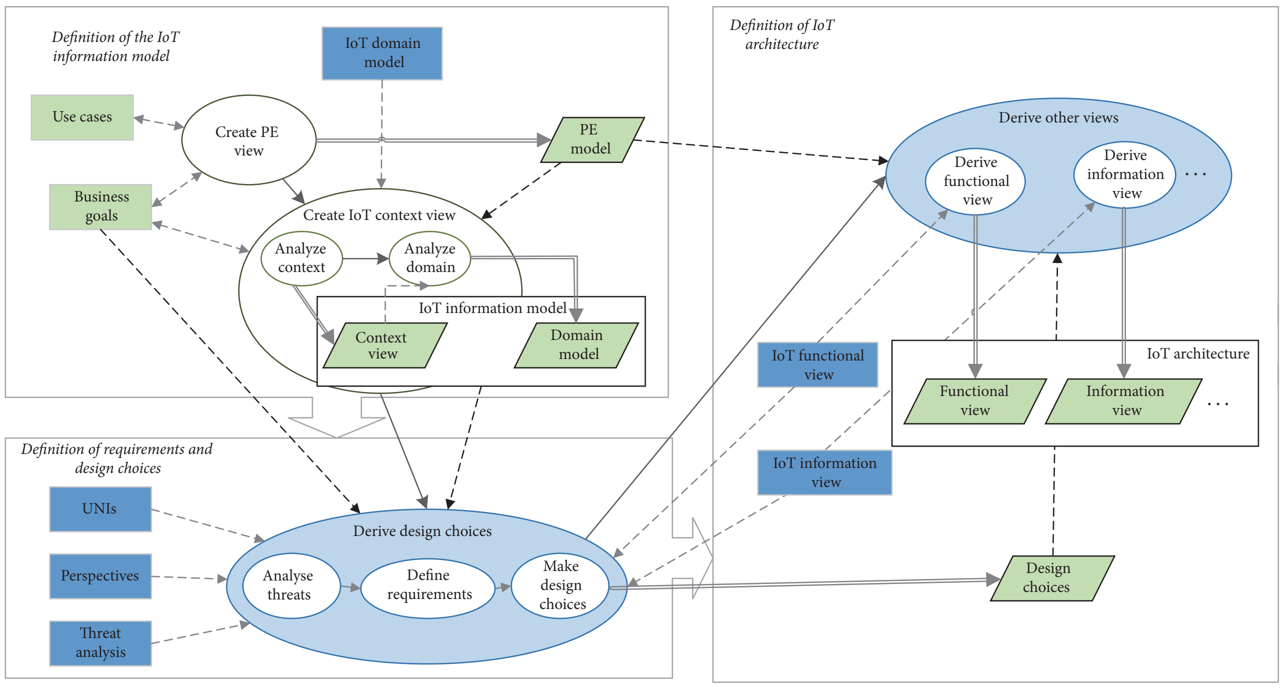


Figure 1: Main steps in the ARM-complaint architecture-generation process. Solid arrows indicate the flow of control, being "Create PE View" the starting process. Transparent circles are application-dependent processes and hence not specified by the IoT-ARM. Blue-colored circles are processes modeled by the IoT-ARM. Dashed arrows indicate inputs and outputs. Blue rectangles represent IoT-ARM inputs and green parallelograms the outputs of processes.

integrating IoT platform that supports the secure and efficient dissemination of IoT data in smart cities. Scalability and user privacy have therefore been the two major quality perspectives for the SMARTIE platform. Scalability is provided by the distribution and decentralization of most of the SMARTIE functionality. Privacy is guaranteed by Functional Components that allow citizens to be in control of the disclosure of their sensitive data: decentralized policy-based access control, encryption, and secure device bootstrapping.

This paper is organized as follows. Section 2 describes the IoT-ARM. Section 3 illustrates the ARM-compliant architecture-generation process through the SMARTIE platform. Section 4 introduces the SMARTIE architecture and describes the main components for user privacy. Section 5 describes the interaction between the main Functional Components of SMARTIE for a particular use case. Section 6 introduces some related work, and finally Section 7 gives some conclusions.

2. The Basics of the IoT-ARM

The IoT-ARM was conceived as an abstract and application-independent reference framework in order to support the generation process of IoT architectures in any IoT domain. Thus, the IoT-ARM defines high-level concepts, semantics, and functions that are common to any IoT platform. It is composed of two main blocks [4]: the IoT Reference Model and the IoT Reference Architecture. The former constitutes a general information model for the IoT that architects can use as the foundation for their application-specific information model. The latter serves as the basis and guidance for the design and derivation of concrete IoT architectures. The rationale behind the IoT-ARM is the development of IoT platforms that satisfy the stakeholders' concerns on quality aspects. To this end, IoT architectures are formed by architectural views that are designed to accomplish well-defined qualitative requirements. The IoT-ARM relies on perspectives to represent qualitative aspirations on (a) evolution and interoperability, (b) availability and resilience, (c) trust, security, and privacy, and (d) performance and scalability.

Figure 1 outlines the three main phases involved in the generation of an IoT-Architecture based on the IoT-ARM: the definition of the IoT Information Model, the definition of design choices, and lastly the definition of the IoT-Architecture. As it can be seen in the last phase, an IoT-Architecture is a composition of architectural views. Architects take the high-level architectural views defined in the IoT Reference Architecture as a reference to design their own views representing their particular application's requirements. The following subsections introduce the fundamental IoT-ARM blocks that support the three phases outlined in Figure 1.

2.1. The IoT Reference Model.

The IoT Reference Model (from now on RM) provides a common understanding of the IoT domain for any IoT platform. The RM provides three basic submodels for the architecturing process: the IoT Domain Model, the IoT Information Model, and the IoT Functional Model. Architects rely on these three models to roughly define the functionality and information flows that their IoT platform should provide.

The IoT Domain Model forms the basis for the rest of submodels by providing a common taxonomy of the main IoT concepts and their relationships [5]. This common taxonomy represents the backbone of the information model of any specific IoT domain. As depicted in Figure 1, the IoT Domain Model is the main IoT-ARM's input for the definition of any application's information model.

In the IoT Domain Model, there are seven core concepts: Physical Entity (PE), Virtual Entity (VE), Augmented Entity (AE), User, Device, Resource, and Service. A PE is any physical object that is relevant from a user or application perspective. An AE is a combination of a PE and its digital representation, that is, its VE. In a typical IoT scenario, VEs are associated with Resources that reflect the state of the related PEs (e.g., the temperature resource of a temperature sensor). Services can expose Resources and can be associated with VEs. Users can interact physically with PEs and digitally with Services. Indeed, a User can be either a person or a software agent.

The IoT Information Model gives more details about VEs (i.e., relations, attributes, and services) at a conceptual level. Thus, this model can be seen as an augmentation of the information provided by the IoT Domain Model.

The IoT Functional Model is an abstract framework for understanding the main Functionality Groups (FGs) of any IoT-Architecture and their interactions.

2.2. IoT-ARM Requirement Process. The IoT-ARM takes a quality by design approach by defining a requirement process as a previous step to the generation of any particular architecture. This process, which is depicted by the second phase of Figure 1, results in a set of choices for the design of the IoT-Architecture. It relies on three main sources of high-level requirements and design choices: Unified Requirements (UNIs), Perspectives, and Threat analysis. Architects can rely on these components to derive or instantiate their concrete design choices as follows.

2.2.1. Perspectives and Tactics. The IoT-ARM links qualitative perspectives to a set of abstract tactics that should be followed to accomplish the perspective's quality properties. Architects have to translate perspectives' tactics to concrete design choices for their system architecture. To support this critical task, the IoT-ARM gives a set of design choices for the architectural views impacted by perspectives' tactics.

2.2.2. Unified Requirements. The IoT-ARM defines a set of UNIs that are formulated on a high abstraction level in order to be applied to any potential domain-specific IoT system. Each UNI is associated with relevant information such as the driving (high-level) business goal, involved concepts of the IoT Domain Model, and the impacted components of the IoT Reference Architecture (i.e., architectural views and FGs). On one hand, architects can take UNIs as the basis to instantiate requirements for their particular needs. On the other hand, through UNIs' associations, architects can easily identify the components that are impacted by UNIs and hence should be especially considered. Moreover, some UNIs are associated with perspectives, thereby allowing architects to explore the quality aspects that should be addressed for these UNIs and the design choices that would help satisfying these UNIs.

2.2.3. Threat Analysis. The IoT-ARM provides a Threat analysis that assesses common risks for any IoT system. This analysis on one hand concludes a set of mitigating design choices and, on the other hand, can serve as an inspiration to define concrete requirements for any particular architecture.

2.3. The IoT Reference Architecture: Architectural Views. Architectural views represent system aspects that can be conceptually isolated, namely, the PE View, the IoT Context View, the Functional View, the Information View, and the Deployment and Operation View.

The PE View identifies the physical entities that will be central for the IoT system. The IoT Context View represents, on one hand, how the system interfaces to the outside world and, on the other hand, the domain model of the system. These two views are not defined by the IoT-ARM since they are use-case-dependent. The Functional View provides a set of Functional Components (FCs) for each FG identified in the IoT Functional Model. The Information View describes how information is handled and exposed by FCs as well as the information flows between them. The Deployment and Operation View is a set of guidelines to help architects to realize concrete systems based on their defined IoT-Architecture.

3. Generation of the SMARTIE Architecture

This section outlines how the SMARTIE platform's architecture has been developed based on the IoT-ARM. We introduce the four most important building blocks for generating an IoT-Architecture (see Figure 1) from our experience: the definition of business goals and use cases, the design of the domain model, the derivation of design choices, and the definition of the architecture's Functional View. We refer the reader to [4] for further information on the ARM-compliant architecture-generation process.

3.1. Business Goals and Use Cases. The definition of the IoT system's use cases and business goals is the starting point for architects. SMARTIE defines several uses cases for smart cities such as intelligent public transportation, traffic management, and energy management and safety in smart buildings [6]. The core business goal of SMARTIE is "*to enable the efficient and secure dissemination of data in smart cities based on a user-centric privacy- and security-by-design approach.*" Moreover, the SMARTIE project has developed a set of smart city services integrated with the SMARTIE platform. For each of them, main business goals that represent functional goals were determined [6, 7]. Due to space limitations, only one of these services is considered: the smart management of emergencies and energy consumption in buildings. In this use case, business goals include the following: "*the system must be capable of detecting emergency events such as fire* *and notifying these events to authorized parties,*" "*emergency notifications should include information about the people that is within the building for quickly and effectively responding to the emergency,*" "*the system must be capable of detecting the building places where there might be users and whether they have mobility restrictions,*" and "*the system must be capable to efficiently reduce energy consumption of the building based on human presence.*"

3.2. Definition of the Architecture's Domain Model. The defined business goals and use cases are used to build the most elementary architectural view, that is, the PE View. In this view, we think about the things of interest and their properties. As stated in the IoT-ARM, a PE is an identifiable part of the physical environment that is of interest to the user for the completion of some goal. In our use case, the most evident physical entities of interest are the people that are within the building. One of the goals of the system is to help on rescuing people in case of emergency. We come back to the above definition of PE to highlight that PEs must be uniquely identified. In our application scenario, people will be provided with a unique Radio-Frequency Identification (RFID) card. Note that a PE can also be any entity that is part of the environment and is needed for a software artifact to complete some goal. It suggests that sensor devices that enable monitoring the presence of people and the status of the building will be PEs of our system too. Indeed, the IoT-ARM RM defines the Device class as a subclass of PE [4]. Thus, we have decided to consider these devices as PEs in our information model. Nevertheless, whether or not to consider devices as PEs is questionable and dependent on each specific application's design choices. In our system, the most evident example of IoT devices as PEs are RFID sensors that control the access of people holding RFID cards. Other useful devices are sensors that indicate when windows are open, when lights or Air Conditioning (ACs) systems are switched on/off, video cameras with human-detection capability, and so forth. Lastly, the IoT system should be able to monitor building spaces such as rooms and halls, thereby being these spaces PEs for our system too.

Based on the PE View and the defined business goals, the next step is to create our IoT Context View that represents the IoT system's information model. This view is composed by the context view and the domain model. Both models are complementary and essential for the rest of the architecting process. The context view describes the relationships, dependencies, and interactions between the system and its environment. The domain model provides a deep insight into the relationships between the system entities and also interactions with the outside world. Usually, it is easier to first build the context view and afterwards the domain model, since the level of detail given for outside interfaces is lower than that of the IoT system.

Figure 2 shows the SMARTIE context view for the building management use case. The Building Management Office provides RFID cards for access to the building. The Building Automation System (BAS) is a smart gateway (GW) between the building's physical world (i.e., sensors and actuators) and the SMARTIE platform. The building's security system is an external entity that provides additional safety information such as presence detection based on video-camera records. The SMARTIE platform will monitor the status of the building and detect locations with human presence. Based on this information, the Energy Management Service will optimize the energy consumption of the building by controlling lights, AC systems, and energy supplies from solar panels. In case of emergency, information about human presence in the building including personal information (e.g., reduced mobility conditions and telephone numbers) will be notified to Emergency Managers.

Figure 3 shows a subset of the SMARTIE domain model that includes the two pivotal concepts in any IoT domain: the PEs of interest and users. As stated in the IoT-ARM, a user is a human person or a software agent that needs to interact with a PE. Users can be either human beings or Active Digital Artifacts (ADAs). An ADA is a running software application, agent, or service that may access other services or Resources. By keeping this definition in mind, we define several users (upper right corner of Figure 3) such as Emergency Managers, Building Visitors, and Building Automation System (BAS). An Emergency Manager is a person that can be notified of emergency events. In case of emergency, this user may need to interact with the system to do some action or get information for safety (e.g., to know if there is some open door in case of fire). A Building Visitor is a person that is within the building and interacts physically with the environment (e.g., he opens a window and turns on an AC). A BAS is a software agent that interacts with the building's sensors and actuators. A BAS acts as smart gateway (GW) between the building and the platform.

The upper left corner of Figure 3 shows the SMARTIE platform's PEs. A Registered Person in Figure 3 represents people that have been previously registered and given an RFID card. Note that a Registered Person PE is a Building Visitor User too, since the former interacts with the building. We have defined classes for the kinds of sensors and actuators of interest for our application. PEs can be an aggregation of other PEs. Based on this property, we define the Smart Space class that represents a building space composed of sensors and actuators. Each PE is represented in the IoT system by a VE that can be an ADA or a Passive Digital Artifact (PDA) such as a database entry. We have defined high-level VEs such as the Smart Space VE that is a composition of other lower-level VEs such as the Light Sensor VE. Services are associated with VEs and expose Resources hosted on PEs. For example, the Emergency Detection Service informs about the state of the Smoke Detector resource (i.e., binary state of "smoke detection") that is hosted by the Smoke Detector PE.

3.3. Derivation of Requirements and Design Choices. Once we have formulated our business goals, PE View, and IoT Context View, we can proceed with the requirement engineering process to finally derive the design choices that will impact on the last process, that is, the derivation of architectural views. The ultimate goal of SMARTIE is to facilitate the integration of user-centric privacy and governance into IoT applications for smart cities. Thus, security and privacy are first-class business goals in order to enable citizens to (1) control their

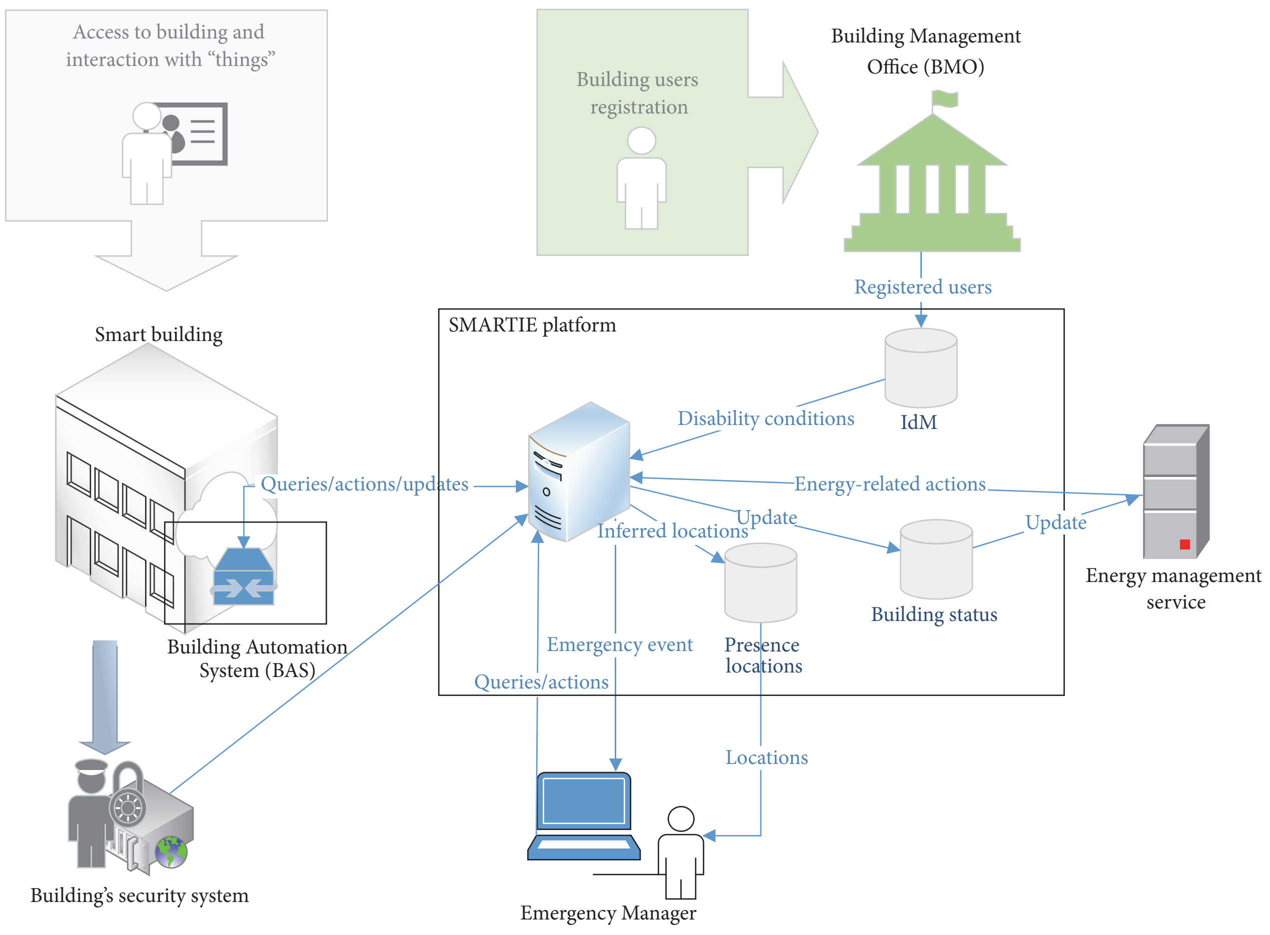


Figure 2: SMARTIE context view for the smart building use case.

devices that join an IoT application to sense and publish data, (2) define fine-grained access control rules for their devices, and (3) decide who can or cannot be in possession of their devices' data. Thus, major requirements and design choices of the SMARTIE platform were deduced from a deep analysis on the IoT-A Threat analysis, UNIs, and tactics. Table 1 shows only a few of SMARTIE design choices, mainly related to access control under different quality perspectives. Section 4 introduces the SMARTIE architecture's components for these design choices. More information about the requirements for the SMARTIE platform can be found at [7].

As it can be seen in Table 1, the IoT-ARM allows determining concrete design choices and correlate them based on different qualitative perspectives and tactics. The design choices taken for the Functional View will impact on the rest of the architecture's views. A relevant design choice about the SMARTIE's functionality was the enforcement of context-aware user access control policies by data producers such as sensors to improve user privacy (e.g., S-DC P.7 under the Privacy perspective in Table 1). Thus, only the actual authorized data will be granted by IoT devices rather than providing sensor data to centralized servers in charge of applying privacy filters. To accomplish this design choice while guaranteeing scalability, other choice on functionality was taken: decentralized access control for IoT devices (e.g., S-DC SP.4 and SP.6 under the scalability and performance perspective in Table 1). To facilitate the extensibility of the SMARTIE platform (e.g., integration with different IoT applications) and device-to-device communication, other design choice was to separate access control from application logic as much as possible (e.g., S-DC EI.1 and EI.2 under the evolution and interoperability perspective in Table 1).

To ensure user privacy, sensitive information needs to be end-to-end encrypted (DC S.10 under the security perspective). This decision requires other design choices at the Information View. For pull communication, SMARTIE uses transport-level encryption (S-DC EI.3 in Table 1). For push communication based on subscriptions, SMARTIE encrypts sensor data based on application-level user-defined attributes that data receivers must satisfy (e.g., S-DC S.5 in Table 1). This design choice ensures user privacy policies regardless of who receives the data (e.g., S-DC P.1 in Table 1); only those receivers that satisfy certain application-level attributes will be able to decipher the data.

3.4. Generation of the Architecture's Functional View. We took the IoT-A Functional View as the foundation for the SMARTIE's architecture. We modified this view based on

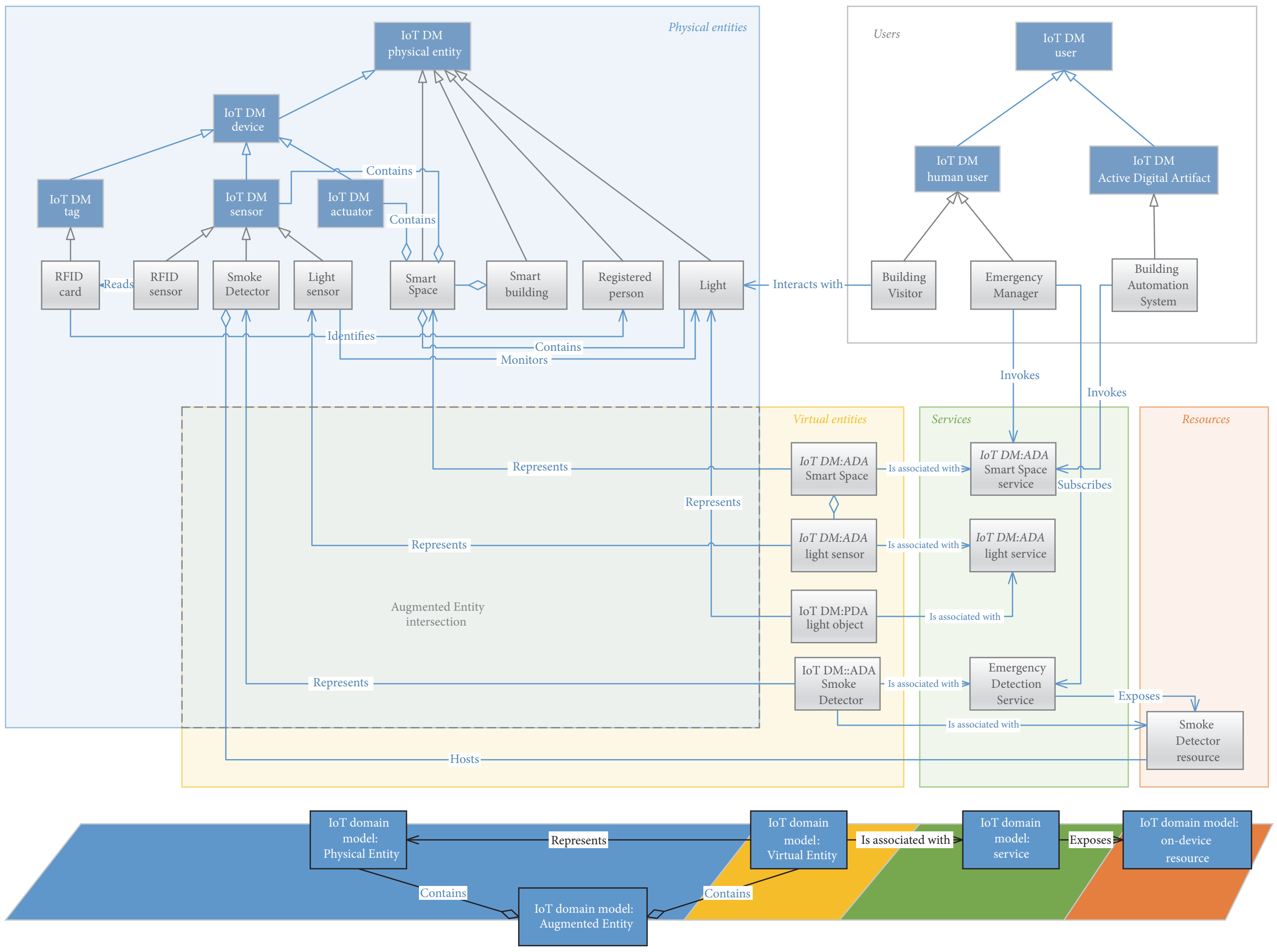


FIGURE 3: Subset of the SMARTIE domain model for the building management use case, and the main associations between the PE, VE, Service, and Resource concepts of the IoT Domain Model. The conceptual combination of a PE with its VE is an Augmented Entity (represented by dashed-lined box).

the design choices for SMARTIE, resulting in the functional architecture shown in Figure 4. In this architecture, the Communication FG represents the variety of communication technologies (e.g., data representation, addressing, and network management) that can be used by devices in IoT systems and provides a common interface for the IoT Service FG.

The Management and the Security FGs contain vertical functionality that can be used by any other FG in the architecture. The former provides all the functionalities that are necessary to govern an IoT system. The latter is responsible for ensuring security and privacy in the IoT system and is further described in Section 4.

The Service Organization FG is used for composing and orchestrating services of different levels of abstraction. The IoT Broker FC provides asynchronous communication (i.e., based on subscriptions/notifications) to match service requests with service offers. This FC relies on the VE Georesolution FC to find out the required IoT Services.

Applications can interact with the IoT system at the VE level that models high-level concepts of the physical world (e.g., "give me the status of windows in the room 102"). The VE Georesolution FC allows registering services by indicating the VE with which they can be associated and discovering services based on location information. The VE-Service FC provides access to VE Services (e.g., "switch off lights in room 102").

Besides the VE level, applications can interact with the IoT system at the IoT Service level by directly communicating with services hosted by devices (e.g., "give me your status" on a temperature sensor). Typically, IoT Services interact with devices and/or network Resources. High-level services are possible such as the Emergency Detection Service and the Energy Consumption Reasoner. The DiGcovery FC allows discovering IoT services by a service description or a service identifier, and it accepts location parameters to filter responses [8]. The Resource Directory with Secure Storage (RD) FC enables the automated registration of services and stores service information encrypted. The RD FC notifies the DiGcovery FC each time a new service is stored in the directory. In turn, the DiGcovery FC will automatically create

Table 1: IoT-A tactics for several perspectives and their associated IoT-A and SMARTIE design choices, identified by "DC" and "S-DC," respectively, for different architectural views. Design choices for SMARTIE are highlighted in italic. The design choices in the Functional View impact the design choices in the Information View and the Deployment and Operation View. In the last view, the design choices indicate the deployment's FCs that will be affected.

Perspectives	Tactics	Design choices (DC) for views		
		Functional	Information	Deployment and Operation
Security (S)	Use access policies	Policy-based service access (DC S.4)	Stored information must be managed to support access control mechanisms (DC S.6)	Authorization FC (DC S.7)
			Stored information can be encrypted based on application-level attributes (S-DC S.1)	*Key Exchange and Management FC (S-DC S.2) Deploy Authorization FC at end-devices (S-DC S.3)*
	Secure communication infrastructure	End-to-end encryption (DC S.10)	*Information transmission channel for devices or pull communication is secured (S-DC S.4)*	End-to-End, Network Communication and Key Exchange and Management FCs (DC S.12)
			Application data is encrypted for push communication (S-DC S.5)	*Authorization, DEM, Service Orchestration FCs (S-DC 6)*
Evolution and interoperability (EI)	Apply design techniques that facilitate change	*Access policies separated from application logic (S-DC EI.1)*	*Access policies must be in a standard, flexible format (XACML or JSON) (S-DC EI.3)*	*Authorization FC (S-DC EI.5)*
		Access control in devices must not require synchronization from centralized servers (S-DC EI.2)	*Authorization information is included in client requests (DCapBAC method) (S-DC EI.4)*	
Privacy (P)	Minimized unauthorized access to implicit information	Access control management (DC P.5)	Stored information managed to support access control mechanisms (DC P.6)	*Authorization, Key Exchange and Management FCs (S-DC P.2)*
			Data is encrypted based on application attributes (CPABE method) (S-DC P.1)	
		Enablement of a scalable and secure key distribution between communicating subjects (DC P.8)	*Constrained services dynamically learns authorized client's key based on authorization tokens (S-DC P.3)*	
	Enable the user to control their privacy settings	*Support flexible privacy rules (S-DC P.4)*	*Privacy rules provided by a Policy Decision Point with a flexible data format (S-DC P.5)*	*Authorization FC (S-DC P.6)*
		Enforce user's privacy settings through authorization rules at edge devices (S-DC P.7)	*Privacy rules translated to rules in authorization tokens for devices (S-DC P.8)*	

Table 1: Continued.

Perspectives	Tactics	Design choices (DC) for views		
		Functional	Information	Deployment and Operation
Scalability and performance (SP)	Reduce computation complexity	FC with reduced capabilities (DC SP.14)	No impact	Less functional component deployed (DC SP.15)
		Pushed information is not authorized (S-DC SP.1)	*Implicit authorization for pushed information based on attributes (CP-ABE) (S-DC SP.2)*	*Authorization, Service Orchestration and Key Exchange and Management FCs (S-DC SP.3)*
		Authorization server does not synchronize access policies at devices (S-DC SP.4)	No impact	*Authorization FC devices (S-DC SP.5)*
	Minimize the use of shared resources	*Devices do not ask authorization servers for access control decisions (S-DC SP.6)*	No impact	*Authorization FC devices (S-DC SP.7)*
	Optimized repeated processing	*Lightweight cryptographic operations for constrained devices (S-DC SP.8)*	*Cryptography based on lightweight methods (e.g. elliptic curve) for devices (S-DC SP.9)*	*Authentication, Key Exchange and Management FCs (S-DC SP.10)*

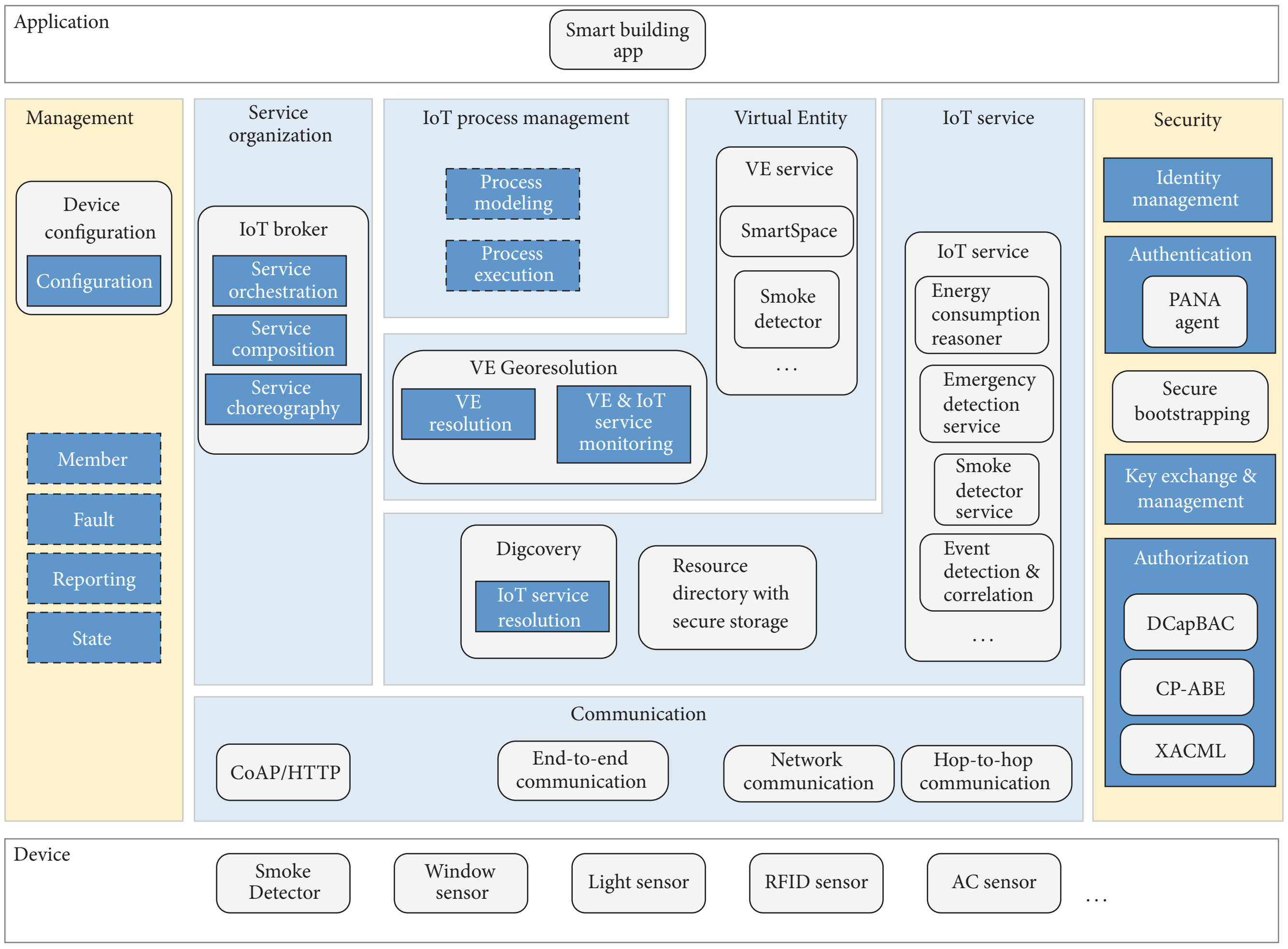


FIGURE 4: Simplified version of SMARTIE platform's Functional View and its Functional Groups (FGs). The device and application FGs are out of the scope of the ARM. Grey-colored rounded rectangles represent FCs defined for SMARTIE and dark blue-colored rectangles are FCs from IoT-ARM.

VE-Service associations in the VE Georesolution FC when new IoT Services are registered.

4. Functional Components for User-Centric Privacy

The Security FG of the IoT Functional View includes FCs for Authorization, Key Exchange and Management (KEM), Trust and Reputation, Identity Management (IdM), and Authentication. The SMARTIE architecture includes all these FCs except the Trust and Reputation FC, that is, planned to be considered in the near future. Figure 4 shows the primordial security-related FCs in the SMARTIE architecture, but the rest of FCs are described in [9].

The IoT Authorization FC is mainly distributed between three SMARTIE FCs: the eXtensive Access Control Markup Language (XACML), Decentralized Capability-Based Access Control (DCapBAC), and Ciphertext-Policy Attribute-Based Encryption (CP-ABE) FCs. These architectural artifacts satisfy many of the SMARTIE requirements and architectural design choices for user-centric governance, privacy, and scalability described in Section 3.3 and listed in Table 1.

The XACML FC as its name states allows defining fine-grained attribute-based access control policies through XACML. This FC stores and handles the user's context-aware authorization policies that will determine the devices that can join specific IoT applications and the entities that can access to these devices. Thus, any IoT application that allows users to define their access control policies by the XACML standard can be easily integrated with SMARTIE.

The DCapBAC FC is a delegated authorization mechanism [10] that flexibly allows a client device or application to access to a resource at a server device. This FC is distributed between the devices (i.e., the server and the client) and the SMARITE platform that provides authorization decisions. This authorizing SMARTIE functionality is called Authorization Server (AS). The client device requests the AS authorization to access to the server device. If this authorization request is granted, the AS generates a self-contained JSON-encoded authorization token that embeds lightweight but context-aware authorization rules. Authorization tokens enable the server device to locally authenticate and authorize the client as long as they are signed by a recognized authority. The DCapBAC FC transforms user-defined XACML-based rules into lightweight JSON-based rules embedded in authorization tokens. Authorization tokens are integrity-protected and are resilient against centralized server failures (i.e., as long as the authorization token has not expired, clients and servers can directly communicate without any centralized authority). The DCapBAC FC therefore guarantees that user privacy policies are accomplished by any of the user's devices. This FC also facilitates the integration of user devices into multiple IoT applications since authorization rules follow a RESTful approach, without application-specific access control logic.

The CP-ABE FC implements an encryption-based authorization mechanism that encrypts data based on dynamic attribute-based policies that data consumers must hold [11]. This FC contains the policy attributes associated with data types and encrypt data based on these attributes. Here, a data type can be associated with a single data producer (e.g., "video from my vc identified by vc-entrance") or aggregated data (e.g., "my location" combines data from multiple sources such as the user's cell phone or laptop). When the IoT Broker FC needs to notify some information to a set of consumers, it requests the CP-ABE FC to encrypt this information by specifying the data type.

5. Real Scenarios for a Smart City

The SMARTIE IoT-ARM-compliant platform has been across country deployed for different use cases such as smart traffic management, public transportation, and environmental alarms [12]. The use case considered in this paper, emergency and energy management in smart buildings, has been tested in University of Murcia (UMU), Spain. The SMARTIE platform has been deployed for a smart building with 8 floors and a total area of 6.500 m^2. For the considered use cases, RFID sensors, video cameras and sensors for windows, doors, AC systems, and lights inform the Emergency Detection Service and Energy Consumption Reasoner about human activity in the building. The performance of SMARTIE in this scenario has been evaluated by taking time measurements of each of its components [13].

Figure 5 shows the main communication flows for the secure dissemination of data from sensor devices and some of the FCs of the SMARTIE architecture in Figure 4. The BAS connects to Home Automation Modules (HAMs) that serve as GWs to sensors and actuators. HAMs are intelligent modules that are distributed throughout the building and provide uniform interfaces to the BAS. The HAMs, BAS, and some devices can directly interact with the platform through RESTful communication (i.e., HTTP or CoAP).

When devices start up, they join the SMARTIE platform based on the Secure Bootstrapping FC that is composed of several other FCs (step 1 in Figure 5). A device first authenticates the PANA FCs that implements an extension of the Protocol for carrying Authentication and Network Access (PANA) [14]. This extension merges device authentication and authorization in order to save Resources at constrained devices [15]. The device needs to prove possession of a previously installed SMARTIE symmetric key and the PANA will query the XACML FC for the authorization of the device's key to join the platform. If this authorization request is successful, the PANA FC will register the device to the RD FC based on the privacy rules set for the device at the XACML FC (step 2 in Figure 5). Moreover, the PANA FC will reply to the device with the public key of the sensor's AS for future client requests. If the device was a sensor that had to periodically publish to the IoT Broker, the PANA would also reply to the device with an authorization token for publication (step 3 in Figure 5).

When the RD FC completes a device registration request, it notifies the DiGcovery FC that the device has been registered. In turn, the DiGcovery FC registers an association between the device's IoT Service and its corresponding VE to the platform's VE Georesolution FC.

When a sensor publishes to the IoT Broker FC (step 4 in Figure 5), the sensor attaches its authorization token to the body of its CoAP request towards the IoT Broker FC. Since the authorization token is signed by the DCapBAC FC, the IoT Broker FC can verify the authenticity of this token. Moreover, the sensor needs to be authenticated by proving possession of the public key contained in the token during the Datagram Transport Layer Security (DTLS) handshake with the IoT Broker FC.

When an application connects to the platform, it authenticates the Authentication FC that relies on the IdM FC (step 5 in Figure 5). After authentication, the KEM FC will provide the application with the proper attributes and cryptographic material to decrypt notifications from the IoT Broker FC. To this end, the KEM FC will communicate with the CP-ABE FC to obtain the necessary information. To subscribe to the IoT Broker, the application first obtains an authorization token from the DCapBAC FC and uses it to subscribe to a given service (steps 6 and 7 in Figure 5, resp.). The IoT Broker FC does not verify if the application is authorized to access to this required service's data. Instead, it creates the subscription and whenever the service produces some data, this FC requests the CP-ABE FC to encrypt this data and notifies all the subscribers of the encrypted data.

6. Related Work

The Alliance for Internet of Things Innovation (AIOTI) (http://www.aioti.org/) was launched by the European Commission in March 2015 in order to create and standardize an IoT ecosystem in Europe. This alliance is currently consolidating an IoT Reference Architecture mainly based on the results from IoT-A and oneM2M. The latter is an ETSI initiative that started developing their Reference Architecture [16] in parallel to the IoT-A project. There are other emerging initiatives that are intended to promote interoperability for large-scale IoT deployments. The IEEE Standard for an Architectural Framework for the Internet of Things (IEEE P2413) [17] defines a three-tier architectural framework, addressing descriptions, definitions, and common aspects in different IoT domains. The ITU-T Y.2060 "Overview of the Internet of Things" recommendation [18] follows a similar approach by providing a more harmonized view about the

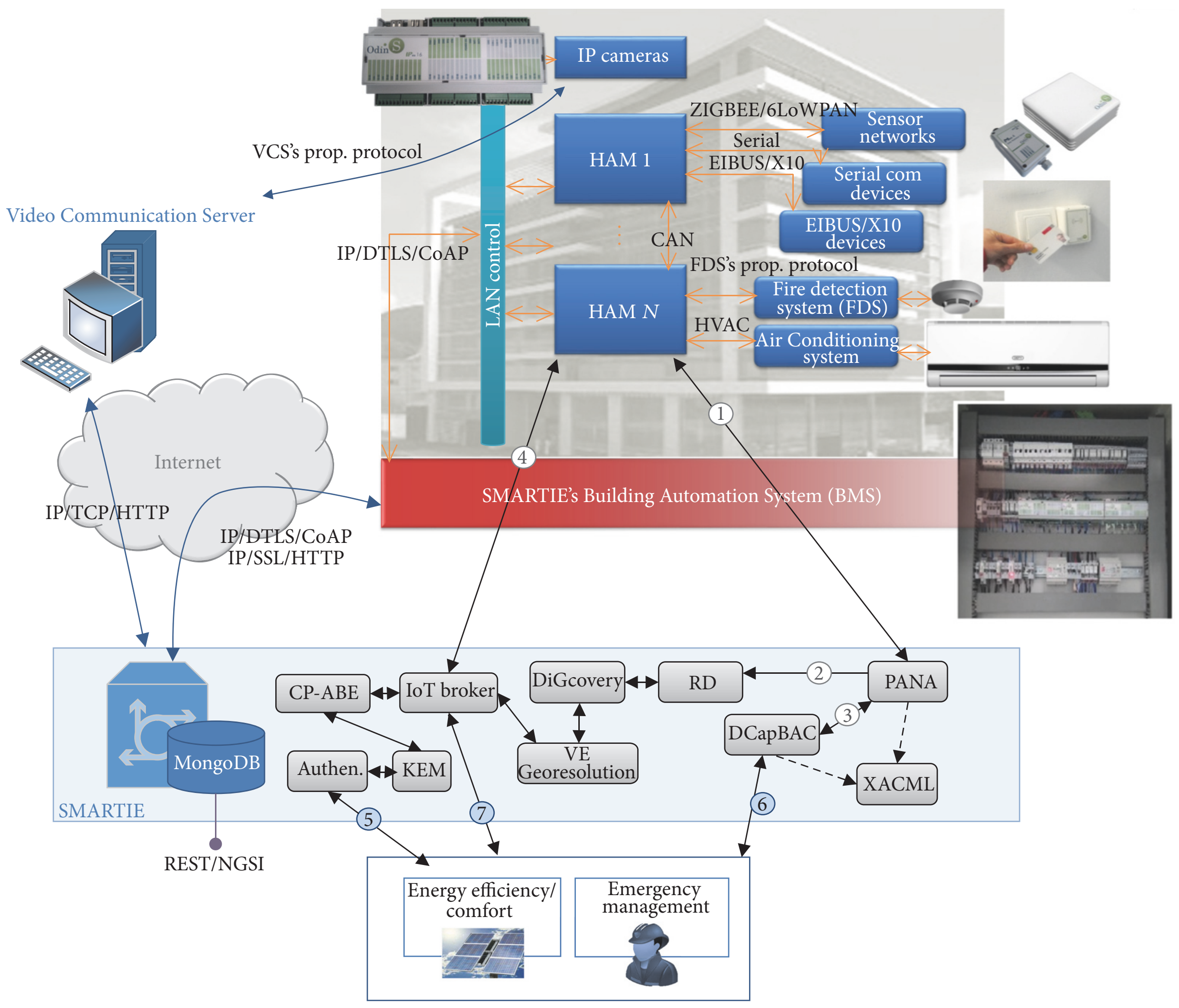


Figure 5: Outline of some Functional Components (FCs) of the SMARTIE platform for the building management use case. Arrows indicate flows of information between FCs. HAMs (picture at the top) are IoT gateways with μ-C-powered boards (32-bit MIPS μC running at 80 MHz, with 16 configurable inputs/outputs) that support RS-232, RS-485, ZigBee, 6LowPAN, and Bluetooth connections. Although arrows only depict the case when sensors are associated with HAMs, IP-enabled sensors to directly communicate with the BMS or the platform. Sensors (images at the top-right corner) have a 16-bit μC that runs at 8 MHz and support 802.15.4/6LowPAN/CoAP.

IoT ecosystem. IoT reference architectures are today a recent research topic and analyses that show the application of the IoT-ARM is scarce. The authors of [19] have to a very limited extent addressed this topic. However, this paper only shows some ARM-compliant architectural aspects of the proposed architecture.

SMARTIE has focused on (1) the by-design integration of security and privacy in IoT applications through the IoT-ARM and (2) the scalability of smart cities based on the decentralization of core functionalities for the secure dissemination of data. Other EU-funded projects have used the IoT-ARM for designing their platforms. The COSMOS project relied on the IoT-ARM to identify the requirements for its platform for decentralized management of things in the IoT [20]. This project provided a trust and reputation model based on different kinds of security threats. The FIESTA-IoT project aims to provide a common framework to access to and share IoT datasets in a testbed-agnostic way. The project provides an analysis of different IoT testbeds based on the IoT-ARM that is used as the foundation for its platform design [21]. Other EU-funded projects have also their focus on the dissemination of city data or the enhancement of the current state of IoT by integrating security and privacy. The City Platform as a Service (CPaaS.io) project is developing a platform for merging city data form a diversity of sources (e.g., social media, IoT data, and government data) and making this data available to third-parties (https://cpaas.bfh.ch/). The RERUM project [22] aims to make IoT more secure by providing a middleware based on OpenIoT [23]. It provides CoAP with JSON signatures based on Elliptic Curve Cryptosystem (ECC) of at least 192 bits for message integrity. SOCIOTAL [24] looks at IoT security from a societal point of view. Its main goals are user trust, user control, and transparency with the ultimate goal of obtaining the confidence of everyday users and citizens. BUTLER project integrates OAuth 2.0 between authorization servers and clients for the obtention of access tokens and the derivation of security material [25].

Since the literature on IoT security is extensive, it falls out of the scope of this paper due to space limitations. We refer the reader to the references in this paper to find out more about the main aspects on security and privacy of SMARTIE: decentralized access control, encryption-based authorization, and secure bootstrapping.

7. Conclusion

This paper has given the authors' insights into the application of the IoT-ARM to generate the architecture of the SMARTIE, an IoT platform for secure and privacy-preserving dissemination of data in smart cities. The main goal of this platform is to empower citizens to take control of their privacy policies and devices. To this end, based on the IoT-ARM guidelines on security and scalability, SMARTIE provides architectural artifacts for efficient and scalable security and user-centric privacy. The paper has introduced SMARTIE user access control for pull communication (i.e., decentralized authorization tokens) and push communication (i.e., data encryption based on application attributes). SMARTIE provides an application-agnostic, scalable, and privacy-preserving platform for data dissemination in large deployments of smart cities.

One of the goals of the SMARTIE EU-funded project has been to evaluate the IoT-ARM for the generation of IoT platforms. Although the IoT-ARM represents a big step towards the homogenization of quality aspects in IoT platforms, further work on this Reference Architecture is necessary. In its current state, its steep learning curve may discourage some architects from using it.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work has been sponsored by European Commission through the FP7-SMARTIE-609062 EU Project and the Spanish National Project CICYT EDISON (TIN2014-52099-R) granted by the Ministry of Economy and Competitiveness of Spain (including ERDF support).

References

[1] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of things: a survey on enabling technologies, protocols, and applications," *IEEE Communications Surveys & Tutorials*, vol. 17, no. 4, pp. 2347–2376, 2015.

[2] G. Ho, D. Leung, P. Mishra, A. Hosseini, D. Song, and D. Wagner, "Smart locks: Lessons for securing commodity internet of things devices," in *Proceedings of the 11th ACM Asia Conference on Computer and Communications Security, ASIA CCS 2016*, pp. 461–472, Xi'an, China, June 2016.

[3] S. Krco, B. Pokric, and F. Carrez, "Designing IoT architecture(s): a European perspective," in *Proceedings of the 2014 IEEE World Forum on Internet of Things (WF-IoT '14)*, pp. 79–84, Seoul, South Korea, March 2014.

[4] A. Bassi, M. Bauer, M. Fiedler et al., *Enabling Things to Talk: Designing IoT Solutions with the IoT Architectural Reference Model*, Springer Berlin Heidelberg, 2013.

[5] S. Haller, A. Serbanati, M. Bauer, and F. Carrez, "A domain model for the internet of things," in *Proceedings of the 2013 IEEE International Conference on Green Computing and Communications and IEEE Internet of Things and IEEE Cyber, Physical and Social Computing, GreenCom-iThings-CPSCom 2013*, pp. 411–417, Beijing, China, August 2013.

[6] "SMARTIE Use Cases", SMARTIE project Deliverable 2.1, http://www.smartie-project.eu/download/D2.1-Use%20Cases .pdf.

[7] SMARTIE Requirements", SMARTIE project Deliverable 2.2, http://www.smartie-project.eu/download/D2.2-Requirements. pdf.

[8] A. J. Jara, P. Lopez, D. Fernandez, J. F. Castillo, M. A. Zamora, and A. F. Skarmeta, "Mobile digcovery: discovering and interacting with the world through the internet of things," *Personal and Ubiquitous Computing*, vol. 18, no. 2, pp. 323–338, 2014.

[9] SMARTIE Initial Architecture Specification", SMARTIE project Deliverable 2.3, http://www.smartie-project.eu/download/D2.3-Initial%20Specification.pdf.

[10] J. L. Hernández-Ramos, A. J. Jara, L. Marín, and A. F. Skarmeta Gómez, "DCapBAC: embedding authorization logic into smart things through ECC optimizations," *International Journal of Computer Mathematics*, vol. 93, no. 2, pp. 345–366, 2016.

[11] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute-based encryption," in *Proceedings of the IEEE Symposium on Security and Privacy (SP '07)*, pp. 321–334, May 2007.

[12] "SMARTIE Real-world test- screenplay", SMARTIE project Deliverable 6.1, http://www.smartie-project.eu/download/D6.1-Real%20world%20test%20-%20Screenplay.pdf.

[13] "SMARTIE Test Report", SMARTIE project Deliverable 6.3, http://www.smartie-project.eu/download/D6.3-Test%20Report .pdf.

[14] D. Forsberg, Y. Ohba, B. Patil, H. Tschofenig, and A. Yegin, "Protocol for Carrying Authentication for Network Access (PANA)," RFC Editor RFC5191, 2008.

[15] D. Garcia-Carrillo and R. Marin-Lopez, "Lightweight CoAP-based bootstrapping service for the internet of things," *Sensors (Switzerland)*, vol. 16, no. 3, article no. 358, 2016.

[16] "oneM2M Functional architecture", TS 118 101 V.2.10.0, August 2016, http://www.onem2m.org/images/files/deliverables/Release2/TS-0001-%20Functional_Architecture-V2_10_0.pdf.

[17] IEEE. Standard for an Architectural Framework for the Internet of Things (IoT) IEEE P2413. 2016. http://grouper.ieee.org/groups/2413/Intro-to-IEEE-P2413.pdf.

[18] ITU-T. Recommendation Y.2060 Overview of the Internet of Things. 2012. https://www.itu.int/rec/dologin_pub.asp?lang=e&amp;id=T-REC-Y.2060-201206-I!!PDF-E&amp;type=items.

[19] J. Fernandes, M. Nati, N. S. Loumis et al., "IoT Lab: Towards co-design and IoT solution testing using the crowd," in *Proceedings of the 2015 International Conference on Recent Advances in Internet of Things, RIoT 2015*, Singapore, April 2015.

[20] S. Citrigno, S. Graziano, and D. Saccà, "Cooperation of Smart Objects and Urban Operators for Smart City Applications," in *Management of Cyber Physical Objects in the Future Internet of Things*, Internet of Things, pp. 157–174, Springer International Publishing, 2016.

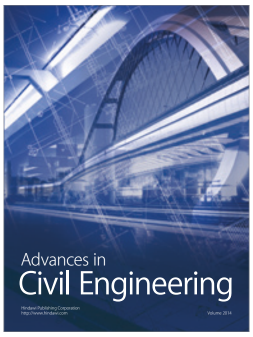
[21] "Analysis of IoT platforms and Testbeds", Federated Interoperable Semantic IoT/Cloud Testbed and Applications (FIESTA-IoT) project deliverable D2.2, http://fiesta-iot.eu/wp-content/uploads/2016/06/FIESTAIoT-WP2-D22-web.pdf.

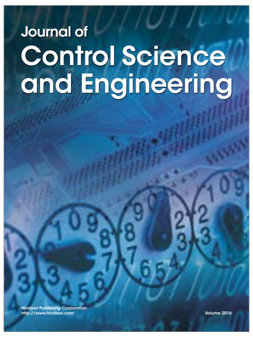
[22] G. Moldovan, E. Z. Tragos, A. Fragkiadakis, H. C. Pöhls, and D. Calvo, "An IoT middleware for enhanced security and privacy: The RERUM approach," in *Proceedings of the 8th IFIP International Conference on New Technologies, Mobility and Security, NTMS 2016*, Larnaca, Cyprus, November 2016.

[23] J. Kim and J.-W. Lee, "OpenIoT: An open service framework for the Internet of Things," in *Proceedings of the 2014 IEEE World Forum on Internet of Things, WF-IoT 2014*, pp. 89–93, Seoul, South Korea, March 2014.

[24] Hernandez-Ramos, J. L., Bernal, J., Skarmeta, A., Elicegui I., Nati, M. Gligoric, N., WP2 – Decentralised governance and trust framework. http://sociotal.eu/sites/default/files/docs/deliverables/SOCIOTAL_D2.2-Framework_specification_for_Privacy_and_Access_Control_Final.pdf.

[25] S. U. Khan, L. Lavagno, C. Pastrone, and M. A. Spirito, "Online authentication and key establishment scheme for heterogeneous sensor networks," *International Journal of Distributed Sensor Networks*, vol. 2014, Article ID 718286, 11 pages, 2014.





Hindawi

Submit your manuscripts at
https://www.hindawi.com