FP7-SMARTCITIES-2013
Project number: 609062
http://www.smartie-project.eu/

SMARTIE

Deliverable D2.3

Initial Architecture Specification

Editor:	UMU
Dissemination level: (Confidentiality)	PU
Suggested readers:	Consortium/Experts/other reader groups
Version:	1.0
Total number of pages:	69
Keywords:	Architecture, interfaces, IoT platform

Abstract

This deliverable is based on the work done in previous deliverables such as D2.2, D3.1 and D4.1. D2.2 analysed the security requirements of IoT ecosystems. The deliverables D3.1 and D4.1 defined the individual components proposed in SMARTIE to accomplish secure information gathering, storage, access control and preserve the data privacy. In this deliverable, the integration of all components is presented in combination with the Smartdata Platform. This deliverable describes the initial architecture of SMARTIE and the main interactions among the components.

Disclaimer

The information, documentation and figures available in this deliverable are written by the SMARTIE partners under EC co-financing (project number: 609062) and does not necessarily reflect the view of the European Commission.

Impressum

[Full project title] Secure and sMArter ciTIes data management

[Short project title] SMARTIE

[Number and title of work-package] WP2 Platform Model
[Document title] Initial Architecture Specification
[Editor: Name, company] Antonio Skarmeta, UMU

[Work-package leader: Name, company] Jens-Matthias Bohli, NEC

Copyright notice

Executive Summary

This deliverable presents the initial architecture of SMARTIE using a methodology and modelling based on the IoT-A Architecture Reference Model (ARM). This is accomplished to the IoT-A guidelines and provides a series of views that describe the initial architecture.

The SMARTIE project includes a set of components that work together covering different aspects, such as secure data exchange and privacy preservation. In heterogeneous IoT ecosystems, these aspects are mandatory to define a secure IoT architecture where all the components can be identified and the information access can be controlled.

Purpose and scope of the document

This document relates to Task T2.3 of DoW. In this task the requirements are mapped into a proposed architecture for the whole system that meets the objectives of the project. An existing architecture (IoT-A) is analysed to be used as a basis. The components to be developed in this project are integrated into the architecture. Information management and a platform of services is enabled as a unifying umbrella, which operates above heterogeneous network devices and data sources in order to provide advanced secure information services.

List of authors

Company	Author
UMU	Antonio Skarmeta, Alfredo Quesada, Rafael Marín Pérez, Dan García, Miguel Ángel Zamora, M. Victoria Moreno
NEC	Jens-Matthias Bohli, Martin Bauer
IHP	Anna Sojka-Piotrowska, Jana Krimmling
DNET	Boris Pokrić
PTIN	Luis Cortesão, Ricardo Azevedo, Telma Mota, Fábio Gonçalves

Table of Contents

List of Tables

List of Figures

Abbreviations

ARM	Architecture Reference Model
CoAP	Constrained Application Protocol
HTTP	Hypertext Transfer Protocol
XML	Extensible Mark-up Language
IoT	Internet of Things
IoT-A	Internet of Things Architecture
RFID	Radio-frequency Identification
UDP	User Datagram Protocol

1 Introduction

The previous deliverables (D2.2, D3.1 and D4.1) provide the main aspects to design the SMARTIE architecture. Concretely, D2.2 includes a set of requirements identified in each use case, grouped in categories such as security or privacy. D3.1 describes the components in charge of handling secure information, gathering and storage. D4.1 presents the components related to information access and privacy preservation.

This document is the first contribution of SMARTIE in the design of a secure architecture for heterogeneous IoT ecosystems. It describes the interoperability among all components in an initial set of typical IoT-interactions to cover the security and privacy requirements.

The SMARTIE architecture is based on the Architecture Reference Model (ARM) proposed in the IoT-A project. Therefore, this deliverable provides an introduction of IoT-A ARM to show the guidelines followed in the design of the SMARTIE architecture. Concretely, we follow the ARM functional groups to organize all components of the SMARTIE project. For each component, we show some preliminary interfaces that allow interoperability with other components and IoT functions. In addition, the Smartdata platform is described in line with the SMARTIE architecture. Functions offered by the platform are described in the same way as the remaining SMARTIE components. In particular, the Smartdata platform provides state-of-the-art functionality for realising IoT-Systems in areas that are not covered by SMARTIE innovations to realise IoT systems.

In this document, the interoperability among the components is described through interactions showing how components are used in typical IoT interactions. For the SMARTIE components, the integration is divided in 8 main interactions such as: Registration, Discovery, Retrieve, Subscription, Actuation, Event Detection, Federation of Systems and Intrusion Detection. These interactions show the specific operations of components and their communication exchanges in order to support the security and privacy requirements of IoT scenarios.

1.1 Relation to S&T Objectives

This deliverable presents the first overall view of the SMARTIE architecture. It is a big step towards the whole overview of the SMARTIE developments and contributes to WP2-underlying objective O1 “Understanding requirements for data and application security and creating a policy-enabled framework supporting data sharing across applications”. The document shows how the SMARTIE components can be used which is the basis of the selection of components for the demonstration.

1.2 Beyond State-of-the-Art Contributions

The SMARTIE components improve IoT systems in particular with respect to security, privacy and trust. This deliverable shows how the innovations can be used in an IoT architecture and provides a view of IoT interaction using the SMARTIE solutions to improve the security, for instance distributed access control and protecting data confidentiality beyond the sensor.

The remainder of the document is organized as follows. Section 2 summaries the Architecture Reference Model (ARM) as defined in the IoT-A project. Section 3 describes the SMARTIE architecture and components according the functional groups of the ARM architecture. Section 4 introduces the Smartdata Platform and the integration in the whole architecture. Section 5 presents the interactions among the components following the architecture proposed. Finally, Section 6 remarks the conclusions of the deliverable and the future works towards the final architecture.

2 IoT-A methodology: ARM Introduction

When facing a design involving the characteristics, features and challenges of the IoT, working with a solid base is important. The "Internet of Things" (IoT) is often seen as vertical silos without interoperability and with models of governance that do not help with collaboration or interoperability. A good way to proceed is to use a previous ground based model that aids to identify the unique challenges IoT systems bring. The IoT-A methodology is chosen for SMARTIE's design because it enables to concentrate the effort of the development on the main aspects of the project reusing concepts already established in the IoT-A ARM.

IoT-A, the Internet of Things Architecture (IP EU project from 2009 to 2012)[1] proposes a baseline design with an Architecture Reference Model, which aims to lower the aforementioned barriers such as interoperability or scalability that IoT designs have to face. This Architecture Reference Model (ARM) uses basic building blocks together with guidelines and principles that are key in the design of protocols, interfaces and algorithms for an IoT system. This methodology and guidelines enable the design of an IoT system providing key features such as scalability and interoperability. Another important aspect of IoT-A ARM is that it provides a language and concepts that improves the communication between the different partners. In addition, such a common language and semantics can be used for comparing the different approaches of each partner. All this is accomplished by means of a functional view and an information view, which, as we will explain later, consist of decomposition diagrams, interfaces, sequence charts, technical use cases and interaction examples. This methodology and guidelines will be summarized in the next section.

2.1 IoT-A Methodology

The IoT-A methodology for designing an IoT system uses a set of views that, in the end, define an architecture. These views are: the physical view, context view, functional view, operational view, and deployment view. The complete definition of the methodology can be found in the Deliverable D1.5 provided by the IoT-A project [2]. Figure 1 is a diagram that summarizes the IoT-A methodology. This methodology will be explained next, in this section.

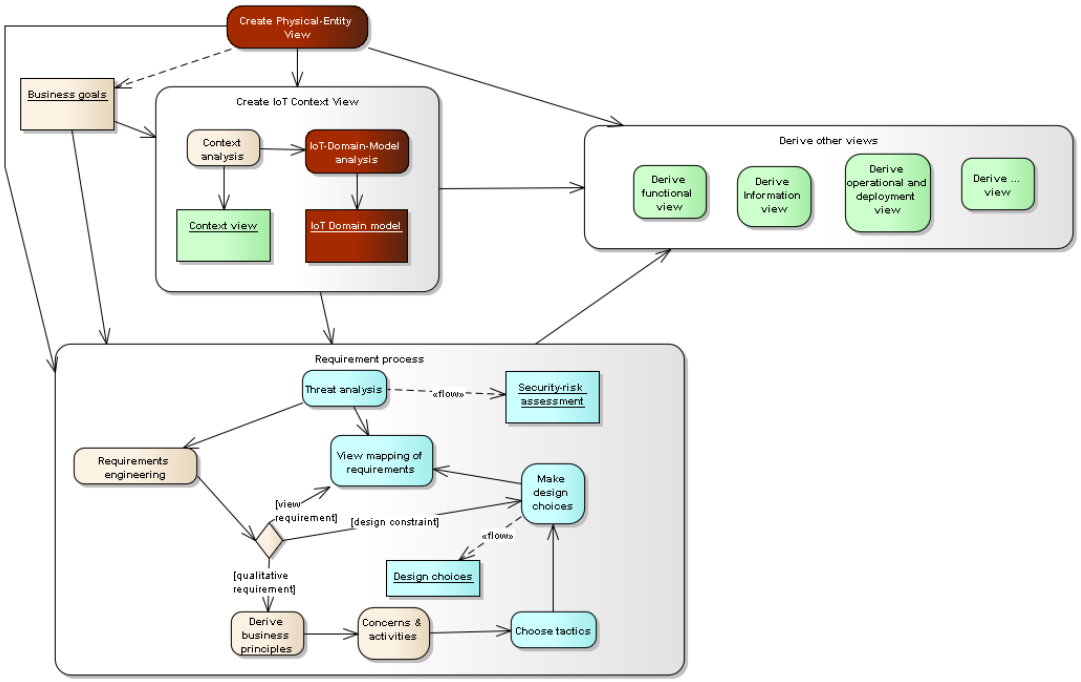

Figure 1: IoT-A Methodology Diagram

2.1.1 Physical Entity View and Business goals

The Physical EntityView describes the physical world scenario, involving physical users and physical entities, not considering virtual or digital objects at this point. In addition, it describes relevant information, measurement, etc. to be processed within the system to achieve some predefined goals.

The purpose of the IoT system is specified in the business goals. Based on this purposes the system is designed and built. This includes the list of actors and things involved in the system.

In SMARTIE this step is done in the deliverable D2.1, where the different trial sites are described and a variety of use cases are described.

2.1.2 IoT Context View

As can be seen in Figure 1, the IoT Context view is composed of two parts: the context view and the IoT Domain Model. The context view is an architecture generated at the very beginning of the architecture design process. It describes "the relationships, dependencies, and interactions between the system and its environment". The context view can be seen in the Deliverables 2.1 and 2.2 of SMARTIE where the use cases and requirements are defined.

For instance, in the Smart Energy Management use case can be seen, that the relation and interaction between the use case and the environment is about managing intelligently the energy use of buildings by means of monitoring and using consumption feedback. Automation systems, sensors as well as actuators are the tools to interact with the environment. Some of the most relevant ways of managing energy efficiency is controlling the lighting and HVAC.

The IoT Domain Model offers an ontological and semantic wrapper to the context view. Therefore aiding to the comprehension of the IoT core entities and the way they are related to each other.

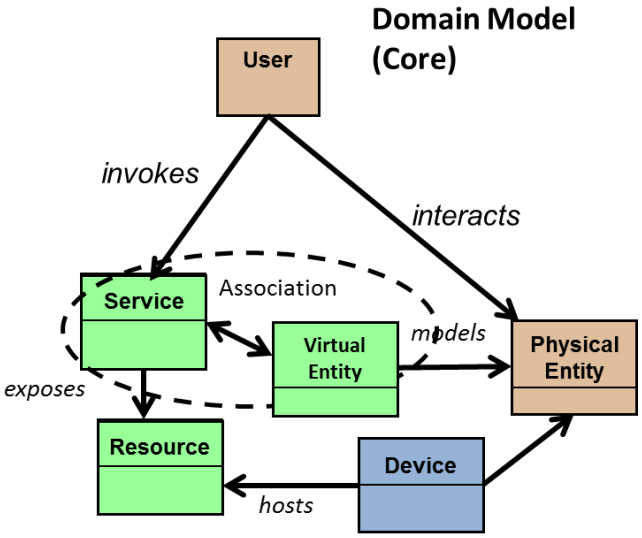

Figure 2: IoT Domain Model

An important concept introduced in the IoT Domain, Figure 2, is the Virtual Entity (VE). The VE is a model of a physical entity, representing the important features that we are interested in. For instance, the status of the cargo of a lorry can be modelled by the temperature, humidity and light of the environment, which can be measured in case the cargo is perishable. That way the modelled cargo can be processed and understood by the system.

2.1.3 Requirement process

At this point, two of the views that constitute an IoT Architecture are described: Physical-Entity view and the IoT Context View. With the input of these two views a threat analysis can be conducted to identify potential weaknesses of the system use cases. That is, those security risks need to be addressed by the requirements.

IoT-A provides a list of requirements that can be addressed by any IoT system, including functional and non-functional requirements; the list of requirements is referred to as UNI.

Based on this list of requirements, the IoT-A ARM process recommends using it as a guideline to derive specific requirements for the system. As an example, UNI.018 says "The system shall support data processing…" In the design of the system should be identified what kind of processing is required, such as the kind of data, etc.

The elicited requirements demand some functionality that has to be provided by the system. This functionality can be grouped so that more or less complex components are generated. The groups of functionality can be related to aspects such as scalability, availability, security, performance, etc. this way, it is easier to specify the components that may already exist or have to be designed, to address the requirements.

2.1.4 Functional View

The ARM of IoT-A identifies functional groups and functional components from the previous stage, the requirement process. This set of functional groups can be considered common to IoT systems as it gathers the essential functionality of an IoT system (see Figure 3).

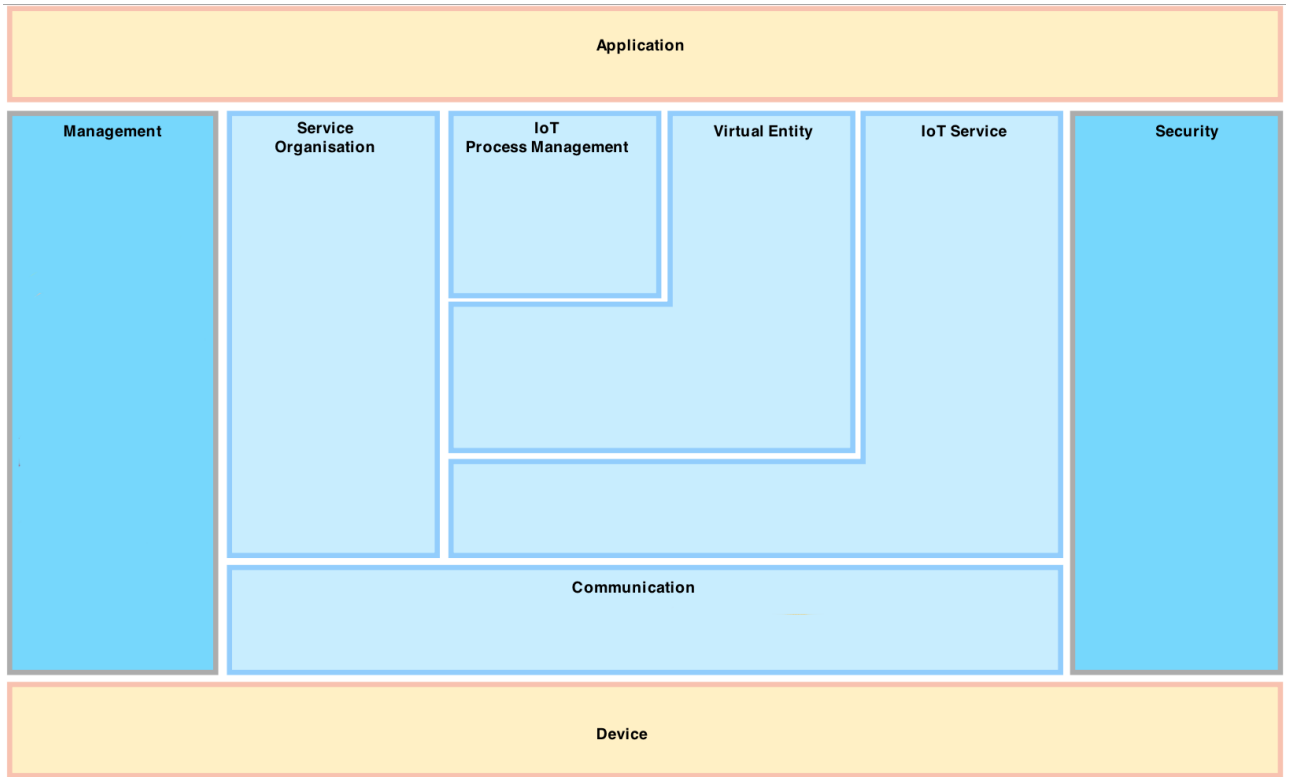

Figure 3: IoT-A Architecture Reference Model (ARM)

IoT-A ARM presents 9 Functional Groups (FGs) detailed in D1.5 of IoT-A ([2], p 108). They are summarized next:

- Application: This Functional Group represents the user or, the application acting on behalf of the user.

- Device: This Functional Group represents the target of the interaction. That being a sensor, source of information, a tag, actuator (modifier of the state of the system), etc.
- IoT Process Management: This Functional Group identifies the new concepts or interfaces that are needed to gear the traditional processes towards the peculiarities the IoT presents.
- Service Organisation: The Service Organisation Functional Group is in charge of managing and orchestrating Services at the different levels of abstraction they are offered on.
- Management: This Functional Group deals with the management of the IoT devices' configuration, monitoring, reporting, faults, etc., as well as the administration of the IoT system as a whole.
- Virtual Entity: The Virtual Entity Functional Group provides the functions necessary for interacting with the IoT system through Virtual Entities (VE's). In the same way it provides the necessary functionalities for discovery and lookup of the services being offered for the aforementioned Virtual Entities.
- IoT Service: This Functional Group gathers the IoT Services and their functionalities such as name resolution, discovery and lookup for those services.
- Communication: The Communication Functional Group serves as an abstraction that models the interaction schemes providing a common interface to the IoT Service Functional Group.
- Security: This Functional Group makes sure security and privacy requirements are met regarding the IoT system.

The Functional Groups are composed of functional components. IoT-A describes the components interfaces and functions, providing a means to design the interactions among the components; what is known as the information view.

Through the process of Functional Decomposition (FD) the different components that make up the IoT ARM are identified and related. Its purpose is to divide the complexity of the system into manageable parts that can be understood and to easy illustrating the relationships among them. FD produces a superset of functionalities that can be used to build the IoT System. The outputs of the Functional Decomposition are the Functional Model and the Functional View.

The Functional View describes the system's runtime Functional Components, including the components responsibilities, their default functions, their interfaces, and their primary interactions. The Functional View is derived from the Functional Model on the one hand and from the Unified Requirements on the other hand.

2.1.5 Information view

The information view addresses the flow of data among the different functional components. IoT-A gives a predefined set of common interactions between two entities: client and server. The goal is to simplify the view's complexity. The different interactions are outlined as follows:

- Request/Response: This is a synchronous communication model between two entities.
- Push (data): This is a one-way communication model that goes from the server to the client.
- Subscribe/Notify: This is an asynchronous communication model between two entities. The client first indicates a preference in some service offered by the server, and the server in turn sends notifications when new information is available.
- Publish/Subscribe: This communication model enables the server to advertise services on a third entity, called broker, making it available to the clients. Clients, on the other hand, can express their interest in some kinds of information to the broker; this entity allows the decoupling of the source of information (server) and the receiver (client) in such a way that information can still be exchanged.

2.1.6 Operational and Deployment view

The purpose of these views are -once the other views are described, interactions explained, and all the functional elements identified- to select the technologies in IoT that will enable the realization of the IoT system in a given deployment. This selection process identifies the technologies that will be used for the three element groups of the IoT Domain model: devices, resources and services. Each one of the element groups being a challenge in the sense that with the technologies defined for each of the three groups, the operational features offered by the system are finally defined.

As an example of this, there are several technologies that are especially suited for IoT. These technologies go from the myriad of sensors and actuators, now available, and the ones that will be; hardware that may not be even invented yet. The networks that interconnect those devices and other technologies such as Radio Frequency Identifiers (RFID), tags, or smart cards, cellular networks or Wi-Fi that can be used for communication, etc.

At this point, in a manner of speaking, the generators of data are defined, how they are going to communicate is also depicted (IoT-A identifies the Internet Protocol IP as a preferable design option given that no other requirements disallow such a choice), so what is next is to make this information available through services and resources. This is software that will be deployed in different levels, on the things themselves (smart objects), on gateways, or could be placed in the cloud.

Once the information is available, it has to be stored somewhere, and different possibilities also emerge, as it can be stored locally, on the web, etc. The use of the stored information, through aggregation, or on demand, allows for the creation of services and virtual entities that provide a wide range of possibilities. A third party in charge of managing some information can offer these services, and Application Programming Interfaces (APIs) can be provided to access those services.

To summarize, the operational and deployment view gives the overall picture of the system and the architecture, though not everything is contemplated in them, as the understanding of the other views, is paramount to be able to perceive the system as a whole.

In the next sections, we follow the functional view of ARM architecture to describe the components of SMARTIE. Moreover, the integration of the Smartdata Platform is presented. Finally, it shows a broad view of the whole functionality of SMARTIE components using a set of predefined interactions.

3 SMARTIE Architecture

3.1 Functional View

As described in Section 2 regarding the IoT-A ARM, the functional view is used to describe a set of functional groups and functional components that are derived from the requirements process and which can be found in several IoT systems. Figure 4 shows how the SMARTIE components are organized in the Functional Groups.

Following the figure, we describe each SMARTIE component in relation with its Functional Group and also the interaction with other components. The focus in this section's descriptions is on the reasoning of the placement in functional groups and design decisions for the particular component.

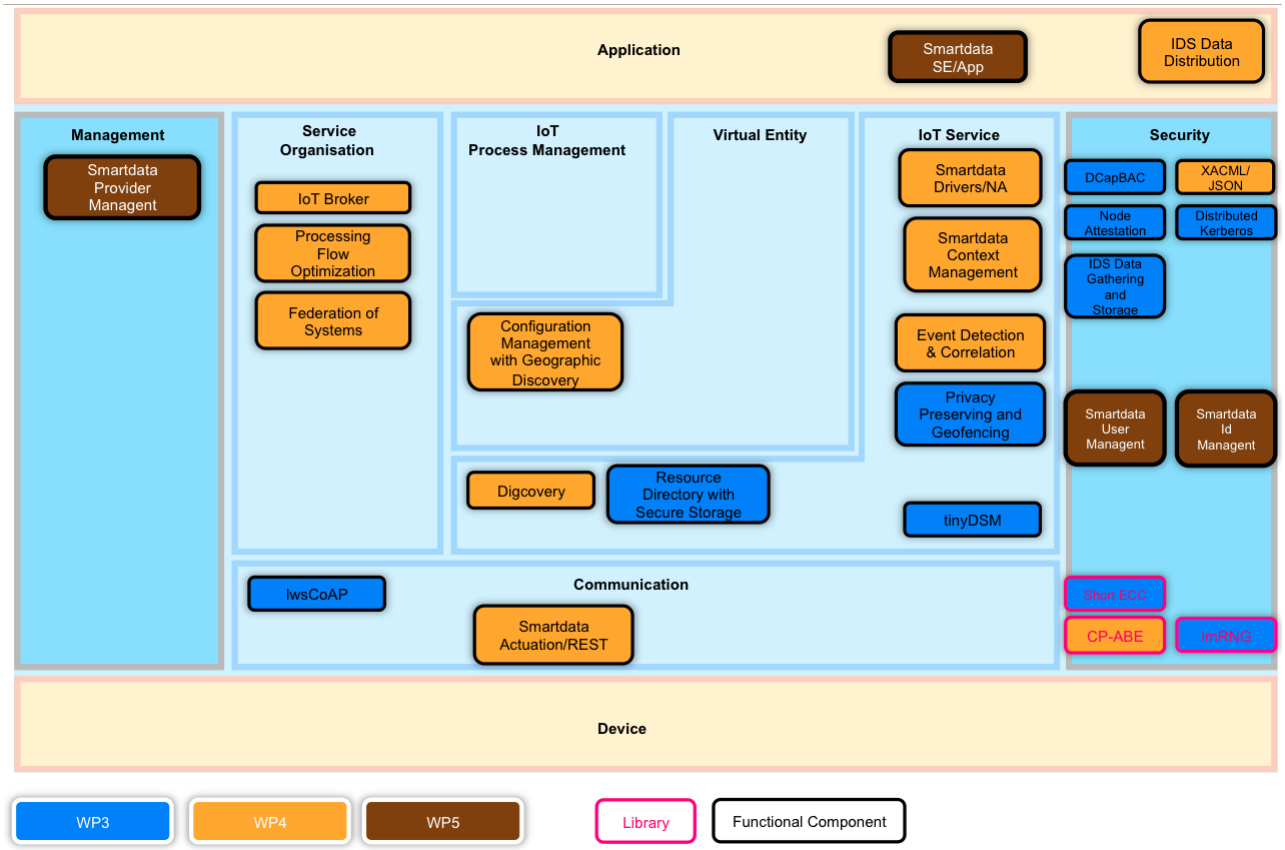

Figure 4: Components for each functional group

3.1.1 Application Functional Group

Although the Application Functional Group is out of the scope of the IoT-A ARM, we present the components that offer high-level functionality to the IoT system. In the SMARTIE project, the Application group is composed by IDS Data Distribution and Smartdata SE/App.

3.1.1.1 IDS data distribution

Functional Description	The IDS is used to scan the network traffic for intrusions to the network and to report unknown or unwanted traffic to the network operator. Therefore detected events are distributed to other devices.
Mapping to Functional	Application

Group	Provides security functions to applications / devices
Assumptions and Constraints	Implementation for low power devices
Key Design Decisions	Attacks on IoT devices have to be reported to other devices / the network operator. The data distribution parts of the IDS component immediately share new security relevant information, mainly using the lwsCoAP component or tinyDSM distributed storage features.
Relation to the Functional Group	The intrusion detection system (IDS) is a security component that offers advanced / additional protection from attacks on devices / protocols using monitoring / prediction mechanisms. The data distribution parts of the IDS component share security relevant information / functionality with other devices / applications / tasks.

3.1.1.2 Smartdata: SE/App

Functional Description	Contains the logic and algorithms responsible for producing relevant context for the application from raw messages. The algorithms may be of different types, depending on which is more pertinent to provide context in a generic way.
Mapping to Functional Group	Application Functional Group
Assumptions and Constraints	This components can be developed and manage by 3rd party entities. The component run in the platform but is not part of the platforms base components.
Key Design Decisions	These components can dynamically be added to the platform by 3rd party entities. There are components that run inside the platform, but are not part of the platform base components
Relation to the Functional Group	This component access the information gathered by the platform. These are mainly 3rd party applications, or services, running on top of the platform, that use, process, etc. the information collected by the platform. They could also order the platform to communicate with an actuator. Services provided by components as Privacy Preserving Event Detection and Correlation and PrivLoc may be incorporated in the platform as Service Enablers, enabling enhanced functionalities through interaction with other available SEs.

3.1.2 Management Functional Group

The Management Functional Group offers features to the IoT System features to reduce costs, attend unforeseeable usage issues, handling fault and adding flexibility to the overall system. In SMARTIE, the Management Functional Group contains the Provider Management of the Smartdata platform.

3.1.2.1 Smartdata: Provider Management

Functional Description	The Provider Management is a set of functionalities that gives other Smartdata components, or 3rd parties, the possibility to create, modify or delete providers. All the entities (i.e. sensors) that generate information that should be collected by the platform are identified as Providers. To send information to the platform, a Provider has to be registered previously; otherwise the information sent by the source is not taken into consideration.
Mapping to Functional Group	Management Functional Group

Assumptions and Constraints	The Provider has to be registered in the platform to be able to publish information to the platform. The registry can be made using the web portal interface, or by using the exposed API.
Key Design Decisions	N/A
Relation to the Functional Group	This component provides basic operational functionalities needed to correctly configure setup and manage the platform in terms of the Providers (i.e. sensors) that are authorized to send data to the platform. Smartdata Provider Management interacts with Directory Services – Resource Directory with Secure Storage and Digcovery. Functionalities as semantic based search may be provided by the component.

3.1.3 Service Organization Functional Group

This Functional Group acts as a communication hub among several components of other Functional Groups. As the basic communication concept in the IoT ARM, Service Organization composes services of different level of abstraction. In SMARTIE, this Functional Group is provided by 3 components: IoT Broker, Processing Flow Optimization and Federation of Systems.

3.1.3.1 Processing Flow Optimization

Functional Description	The Processing Flow Optimization creates processing flows according to an optimization criterion, taking into account constraints. In a processing flow, sources (e.g. sensing services on sensor nodes) provide information that is processed by one or more information processing services on computing nodes. The information then "flows" from the sources through the information processing services, providing the required information to the requester. The selection of sources and the placement of information processing services on computing nodes are optimized according to an optimization criterion, e.g. required bandwidth or required processing power. Computing nodes can, e.g., be sensor nodes, gateways, dedicated servers or cloud nodes. Constraints like trust or availability of security keys are taken into account. The IoT Broker in cases a request cannot be fulfilled by existing information sources can use processing Flow Optimization.
Mapping to Functional Group	Service Organization Functional Group
Assumptions and Constraints	The Processing Flow Optimization needs to be able to discover suitable sensor nodes with sensing services, processing service software components together with their requirements, and compute nodes, on which the processing service software can be deployed. The Configuration Management with Geographic Discovery can provide the former. The latter has to be developed as part of the Processing Flow Optimization. The information about the sensor nodes, processing service components and compute nodes has to provide the necessary information to enable the optimization and to check the constraints.
Key Design Decisions	To work with the IoT Broker, Input and Outputs of the information (processing) services have to be specified according to the Entity-Attribute model and the interfaces have to conform to the NGSI Context Interface specification [5]
Relation to the Functional Group	The Processing Flow Optimization belongs to the Service Organization Functional group as it orchestrates sensing services (on sensor nodes) and processing services (offered by information processors) so that information can flow from the services

	providing the required information to the requestor. It will be integrated with the IoT Broker, so the information is on a Virtual Entity level.

3.1.3.2 IoT Broker

Functional Description	The IoT Broker provides applications with one single access point for accessing IoT related information from a possibly large set of IoT sources like sensors, but also information processors. The IoT Broker accesses the information from the different sources, aggregates it and returns it to the requesting application. The IoT Broker supports synchronous one-time requests for accessing information, as well as asynchronous subscribe / notify interactions, where notification conditions specify when notifications (updates) are sent.
Mapping to Functional Group	Service Organization Functional Group
Assumptions and Constraints	It is assumed that information is provided by a set of distributed data sources. To find the required sources, the IoT Broker utilizes the Configuration Management with Geographic Discovery to which the sources are dynamically registered and deregistered. Information sources have to provide their information using the NGSI Data Model for context information [5]
Key Design Decisions	Information is provided using an Entity-Attribute model, which is implemented using the NGSI Context Interface [5]
Relation to the Functional Group	The IoT Broker belongs to the Service Organization Functional Group. As it follows an information-centric approach, all "orchestrated" services provide information on the Virtual Entity level. The services it interacts with belong to the Virtual Entity functional group and it makes use of the Configuration Management with Geographic Discovery

3.1.3.3 Federation of Systems

Functional Description	The Federation of Systems (FoS) allows temporary cooperation of independent systems in order to provide a given service, which cannot be provided by the involved sub-systems separately. The Federation of Systems will employ the components we develop in WP3 and WP4. It will be related to the Smart City Information Centre Use Case described in D2.1 [16]. It is an integrative use case showing the possibilities of cooperation of independent systems deployed in the Smart City in order to provide new functionalities. We intend to employ the Federation of Systems in order to enable the Smart City Information Centre to get the information from the temperature sensors deployed in different sub-systems (temperature sensors are available in our Smart Energy Management and Traffic Management scenarios). The Federation of Systems requires the tools looking for devices and services matching its requirements (our Digcovery, IoT Broker and Configuration Management and Resource Directory components enable device/service discovery and access). Also the authorized access control needs to be ensured (our Distributed Kerberos and DCapBAC solutions are responsible for access control). The communication within the FoS needs to be secured in order to ensure that only intended recipients will be capable to retrieve the information (our CP-ABE component and shortECC+lmRNG libraries can be used to secure the communication within the FoS). The FoS system consists of independent sub-

	systems that do not necessarily have the detailed information about each other, thus it is important to be able to detect the malicious or tampered devices and also the anomalies in the network traffic (our Node Attestation and Intrusion Detection components are responsible for these tasks). Since the main task of the Federation of Systems is to observe some phenomena (temperature in our case) it would be an advantage to detect the unusual measurements' values as soon as possible. Such an early detection is provided by our Privacy Preserving Event Detection and Correlation and by tinyDSM components. The Federation of Systems will provide an interface to the authorized entity, which will enable her to specify the requests, targeted to the FoS. The FoS's management layer will process the specified needs, which will be responsible for involving the appropriate SMARTIE components in order to fulfil the request.
Mapping to Functional Group	Service Organisation Functional Group
Assumptions and Constraints	Assumes existence of trust establishment and assessment mechanisms
Key Design Decisions	The Federation of Systems is built from the existing SMARTIE components, which interact in order to fulfil the FoS's requirements and requests.
Relation to the Functional Group	The Federation of Systems provides mechanisms that allow organizing the services offering required functionalities on demand. The services are organized from the components being parts of all Functional Groups.

3.1.4 Virtual Entity Functional Group

The Virtual Entity Functional Group provides functionality for interacting with the IoT System based on Virtual Entities, as well as functionality such as discovery and lookup of services that provide information about Virtual Entities. Furthermore, it enables functionality for managing associations and dynamic finding new associations and monitoring their validity. In SMARTIE, the Virtual Entity Group contains the Configuration Management with Geographic Discovery.

3.1.4.1 Configuration Management with Geographic Discovery

Functional Description	The Configuration Management allows the look-up and discovery of information sources, e.g. sensing services that provide certain information, e.g. the indoor temperature of a room or the speed of a bus. For the discovery, a geographic scope can be specified using geographic coordinates, e.g. discover all services providing information about busses that are currently within a geographic area given by a geographic segment specified by the geographic coordinates of two opposite corners. The IoT Broker and the Processing Flow Optimization use the Configuration Management.
Mapping to Functional Group	Virtual Entity Functional Group
Assumptions and Constraints	Information sources register with the Configuration Management, specifying what information they can provide. The information is specified according to the NGSI Data Model [5] for context information, e.g. I can provide the indoor temperature of room24. To support geographic discovery the geographic area for which information is provided has to be given. It can also be specified that an information source provide information for a type of entity, e.g. the location of all busses within the geographic area that represents downtown Novi Sad.

Key Design Decisions	Information is provided using an Entity-Attribute model, which is implemented using the NGSI Context Interface [5]
Relation to the Functional Group	The Configuration Management with Geographic Discovery is part of the Virtual Entity Functional Group, and more specifically it represent an instantiation of the VE Resolution Functional Component as described in the IoT Functional View presented in the IoT-A Deliverable D1.5 [2]. It allows the look-up and discovery of services related to virtual entities, e.g. a service providing the indoor temperature of a specific virtual entity of the type room. Geographic location specified as geographic coordinates can be used to scope the discovery

3.1.5 IoT Service Functional Group

The IoT Service Functional Group enables functionalities and services such as discovery, name resolution and look-up of IoT services. In SMARTIE, the IoT Service Group is composed by the following components: Digcovery, Resource Directory with Secure Storage, tinyDSM, Privacy-Preserving Correlation and Privacy-Preserving Geofencing, and also the Drivers/NA and Context Management of the Smartdata platform.

3.1.5.1 Digcovery

Functional Description	Digcovery is a framework that enables scalable and automated registration of devices for IoT environments using well-known protocols such as CoAP [6] and CoAP-RD. Other features offered by Digcovery are search and the use geospatial information to enhance the search capabilities of the framework.
Mapping to Functional Group	IoT Service Functional Group
Assumptions and Constraints	A trusted authority will be relied on to provided authentication of the devices that will enter the platform.
Key Design Decisions	A common and previously agreed semantic has to be used in order to homogenize functionality and capabilities of the sensors.
Relation to the Functional Group	Digcovery offers functionality as repository of information for the different things in the system that provide service. An example of this can be a sensor. Digcovery then allows keeping the information about the things updated and available to the users and the overall system. This service is of the upmost importance since the IoT system will rely on the services offered to complete needs to be accomplished. The use of CoAP as protocol for interacting with Digcovery enables the constrained devices to communicate with Digcovery and obtain information about services they might need.

3.1.5.2 Resource Directory with secure storage and user authentication

Functional Description	The RD [11] component provides information on the available resources and service within the system that has been previously registered end enables efficient search and discovery in a secure manner. It relies on the secure storage and token-based user authentication and authorisation components in order to enable secure access to the dynamic information about all available resources in the system at a given time only allowing the access to the users and components having sufficient rights to access the required information. The resources are registered in the RD

	using specific message in XML format using HTTP POST. This message contains meta-data about the resource and/or service, capabilities, location and other relevant data. Subsequent search and discovery is performed using XML message sent using HTTP GET containing appropriate filters, indicating desired services or resources. The core of the RD functionality is implemented in JAVA technology in combination with a MySQL database and encrypted data for persistent storage of data. Transactions are executed in stored procedures. The RD objects are passed through JAVA DAO (Data Access Object) class to stored procedures for data management.
Mapping to Functional Group	IoT service
Assumptions and Constraints	Current implementation of search and discovery components do not take into account security aspects of the information stored within these modules. Secure storage mechanisms and users authentication and authorisation methods will be employed as a central part in order to enhance the security aspects of the RD. This component relies on the cryptographic primitives and authentication and authorization components within the system.
Key Design Decisions	The secure storage will be based on defining an optimal data set to be encrypted within the MySQL database, which will not deteriorate the overall performance of the search and discovery process. Furthermore, the existing web services for accessing the RD will be extended to handle the token-based authentication and authorisation mechanisms.
Relation to the Functional Group	In order to support efficient and secure search and discovery, RD with secure storage and user authentication is provided. In this way, the meta information associated to services and resources is securely stored and access to it only allowed to the authorised users.

3.1.5.3 tinyDSM

Functional Description	The tinyDSM middleware provides a shared data storage for the nodes in the wireless sensor network. The data pieces are defined as variables, each with a specified type and policy controlling the behaviour of the middleware regarding handling of the variable. The sharing of the variables is based on replication and the policies specify its details in terms of the replication area and consistency parameters. The replication helps to assure the data availability in case of node failure and provides a consistent view of the replicas. The configuration is application specific, static and common for all the nodes in the network. Exchanging the policy _file allows creating different versions based on the same application code. A reliable shared data storage with data monitoring makes it possible for the nodes to cooperate more autonomic and independent from a central station by injecting more intelligence into the network.
Mapping to Functional Group	IoT Service Functional Group
Assumptions and Constraints	It is designed for low power devices.
Key Design Decisions	The tinyDSM middleware is implemented in pure C programming language and provides a clear interface to both, the underlying operating system and the application on top of it. Encapsulated in an OS adaptation layer (wrapper) it can work with all C based operating systems, and thus, it supports heterogeneity in both, hardware and software (or OS) dimension.
Relation to the Functional Group	tinyDSM service provides distributed shared memory and event detection for wireless sensor networks. Reliable shared data storage with data monitoring makes it possible for the nodes to cooperate more autonomic and independent

	from the central station by injecting more intelligence into the network. The service provides also data redundancy in WSN.

3.1.5.4 Privacy-preserving Geofencing

Functional Description	PrivLoc offers secure location-based services, in particular a secure geo-fencing service. Location-based services are increasingly gaining importance. Not only end users but also companies can make use of location data to track assets (e.g. public transport services, users looking for transportation or logistics companies). PrivLoc uses a standard spatial database for geo-locations (e.g. postgreSQL, MySQL, mongoDB) without modifications. The core of PrivLoc is a lightweight encryption procedure for location data that preserves operations like intersecting geometric objects and nearest neighbour search within proximity. In a typical geofencing setting users can subscribe to an area in order to get notification if any of the tracked objects enters or leaves the area. A spatial database hosted on the cloud receives regular movement vectors the objects, and checks if these movements cross a geofenced area which has been subscribed to within the database.
Mapping to Functional Group	IoT Service Functional Group
Assumptions and Constraints	The component requires the availability of a trusted proxy for location reports and subscribes operations.
Key Design Decisions	Location privacy of users and participating enterprises should be protected, i.e. tracking of devices sending locations updates must be impossible. We aim for a solution that works together with standard spatial databases and does not need any modification on the server or platform side. Only pre- and post-processing of the data is possible. Any overhead incurred by the solution on users and the proxy must be smaller than hosting a spatial database in a private data centre.
Relation to the Functional Group	PrivLoc is assigned to the IoT Service Functional Group, because it provides location-based service called geofencing. The core of PrivLoc is a scrambling scheme for location information that allows location information to be processed while scrambled. Compared to state-of-the-art location privacy schemes, PrivLoc maintains exactly information without adding noise and achieves high privacy by a permutation on a coarse granular level.

3.1.5.5 Privacy-preserving event detection and correlation

Functional Description	This component allows privacy preserving processing of sensor data, e.g. in order to detect events. An example would consist of an intelligent and efficient power distribution according to the location and number of participants of the scheduled events on citizens' agendas (for instance at a building level appropriately switching on and off the air conditioning systems in different floors of an offices building according to the scheduled meetings in different rooms). The great success of this idea will not materialize just with a mere gathering of the information generated by the multiple sensors deployed everywhere in a smart city. An intelligent and powerful correlation amongst such enormous amount of data is definitely a must to fulfil such a goal. By linking, grouping, associating and/or clustering all the collected data given certain criteria, more meaningful and

	useful information, such as behavioural patterns, for instance, could be extracted. In order to preserve the citizens' privacy in the context of smart cities, we suggest to encrypt the data before being collected, in such a special way that its original content is never revealed to the information gathering entity, but it is yet usable for certain purposes. In particular, it should be possible to perform a number of operations devoted to achieve a smart and useful correlation of such encrypted data. To this end, some alternatives that we would like to analyse and possibly leverage would be, amongst others: i) Cryptographic techniques such as searchable encryption or private set intersection, ii) Clustering algorithms, such as k-means and k-nearest neighbours for encrypted data,
Mapping to Functional Group	IoT Service Functional Group
Assumptions and Constraints	The functionality that can be realized currently is limited compared to plaintext processing. Today, mainly keyword search on encrypted data and private set intersection is practical.
Key Design Decisions	Information should be end-to-end protected and therefore processed while the data is encrypted.
Relation to the Functional Group	Privacy-preserving event detection and correlation is part of the IoT Service Functional Group, because it provides a service that processes IoT data to detect events or correlated data. The scheme protects the data by encryption while allowing certain operations being executed on encrypted data, or limiting the data that is leaked during decryption and processing. Existing solutions only encrypt information during storage and transmission, while privacy-preserving event detection and correlation allows keeping data encrypted end-to-end.

3.1.5.6 Smartdata: Drivers/NA

Functional Description	A driver is a component able to translate the information (i.e. the payload of the information sent by a COAP sensor) to a well-known language understandable and manageable by the platform. By message interpretation from different sensors, its persistence is guaranteed.
Mapping to Functional Group	IoT Service Functional Group
Assumptions and Constraints	At least a driver, able to understand and translate the information sent by the sensors, has to exist.
Key Design Decisions	Drivers are components that can dynamically be added to the platform by 3rd party entities. There are components that run inside the platform, but are not part of the platform base components.
Relation to the Functional Group	The Drivers/NA component is a "low level" component working at the IoT layer, once it provides the translation mechanisms between the platform and the sensors.

3.1.5.7 Smartdata: Ctx management

Functional Description	The Ctx Management component deals with the routing mechanism necessary to move the messages from the module that receives it to the module that should consume the message.
Mapping to Functional Group	IoT Service Functional Group
Assumptions and Constraints	All the messages that go through this component are stored in the big data infrastructure.
Key Design Decisions	N/A
Relation to the Functional Group	The Ctx Management component deals essential with the data communication between the providers (i.e. sensors) and the "higher layer" platform components, such as the SEs. It provides the queuing and routing functionalities. Ctx Management uses the components in the Security Functional Group to guarantee that the providers are authenticated and authorized to publish data to the platform.

3.1.6 Security Functional Group

The Security Functional Group is in charge of enabling privacy and security of the IoT system. In SMARTIE, this is the most important Functional Group that contains many relevant components such as DCapBAC, XACML/JSON, IDS Data Gathering and Storage, Distributed Kerberos, Event Detection and Correlation, ShortECC, CP-ABE, lmRNG and the Smartdata Components User Management and Id Management.

3.1.6.1 DCapBAC

Functional Description	DCapBAC is an authorization scheme that takes access control decisions before the actual service is accessed. It does this by giving a signed authorization token to a user who is asking for any particular service or functionality offered by a thing. The authorization token is sent along with a request to the thing that verifies the validity of the request and the authorization token, delivering the requested data, if successful.
Mapping to Functional Group	Security Functional Group
Assumptions and Constraints	The token is signed by a platform, which has to be trusted by both parties. The platform will also issue the credentials for the user to sign the requests.
Key Design Decisions	Uses public key cryptography where token is signed by private key of the DCapBAC component and the request by the private key of the user.
Relation to the Functional Group	The DCapBAC component integrates the Security Functional Group and provides operations as distributed authorization which enables not only the task of enforcing the authorization decision to the source of information, but also relieve the load upon a centralized entity that would, in other case, take upon the responsibility of resolving the authorization and maybe enforce it. This component then provides characteristics that a distributed environment as an IoT System can leverage in terms of security functionality.

3.1.6.2 XACML/JSON

Functional Description	XACML/JSON [20] is a framework that is consistent with Attribute Based Access Control (ABAC) [19] that uses policies to express rich and fine-grained access control access control decisions.
Mapping to Functional Group	Security Functional Group
Assumptions and Constraints	Policies have to be managed in an efficient way to handle all rules and policies, having into account possible conflicts, duplication, etc. among them.
Key Design Decisions	A centralized entity would take care of managing, validate, and integrate new policies into the system.
Relation to the Functional Group	XACML pertains to the Security functional Group and it is also of great importance for the correct function of the overall system. Access Control is a function that regardless of the system has to be present to avoid uncontrolled and unsupervised usage of the resources. Furthermore, IoT Systems are distributed and need context aware information to be able to cope with the myriad of situations that can arise. XACML provides access control functionality following the current state of the art standard in this area. Moreover, the constrained characteristic of some of the devices, and the amount of requests that are expected of IoT Systems demand of not only efficient context aware Access Control Systems, but also efficient. This is why the use of JSON in conjunction with XACML is considered as part of the functionality of this component.

3.1.6.3 IDS Data Gathering and Storage

Functional Description	The IDS is used to scan the network traffic for intrusions to the network and to report unknown or unwanted traffic to the network operator. There for it gathers and stores data to build a knowledge base for detection.
Mapping to Functional Group	Security Functional Group Provides security functions to applications / devices
Assumptions and Constraints	Implementation for wireless sensor networks
Key Design Decisions	IoT devices will likely be exposed to attacks from the internet. Encryption and authentication mechanisms may not be sufficient to protect those kinds of distributed low power devices. The IDS component detects on-going attacks on IoT devices. Therefore, the data gathering and storage parts of the IDS component scans the network traffic (mainly from lwsCoAP component) for intrusions to the network / devices and may correlate possible events from other event sources (applications / tasks) for security relevant data. It gathers and stores part of the security relevant data to build a knowledge base for further detection. Data storage may include local storage possibilities as well as distributed tinyDSM storage features.
Relation to the Functional Group	The intrusion detection system (IDS) is a security component that offers advanced / additional protection from attacks on devices / protocols using monitoring / prediction mechanisms. The data gathering and storage parts of the IDS component gather data from network interfaces and / or events from other applications / tasks.

3.1.6.4 Node Attestation (IMASC)

Functional Description	IMASC is an integrity measurement framework that makes use of a cheap and common secure hardware, the SmartCard, to provide a trusted running environment for most of the embedded devices that has suitable port support. Since the SmartCard is a secure microcontroller that is very difficult for the attacker to hack, we use it as a trust anchor in the architecture of the device node to maintain the integrity of its software stack. The firmware of the device will be tailored to a measure-before launch execution scheme, so that tampered code or unknown libraries will be detected and audited before the device decided to execute it. Meanwhile, the SmartCard protects the integrity of the measurement results so that a Trusted Third Party can attest each node remotely. IMASC also provides flexibility for the existing device platforms. Any devices with USB port, or SD card slot, or SmartCard reader slot can be extended and have IMASC running on top.
Mapping to Functional Group	Security Functional Group
Assumptions and Constraints	The device needs to have the possibility to connect to a smart card, e.g. via SD card slot or USB. The device operating system needs to include the Linux IMA kernel module first included in the 2.6.30 kernel.
Key Design Decisions	Attestation and integrity measurements should be based on trusted hardware that is lightweight compared to a trusted platform module.
Relation to the Functional Group	IMASC is part of the Security Functional Group, because it provides efficient security for the devices. An IoT system is only as secure as its weakest component. Commonly the sensors are very restricted devices and only implement limited protection mechanism. State-of-the-art solutions such as the trusted platform module are too complex to be integrated in most devices. Pure software solutions do not provide high security. IMASC provides a practical intermediate solution to improve the device security.

3.1.6.5 Distributed Kerberos

Functional Description	Distributed Kerberos is a component to compute Kerberos-like authentication tokens. The service can be enriched with authorisation, e.g. with capabilities based on the DCapBAC module to obtain lightweight capability tokens for authentication and authorisation. With symmetric cryptography, the choice of suitable authentication and authorization schemes is limited. A popular choice is the Kerberos protocol, which is however designed for an enterprise scenario, where all resources and the central Kerberos servers are in the same domain. Distributed Kerberos increases the trust in the Kerberos server by distributing it over two entities, e.g. two servers or even two organisations. As long as at most one server part is compromised, all obtained Kerberos tokens follow the access control policy. For a successful attack, both servers would need to be compromised. Therefore, Distributed Kerberos can be used as a publicly available service across domains.
Mapping to Functional Group	Security Functional Group
Assumptions and Constraints	The access-controlled device is a restricted device that cannot manage user accounts or deal with public key cryptography. The component requires that the user requesting access to a device has access to the platform/cloud.

Key Design Decisions	The main design guideline for this component is to prevent that any single unauthorized entity is able to obtain a capability, thus a distributed design is necessary. The component should only use symmetric cryptographic primitives.
Relation to the Functional Group	Distributed Kerberos provides authentication and authorisation which are core functionalities in the security functional group, additionally a key exchange between user and device is realised. The Kerberos service is used to generate access tokens to devices. The distributed implementation of the service offers higher security against key compromise compared to current solutions.

3.1.6.6 ShortECC

Functional Description	shortECC library provides security mechanisms, i.e. encryption and digital signature for low constrained devices. The approach uses elliptic curves with key length between 32 and 64 bits, but can also use standard ECC key lengths.
Mapping to Functional Group	Security Functional Group Used by those Functional Components that require encryption or digital signature
Assumptions and Constraints	It is designed for low power devices. It is used in trusted environment. It requires powerful device for generation and distribution of new shortECC parameters.
Key Design Decisions	Uses tinyDSM for key management. Uses lmRNG in order to generate the pseudorandom numbers.
Relation to the Functional Group	The shortECC library is part of the Security Functional Group. It provides the encryption and digital signature functionalities for low power devices, e.g. wireless sensor nodes. It can be used by another Functional Components requiring the mentioned functionalities.

3.1.6.7 lmRNG

Functional Description	LmRNG library provides a lightweight approach for generation of the cryptographic secure pseudorandom numbers. The numbers are generated on computationally constrained devices.
Mapping to Functional Group	Security Functional Group Used by those Functional Components that require pseudorandom numbers.
Assumptions and Constraints	It is designed for low power wireless sensor networks. It requires ADC for generation of the seed.
Key Design Decisions	A hardware module does the seed generation. Its outputs need to fulfil the NIST [14][15] requirements according to entropy sources. The modular arithmetic performed by the lmRNG needs to be adjusted to the finite field order in which the cryptographic operations are performed.
Relation to the Functional Group	The lmRNG generator is part of the Security Functional Group. It allows generation of cryptographic secure pseudorandom numbers on low constrained devices, e.g. wireless sensor nodes. It can be used by security libraries, e.g. shortECC or by another Functional Components requiring pseudorandom numbers.

3.1.6.8 CP-ABE

Functional Description	CP-ABE [17] is an encryption schema that ciphers information under a certain policy of attributes, and the keys of the users are associated with sets of attributes. This enables dissemination of information using different information sharing models where the producer of the information is always on control of how the information is accessed.
Mapping to Functional Group	Security Functional Group
Assumptions and Constraints	Constrained devices nowadays are not powerful enough to handle this kind of ciphering schema.
Key Design Decisions	Ciphering with this schema will be done by a central entity, powerful enough to handle the computational requirements of the ciphering primitives used by this component.
Relation to the Functional Group	CP-ABE is included, as part of the Security Functional Group, because IoT Systems need not only effective ways of coping with security issues but also efficient ones. In this sense, The classic security schemes such as SKC and PKC, while they provide proven security primitives, they may not be suitable for distributed systems as IoT. This is why the use of CP-ABE is an interesting improvement to the other ciphering schemes. CP-ABE allows ciphering the information, as explained above, in a way it can be only ciphered with the policies previously defined, and all users whose keys satisfy the policy are able to decipher the information. This not only saves resources, but might also be well related with the attribute-based nature of XACML, giving consistency to the security functionality.

3.1.6.9 Smartdata: Id management

Functional Description	This component provides authentication and authorization functionalities. It guarantees the authentication of each entity and issues the needed tokens after a successful authentication. All the services or components need to have a token in order to perform request to the platform. The authorization provides access control at database level, as well as, a service to evaluate (PDP) the access control policies to all platform's components.
Mapping to Functional Group	Security Functional Group
Assumptions and Constraints	All the services or components need to have a token in order to perform request to the platform.
Key Design Decisions	OAuth 2.0 and OpenID Connect are used for authentication and access token management. The authorization is evaluated through the access token generated during the authentication process.
Relation to the Functional Group	The Id Management includes the functionalities related with the authentication – token management – and authorization used by the platform components. It guarantees the needed identification, authentication and authorization to all the components that communicate with the platform.

	This component incorporates • XACML/JSON - provide a more lightweight approach to the platform PDP • CP-ABE - although being an encryption mechanism, the Id management can incorporate part of it, namely responsibility for managing the keys • DCapBAC - provides the platform with a lightweight token generation in order to provide authorization to more constrained devices • Distributed Kerberos -enables platform authentication for constrained devices With the integration of these components, Smartdata Id management can be a security provider for all services in the SMARTIE architecture, being able to authenticate and authorize constrained and non-constrained devices.

3.1.6.10 Smartdata: User management

Functional Description	This component governs the user related functionalities, in terms of creation, modification or removal. It guarantees a unique credential store for users that need to access the platform. Moreover it also manages the information related with services credentials and attributes.
Mapping to Functional Group	Security Functional Group
Assumptions and Constraints	All the users and services that need to use the platform should exist in the User Management databases.
Key Design Decisions	The user's attributes and credentials store is based on a SQL Database
Relation to the Functional Group	The User Management component is composed by a user's attribute and credential stores and the services needed to manage those stores. There is a close relationship between this component and the Id Management, as it provides access to the attributes and credential stores through services. This component is essentially a security component.

3.1.7 Communication Functional Group

The Communication Functional Group is an abstraction used for modelling the variety of scheme interactions derived from several technologies belonging to IoT Systems. This Functional Group also provides a common interface to the IoT Service Functional Group. In SMARTIE, the Communication Group is composed by lwsCoAP and also the Actuation/REST component of the Smartdata platform.

3.1.7.1 lwsCoAP

Functional Description	Light weight Secure CoAP (lwsCoAP) component is used to provide secure data channel between the IoT devices and the backend cloud platform utilising secure channel and employing light-weight encryption schemes. It is building on the CoAP [6] which is an application layer protocol designed to lower the complexity for the constrained networks but, also, to enable communication over the existing internet infrastructure. The core of the security system is the cryptographic primitive based on ECC [7], which can be successfully scaled up and down to provide variable level of protection at the expense of using more or less resources (i.e. processing power, memory, generated overhead). The solution is based on

	ISO/IEC 29192 standards [8], which aim to provide lightweight cryptography for constrained devices, including, block and stream ciphers and asymmetric mechanisms. This method is further optimized in order to reduce the key size and make the algorithm more efficient in terms of computational requirements and still provide the satisfactory level of the security.
Mapping to Functional Group	Communication Functional Group
Assumptions and Constraints	Existing secure CoAP mandates the use of datagram transport layer security (DTLS) [9] as the underlying security protocol with associated encryption schemes such as AES for authenticated and confidential communication. DTLS, however, was originally designed for comparably powerful devices that are interconnected via reliable, high-bandwidth links, which is often not the case. The challenge is to deploy simplified encryption methods as part of the CoAP, which can be executed on very constrained devices.
Key Design Decisions	Major design decision is to utilise the existing CoAP implementation such as Californium or libCoAP[10] and extend it with the new set of encryption methods based on ECC encryption schemes. The solution will still use the DTLS channel for the secure transfer, but the cryptographic primitive will be replaced with the appropriate light-weight counterpart.
Relation to the Functional Group	The lwsCoAP component is part of the Communication Functional Group providing features for the secure data transfer between IoT devices and back-end infrastructure. This component provides secure transfer of data with support of light encryption mechanisms within one of the major standardised communication protocol within the IoT domain. The advantage of this component is the ability to be deployed within the constrained environments such as IoT devices often having limited computation power, limited power (such as ones that are battery operated) and limited memory resources.

3.1.7.2 Smartdata: Actuation/REST

Functional Description	This component is responsible for the communication between the platform and the actuators. It translates the abstract message known by the platform to the specific message that it is understandable by the sensor
Mapping to Functional Group	Communication Service Functional Group
Assumptions and Constraints	This module has to exist in order for the platform to talk with the sensors.
Key Design Decisions	These components can dynamically be added to the platform by 3rd party entities. There are components that run inside the platform, but are not part of the platform base components
Relation to the Functional Group	This component is able to route the message from the platform (or from the SEs) to the actuators. Moreover it also translates the messages, from an abstract view to an actuator's understandable message.

3.2 Architecture Configurations

The SMARTIE architecture contains a number of components – described in the previous section – that have the same or at least similar functionalities, but differ in their characteristics, which make them suitable for different scenarios. Following from that, Section 5 shows multiple interactions for registration, discovery, retrieval, subscription involving different sets of components.

In this section, for a few high-level scenarios that have specific characteristic and requirements, we identify a set of components especially suited for the given setting. The scenarios only serve as examples, the WP6 use cases may decide on different sets of components as the result of a more thorough analysis. More complex scenarios may have multiple heterogeneous requirements, which may result in having different levels and a set of components for each level. In this case integration between the different levels is needed. This may also be the case for WP6 use case(s). We will continue to work on such integrated cases, identifying gaps and required "glue" among components. The resulting more complex architecture configuration(s) will be presented in the final SMARTIE architecture i.e. in deliverable D2.4.

3.2.1 Scenario 1 – Constrained Devices

Scenario 1 is characterized by a number of constrained devices, e.g. sensor nodes, and a limited not (heavily) constrained infrastructure, e.g. a single gateway. The constrained devices interact with the gateway, but also with each other to achieve the desired functionality. An example for such scenario could be environmental monitoring in a remote area with limited infrastructure.

The devices are constrained with respect to computational power, energy and communication bandwidth. For this reason, there are few suitable off-the-shelf components available that could be used, especially if specific security requirements are to be fulfilled. Table 1 shows a number of high-level functionalities required in this scenario. For each of these, the selected SMARTIE components and libraries are provided, together with an explanation why they have been selected.

Table 1 - Components for Scenario 1 – constrained devices

High-level Functionality	Components and Libraries	Explanation
Communication	lwsCOAP, lmRNG, shortECC, Smartdata Actuation/REST	Constrained devices have to securely communicate with existing infrastructure and among themselves, off-the-shelf secure communication is too expensive for constrained devices
Discovery	Digcovery, lwsCOAP	Sensors need to register themselves and find out about services, so the interaction with the discovery component must be light-weight
Storage	tinyDSM	Constrained devices need very light-weight storage
Device Management	Smartdata Provider Management	Enables to create, modify or delete providers. All devices, constrained or not constrained, must be registered in order to send data to the platform. Smartdata Provider Management incorporates other components as explained in section 3.1.2.1.
Access Control	DCapBAC, XACML/JSON, Smartdata Id management, Smartdata User	Constrained devices require light-weight and distributed mechanism to control the access of their services and resources. Smartdata Id management incorporates components to provide Access control to constrained and non-constrained

	management	devices as explained in detail in Section 3.1.6.9
Authentication	Distributed Kerberos Smartdata Id management, Smartdata User management	Kerberos tokens do not require public-key cryptography and can be processed on restricted nodes. All entities must be authenticated in order to access the platform. Smartdata Id management incorporates components to provide Access control to constrained and non-constrained devices as explained in detail in Section 3.1.6.9
Driver	Smartdata Drivers/NA	As many devices and protocols can be used in order to communicate with the platform, a driver is needed to translate the information to a well-known language in order to guarantee its persistence.
Routing	SmartdataCtx Management	A routing mechanism that moves the messages from the module that receives it to the module that consumes the message.
Information Processing	Smartdata SE/App	Processing the information from the application raw messages in order to provide relevant context for the applications (see Section3.1.1.2)

3.2.2 Scenario 2 – non-constrained IoT Services

Scenario 2 is characterized by IoT services based on devices with sensors and actuators. Devices are not (heavily) constrained, e.g. involving smart phones, and there is some infrastructure. There may be higher security requirements, e.g. a requirement for device attestation. An example for such scenario could be a medical home care application, where a smaller number of sensors and actuators are deployed and it is important that information is secured and tampering with devices does not go undetected. **Fehler! Verweisquelle konnte nicht gefunden werden.** shows a number of high-level functionalities required in this scenario. For each of these, the selected SMARTIE components and libraries are provided, together with an explanation why they have been selected.

Table 2 - Components for Scenario 2 – non-constrained IoT Services

High-level Functionality	Components and Libraries	Explanation
Communication	Off-the shelf secure communication, e.g. COAP, HTTPS, CP-ABE, Smartdata Actuation/REST	Envisioned devices can handle suitable cryptographic primitives and protocols.
Discovery	Resource Directory	Resource-level registrations suitable for IoT Services, a secure database as provided by the Resource Directory requires some infrastructure.
Device Attestation	IMASC	Non-constrained devices can support device attestation.
Device Management	Smartdata Provider Management	Enables to create, modify or delete providers. All devices, constrained or not constrainer, must be registered in order to send data to the platform. Smartdata Provider Management incorporates other components as explained in section 3.1.5.2.

Access Control	DCapBAC, XACML/JSON, Smartdata Id management, Smartdata user management	Non-constrained devices can support distributed access control. Smartdata Id management incorporates components to provide Access control to constrained and non-constrained devices as explained in detail in Section 3.1.6.9
Authentication	Smartdata Id management, Smartdata user management	All entities must be authenticated in order to access the platform. Smartdata Id management incorporates components to provide authentication to constrained and non-constrained devices as explained in detail in Section 3.1.6.9
Driver	Driver/NA	As many devices and protocols can be used in order to communicate with the platform, a driver is needed to translate the information to a well-known language in order to guarantee its persistence.
Routing	SmartdataCtx Management	A routing mechanism that moves the messages from the module that receives it to the module that consumes the message.
Information Processing	Smartdata SE/App	Processing the information from the application raw messages in order to provide relevant context for the applications (see Section 3.1.1.2)
Location Services	PrivLoc	If the device has a GPS or similar sensor for locating itself, it can encrypt the location using PrivLoc for processing the data in the cloud.

3.2.3 Scenario 3 – Large scale IoT Deployment with Virtual Entity level services

Scenario 3 is characterized as a large-scale IoT scenario with a strong infrastructure. A higher abstraction level is supported that allows discovering and querying information on a Virtual Entity level, e.g. request information like the occupancy of a room or the power consumption of a building instead of requesting the measured value of sensor A. Processing of information is required as aggregated results are requested and there are different players involved. Processing can be done on different nodes, including sensor nodes, gateways, special servers or the cloud, but there may be different trust levels, which has to be taken into account for selecting appropriate nodes for processing. An example for such a scenario could be a smart city scenario with the city itself, an energy provider, a public transport company etc.

Table 3 shows a number of high-level functionalities required in this scenario. For each of these, the selected SMARTIE components and libraries are provided, together with an explanation why they have been selected.

Table 3 - Components for Scenario 3 – Large scale IoT Deployment with Virtual Entity level services

High-level Functionality	Components and Libraries	Explanation
Communication	Off-the-shelf	No constraints, so existing secure communication can be used.

Discovery	IoT-Broker, Configuration Management	Configuration Management supports registration, look-up and discovery of VE-level services with geographic scopes; IoT Broker provides point of access for applications accessing information.
Processing	Processing Flow Optimization	Optimization of where processing takes place and how information flows through system, takes into account security constraints, e.g. trust level, availability of cryptographic keys.
Authentication	Smartdata Idmanagement	All entities must be authenticated in order to access the platform. Smartdata Id management incorporates components to provide authentication for virtual entities as explained in detail in Section 3.1.6.9
Event detection	Event detection and data correlation	Event-detection or data correlation on encrypted data must be set-up on large processing nodes, e.g. in a cloud infrastructure. Sensors or their virtual representation will encrypt the data.

Once the SMARTIE architecture is established following to the IoT-A guidelines and several scenarios with different computing requirements, we provide a detailed description of the Smart Data platform with its subcomponents and also the interoperability among all components of the SMARTIE project in the next sections.

4 Smartdata Platform

This section describes the Smartdata platform that will be used to support the experimental and real IoT scenarios test-beds. For the SMARTIE project, the Smartdata platform provides functionalities that extend the innovative components developed by the project, therefore allowing the project components to be tested in a complete end-to-end IoT environment. With the new components integrated, the platform provides a way to interface between different components where needed. Using an existing platform helps evaluating the feasibility of components described in Section 3 in a real IoT system and, consequently, assure their adequacy to provide stronger trust, security and privacy in the IoT world.

4.1 Smartdata High Level Architecture

The Smartdata platform is a service oriented architecture aligned with principles of service provisioning approach (offer and reutilization of services) from the TM Forum Service Delivery Framework, based on consumer-producer concepts where multiple processes synchronization is crucial. It provides a multi-tenant and open platform to collect, enquire, route and process information produced in an IoT domain. A Consumer is the entity for whom the information is relevant (i.e. a mobile application); a Producer is any entity that produces information (i.e. a sensor).

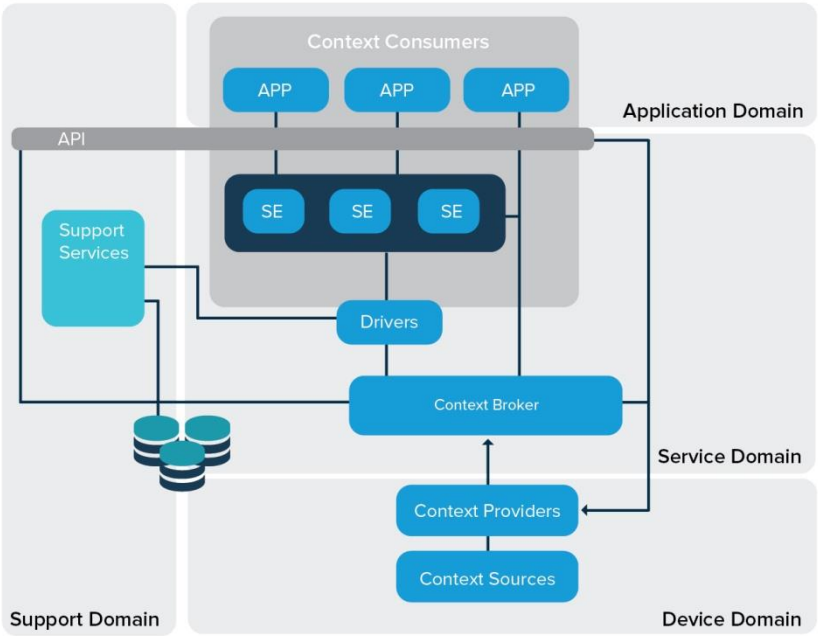

Figure 5: Smartdata High Level Architecture

The high level architecture presented in Figure 5 is intended to be generic, in order to support information originating from heterogeneous environments and several types of objects. The platform ecosystem is split into four domains: Device Domain, Service Domain and Application Domain.

- **Device Domain** - This block contains the sensors, devices and gateways and any stream of data that may be used as an input to the platform. An example of such streams is the information collected from the social network connectors, sensors (e.g., temperature, humidity, blood pressure,), set-top boxes (STB), internet APIs (e.g., Meteorological information), etc. Typically, gateways receive data from various sensors, execute a basic post-processing over the raw data and expose APIs, enabling information access agnostic to sensors associated technology, but dependent on API specification and used information model.
- **Service Domain** – is the platform core and is divided into Context Broker, Context Consumer and REST generic API:

- Communication with devices (Context Broker) - The southbound domain connects the platform to sensors, gateways, devices, etc. It implements several communication protocols in order to comply with different technologies, requirements and scenarios. This block not only contains the needed functionalities to route the data originating from the outside world to the platform, but is also responsible for translation, aggregation, Complex Event Processing (CEP) and creating messages with semantic meanings that can later be used internally by the services running on top of the platform.
- Service Enablers (SE) - Service enablers are functional atomic services that add functionalities to the platform. Service enablers are seen as units of functionalities that can be used to create composite services based on business rules. Smartdata platform has a set of built-in SE (aggregation, inference, machine learning and prediction), that can be used by 3rd party providers.
- Service Exposure - This is the northbound connection with information consumers, i.e., applications and services. Translates the semantic aware information that flows in the platform to the protocol in which the application is expecting the information.

- **Support Domain** - This block contains functionalities that are transversal to all the other modules. Those services are used in the platform based on business rules and include Storage, Real-Time Data query, Registry, SLA functionalities, Authentication, Authorization, Accounting and Auditing.
- **Application Domain** - This domain includes all applications and services that take advantage of the platform. This includes Mobile Apps, Web Apps, etc. Based on a real-time enrolment it is possible for a Smartdata customer to register a new application. That registry includes the information that the application or service expects to receive.

4.2 Technical Architecture

The Smartdata solution generic components were implemented with the technologies referred in Figure 6.

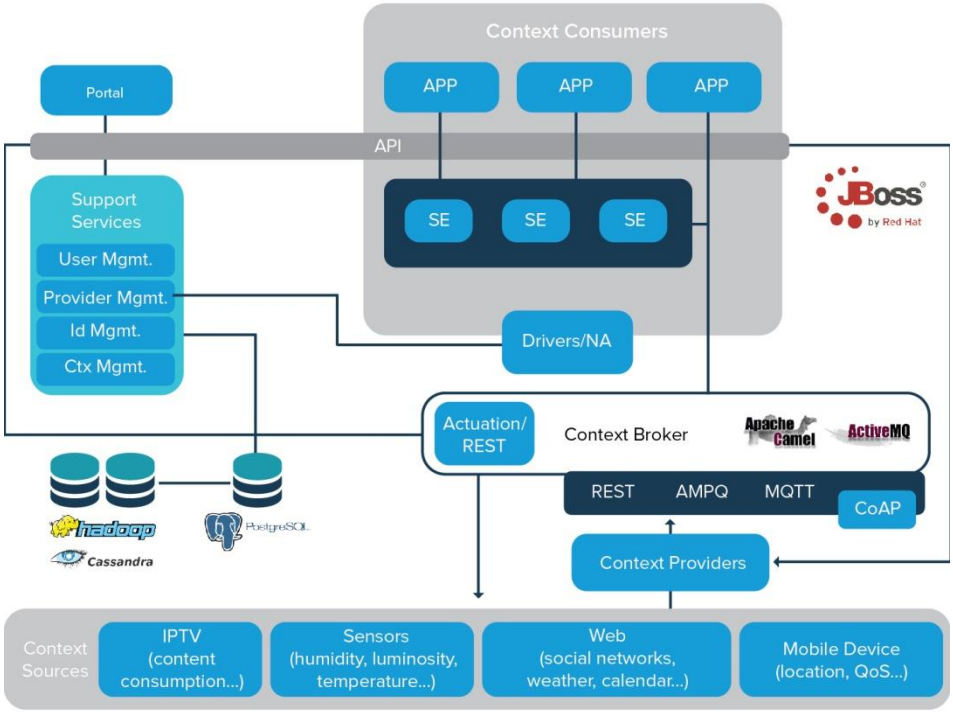

Figure 6: Smartdata Technical Architecture - Technologies

- **Support Services –** This component plays a fundamental role in the architecture. It is responsible for persisting all resources available in the API as well as all the messages received by the context broker. Moreover it contains services, related with the security, overall management, etc. used by the platform components.
- **Context Broker –** The communication between the sensors/gateways domain (Context Sources/Providers) and service domain (Context consumers) occurs through the context broker. It works as protocol adapter based on queue management and as a message forwarder. In order to provide support to requirements from different scenarios and, consequently, different messaging protocols, two message brokers were evaluated: ActiveMQ and RabbitMQ. ActiveMQ was chosen because, currently, it has the highest number of installations and integrates well with JAVA (JBOSS, TOMCAT, etc.). The message protocols supported by Smartdata are: MQTT, AMQP, and HTTP (REST). A COAP gateway was also implemented, since ActiveMQ does not natively support this protocol. If required by the provider, this block also has a security layer which provides communication and authentication tools to the resources. This module is also responsible for the message routing, through the creation of subscriptions in the ActiveMQ topics, using the pattern publish-subscribe, e.g., every message published to a topic is forwarded to all those who subscribed the topic. To provide a dynamic publish-subscribe environment, we rely on the Apache Camel framework. This framework provides libraries to define routes in several specific languages and through several message transport models, such as HTTP and JMS.
- **Portal –** Web Portal allows any Smartdata client to configure the platform according to its needs. This module provides tools to register and manage users, as well to all the resources available in their accounts. All accounts in the Smartdata platform are attached to a tenant (a client). Through this portal the user can manage new sensors, consumers, actuators and also how and to where the data should flow. It is also possible to consult data associated with each of its resources. Technologically, this portal provides an interface over the Smartdata management platform API using JavaScript (Ajax requests using the jQuery library). Protection mechanisms were implemented to secure the portal user keys - a Servlet was implemented to act as a "mediator" between JavaScript and the API. The primary keys reside only in the server side, therefore reducing its availability in external networks.
- **Driver/NA (Network Adapter) –** This modules act as sensor/gateway driver, e.g., allowing interpretation of the messages from different sensors, as well ensuring their persistence. A basic example of a NA is the one responsible for storing the message in the database - it is agnostic to the message format, being a generic driver that stores all messages as received. Other more semantically evolved NAs will be eventually required, depending on the gateways, their API complexity and information models.
- **SEs (Service Enablers) –** the SE blocks contain the logic and algorithms that process the raw messages, in order to produce relevant context for the application. These algorithms may be of different types - learning, prediction, statistics and inference - depending on which is more pertinent to the specific needs of the applications. The Apache Cassandra and Hadoop are used in the processing and storage (e.g. patterns identification, statistic treatment) of the produced information. SEs can feed the applications and/or databases directly, as well as produce more elaborate post-processed information from the raw data received from the sources.
- **Infrastructure –** it is based on Docker which is a technology in the virtualization area, creation, availability and execution of applications in a distributed way and independently from the operating system (OS). The traditional virtual machines contain applications (including binaries and libraries) and a complete OS, while a Docker container has the applications but not the OS; it only runs an isolated process in the main OS sharing the kernel with other containers. It can, therefore, take advantage of allocation and resources isolation of VMs and ensure greater efficiency and portability.

  With Docker it is assured that:

  - The platform might be easily reused/deployed in any environment (Portability);
  - Different versions of the application may be used (Version control system);

- Installed and configured tools can be used as base for other applications (Sharing).

Platform operation relies on two applications - one with the application server JBossWildfly and one other with the message broker Apache ActiveMQ. Two management applications are also available - continuous integration through Jenkins and monitoring through Nagios.

- **Storage -** A Hadoop ecosystem is used to implement the storage service. Hadoop is an open-source software framework for storage and large scale processing of data-sets on clusters, has modules like HDFS (Hadoop Distributed File System, a distributed file system that provides high-throughput access to application data) and MapReduce (Hadoop MapReduce is a YARN-based system for large data sets parallel processing). This service is able to handle a large amount of data in a distributed and scalable way, and provides MapReduce processing. With MapReduce we are able to process huge datasets, using a large number of computers with parallel and distributed algorithms.

4.3 Privacy and Security

4.3.1 Authentication

Two user authentication protocols were implemented - OAuth 2.0 and OpenID Connect. Users can choose one of these protocols to obtain a token to prove their identity.

The sensors/gateways can only use OAuth 2.0 protocol. However, as there are resources that are not able to send credentials, it is possible to communicate without authentication (only advised in mutual trust situations).

The user authentication to allow access to a resource, using OAuth 2.0, must follow the "resource owner password credentials" flow in the [host]/auth/oauth/token endpoint. In this model, the service sends a request with the user credentials together with an HTTP Authorization Header" containing the service credentials. If the user and service are valid the server will return a token to access the platform (access token) and a second a refresh token to be used to get a new access token. Due to its modularity, the internal modules, that use the exposed APIs internally, have to authenticate themselves when invoking the exposed APIs/functionalities.

As the OAuth protocol was initially designed only to support authorization, a new approach was proposed - OpenID Connect. One of the described modes is the "authorization code flow". In this mode of operation, the service redirects the user to an endpoint [host]/auth/opened/authorization (to be authenticated) for authentication. After a successful authentication, the user is redirected to the application together with a code. With that received information the application makes a request to the [host]/auth/openid/token endpoint together with a HTTP Authorization Header with its own credentials. Finally, the application will receive the OAuth access and refresh tokens along with the user's information.

4.3.2 Authorization

The Smartdata platform API is protected - a service is able to successfully invocate the API only after it includes in the request the header "authorization header", with the token received from the authentication process. Through this token the platform can identify which service is making the request and which user is using it; if it is an internal service the token will only be associated with a service.

The authentication process is made through a PTInS product IAM, which uses a role based access control (RBAC). That is, user and service permissions are made through roles (i.e. if the same role is assigned to a user and a sensor, they will have the same privileges). However, if the request originates from a service, the permissions only depend on its role but, if it originates from a user, it requires analysis of permission intersection between the user (or resource) and service roles.

4.3.3 Communication

The communication between the platform and the services is ensured with the secure protocol HTTPS which allows the creation of a secure channel between the platform and the service invoking the API.

The resources can communicate with the platform using HTTP, AMQP and MQTT; all these protocols can be secured using SSL. Communication through CoAP is also possible, but it will be insecure.

4.4 Supported Protocols

This section describes the different protocols available in the Smartdata platform, to allow the providers to send their messages (readings, values, etc.) to the platform itself. The platform supports the following protocols: MQTT, AMQP, CoAP and REST (HTTP), as shown in Table 4. All these protocols, except CoAP, support TLS for public-key encryption of the connection as well as an Authentication/Authorization of the endpoint.

TLS implies the use of certificates. Therefore, the endpoints that connect to the platform need the x.509 certificate of the platform so they can validate the identity of the platform.

The Authentication/Authorization process is the same as the one used by the other interfaces of the rest of the platform. It is based on credentials (username/password) saved in the database at the time the endpoint registered.

Table 4 - Protocols supported by the platform

Protocol	Transport	Encryption	Authentication/Authorization	PORT	PORT Enc.
MQTT	TCP	TLS	Credentials	1883	8883
AMQP	TCP	TLS	Credentials	5672	5671
COAP	UDP	DTLS*	Credentials*	5683	4.4.1
REST	TCP	DLTS	Credentials	8162	8161

*Not supported yet by the platform

5 Descriptions of interactions (Information View)

This section describes the typical interactions providing the basic functionality of SMARTIE. Each interaction shows different solutions depending on the computing requirements of the SMARTIE components included in them. Some interactions between components are shown to complement the functionality of others.

These interactions are:

- Registration of devices: This interaction shows how the devices are registered into the system to make their services available.
- Discovery: This interaction shows how a user is able to search for specific services he is interested in, obtaining as a result the list of sensors that match the aforementioned search. Along with this information will be available how the user can communicate with the sensor to interact with it.
- Retrieve: This interaction shows how the user is able to interact with a sensor once discovered in the previous interaction.
- Subscribe: This interaction shows how a user can express an interest in some kinds of information and receive that information as soon as it is generated.
- Actuation: This interaction shows, in a similar way as the one depicted in the retrieve interaction, how to modify or alter the state of the system.
- Event Detection: This interaction shows how information about the system being monitored is used to detect unusual behaviours and possible misuse of the system.
- IDS Data Gathering and Storage: This interaction shows how this component monitors the network traffic collecting and storing security related events.
- IDS Data Distribution: This interaction shows the distribution of IDS events detected as well as IDS data to SMARTIE devices.
- Federation of Systems Service: FoS enables the communication and collaboration between different subsystems.
- IDS Service Discovery and Data Retrieval. This interaction shows the discovery and retrieval operations using the components: IDS data distribution and Federation of Systems.
- Node Attestation: This interaction shows how misuse of the resources by unauthorized parties is detected.

5.1 Registration of devices

This subsection describes the process by which IoT devices are registered into the SMARTIE system what enables them to become members in order to offer their services.

This scenario focuses on the process by which each sensor, once installed and activated, is integrated into SMARTIE. The integration of a sensor is done in two steps. First, the sensor is authenticated using a secure mechanism based on a shared key, certificate or similar. Second, the sensor publishes and offers its resources and services.

Once the sensor is registered into SMARTIE, the information of the sensor is processed. For example, the advertised services are included into the resource discovery component of SMARTIE. Moreover, it updates the authentication and authorization components in order to make the services available to the end users.

Components involved in this interaction:

- Digcovery (database of sensors)
- XACML/JSON (updating policies for the new sensor)
- lwsCoAP

- Configuration Management with Geographic Discovery
- Device registration using secure storage RD
- IDS data distribution
- Federation of Systems
- Distributed Kerberos

5.1.1 Device registration using secure storage RD

This process shows how the device (with associated sensors) would register into the RD using the authentication mechanism based on distributed Kerberos and associated DCapBAC tokens. The IoT device needs to be registered into the RD in order to allow efficient search and discovery of associated services and data. Only devices which are authorised to do so can be registered into the RD using the Distributed Kerberos and DCapBAC capability tokens. Once authorised, the devices can access the RD which then receives the meta-data and stores it into the secure storage. This process is shown in the following sequence diagram.

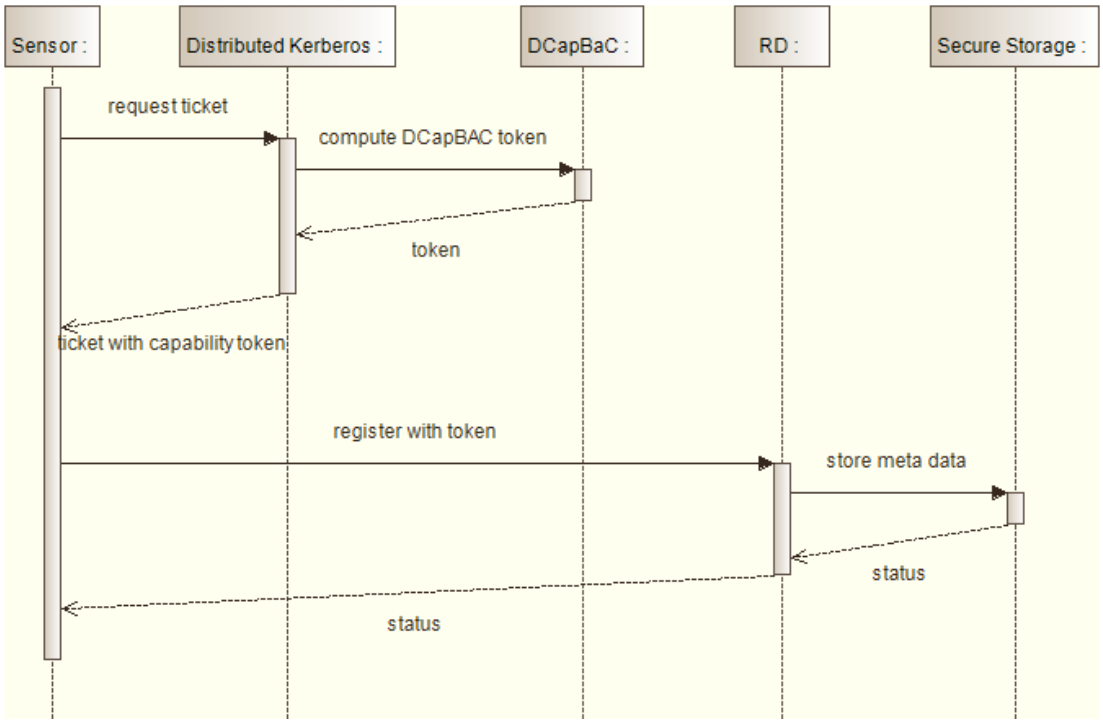

Figure 7: Registration of IoT device into the secure storage RD.

5.1.2 Device registration with Digcovery

The integration of a sensor into the system is done by means of communicating with Digcovery, authenticating itself by any means provided to the sensor (shared key, certificate, etc.) before publishing the features that the sensor offers.

Once the sensor is registered into Digcovery, the access control module (XACML) should create or adapt the existing policies to ensure the access to the sensor is incorporated in to the access control system. Then, Digcovery can push the newly inserted data to into Configuration Management in order to make the information widely available. This process is shown in the next figure.

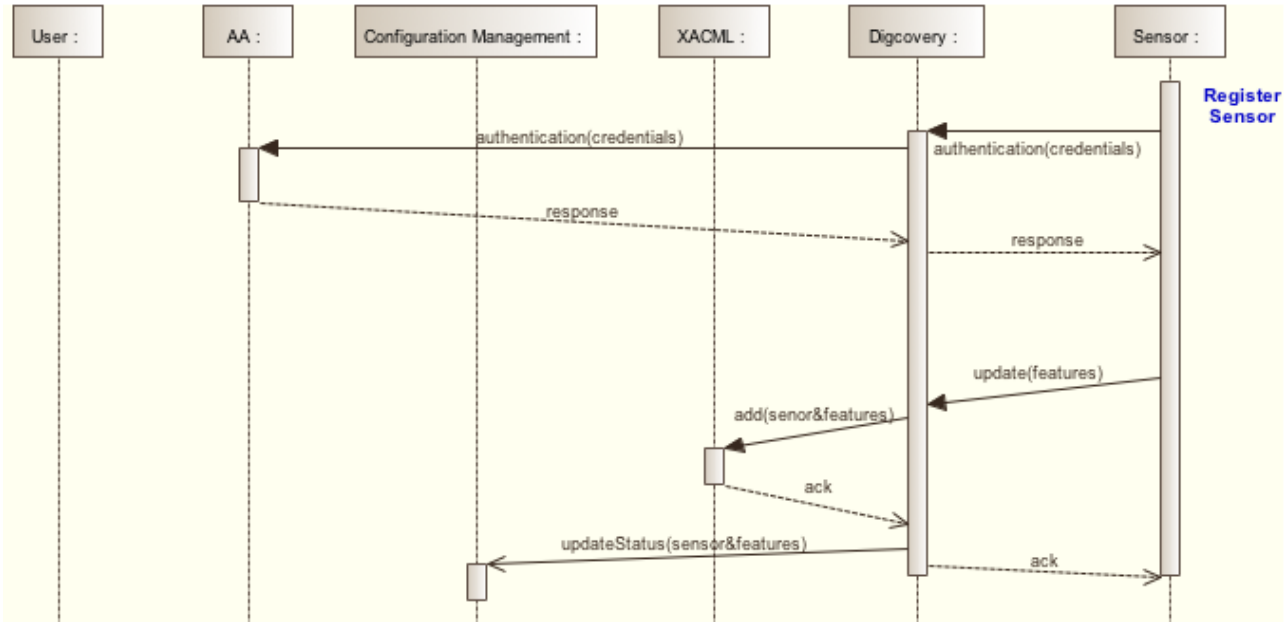

Figure 8: Registration of devices into Digcovery

5.1.3 Registration of Information Services with Configuration Management

In order to be discoverable in the distributed setting and enabling access using the IoT Broker, sensor services as well as information processing services have to register with the Configuration Management. Both sensor services and information processing services have to provide an NGSI-10 Context Interface [5] that can be accessed by the IoT Broker or directly by applications. The respective services may be hosted on sensor nodes, on gateways, on special servers or in the cloud. The registration uses the NGSI-9 *contextRegistration* operation [5]. To enable geographic discovery, the geographic location for which information is provided has to be provided in form of geographic coordinates, as shown in Figure 9. This could be a single point representing the location of a sensor in a simple case, but could also be a geographic area, e.g. in the case of an information processing service aggregating information for a whole area.

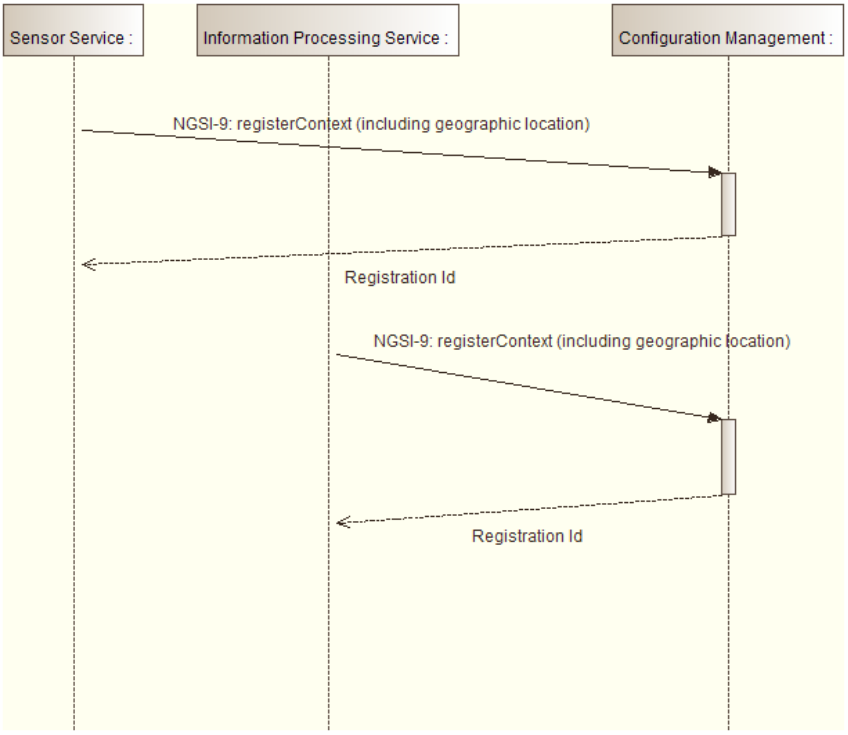

Figure 9: Configuration Management Registration

5.2 Discovery

This subsection describes how users can discover the IoT devices registered into the SMARTIE system in order to establish secure M2M communications.

Components involved in this interaction:

- Digcovery (database of sensors)
- RD (meta data for the IoT devices and services)
- XACML (update policies to include the new sensor)
- Configuration Management with Geographic Discovery

5.2.1 Discovery with Digcovery

As a first step, the user logs into SMARTIE, if not logged in already, prior to any other information exchange. The user then, sends a search request for the discovery, getting a list of the targets matching the search description. Then the user will receive the information needed to access the sensor, the actual target of the search, that being one or more, and complementary information such as the communication models available (publish/subscribe, request/response, etc.) that can be established between the sensor and the user, and other services that the sensor offers that might be relevant to the search that was submitted.

As an example a user requests all temperature sensors in the first floor of a building. Assuming the user is already logged into the system, he submits the request to Digcovery, which resolves the requested information and returns a list with the sensors that fit the search parameters, along with information for communicating directly with them if possible, and the modes of communication.

For the purpose of this example, Digcovery returns the value of a temperature sensor in a server room in the first floor, along with the IP address and port of the sensor. Digcovery may return additional information regarding the sensor like the possibility of communicating directly with the sensor, and subscribing to receive updates every minute. This process is shown in the next sequence diagram.

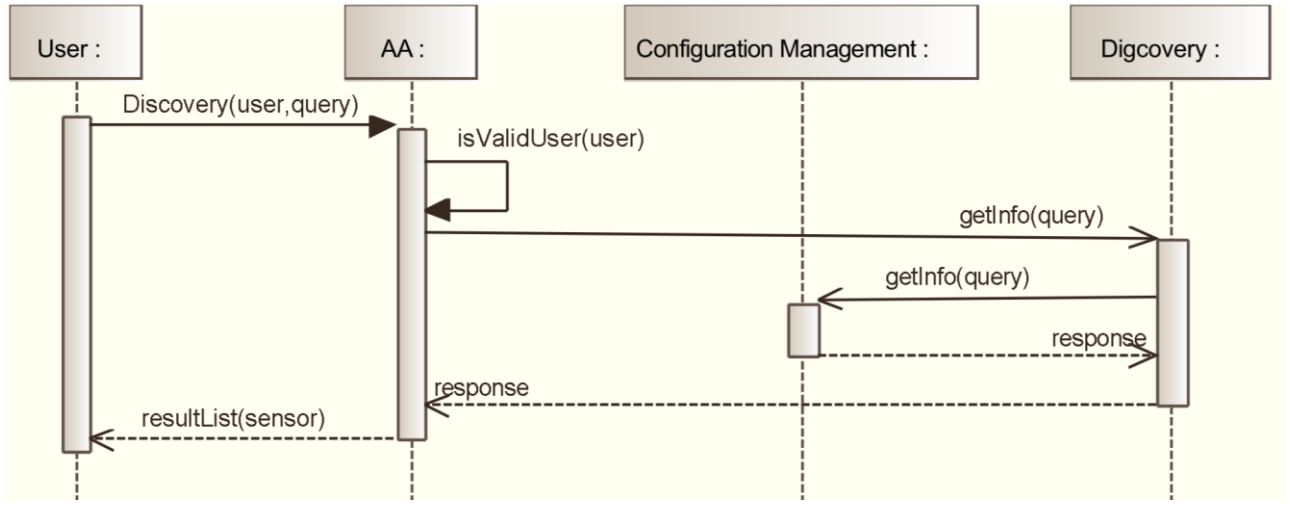

Figure 10: Discovery search with Digcovery

5.2.2 Discovery with Configuration Management

The Configuration Management enables users to discover the information services that provide the requested information. A geographic scope using geographic coordinates can be used to limit the provided services to those that are within the geographic area of interest, e.g. not all services providing outdoor temperature, but those in a certain area of the city. This flow is shown in figure 11.

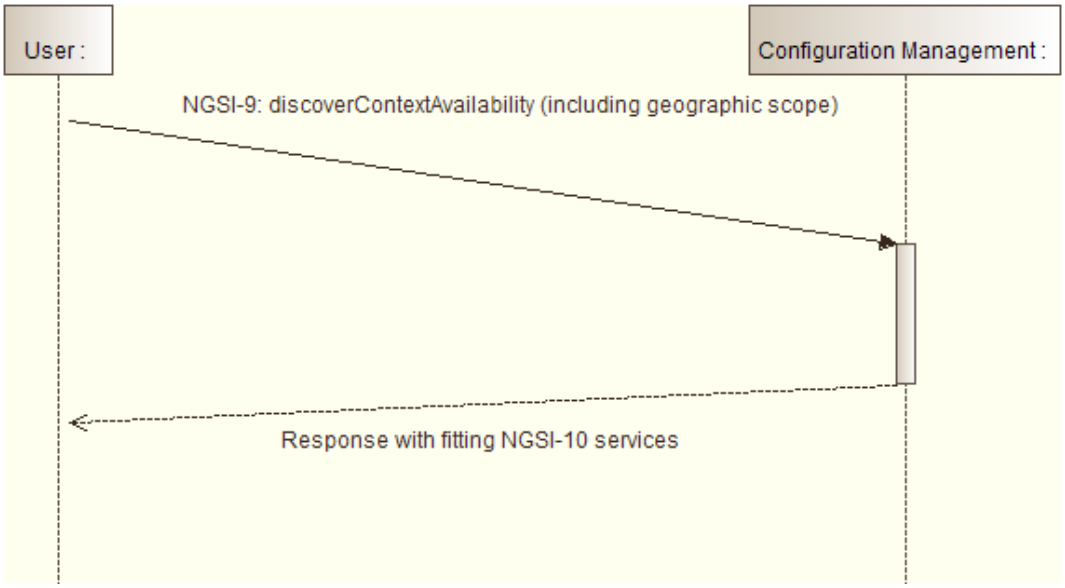

Figure 11: Discovery using the Configuration Management

5.2.3 Discovery with RD

Discovery using the RD is performed in a similar manner as in the previous cases. The user contacts the RD with the specific query indicating the search and discovery criteria such as location, sensor capability, type of sensor, sensitivity etc. The RD then checks the validity of the user by contacting the Authentication and Authorisation Authority. Providing that the user is valid and allowed to access the requested information, the RD performs the internal search procedure and returns this information to the user, as shown in Figure 12.

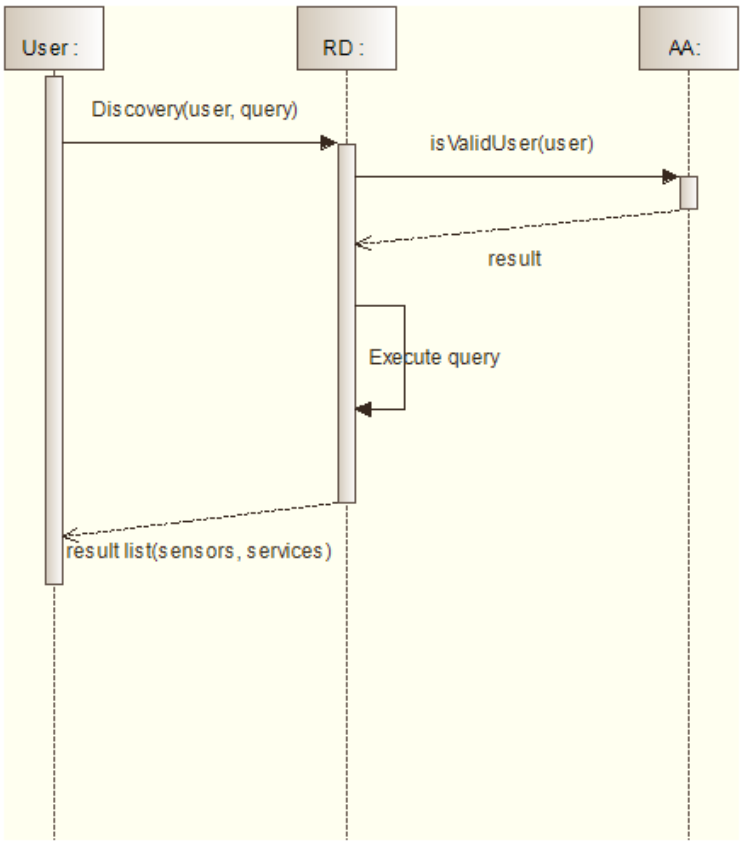

Figure 12: Discovery using the RD

5.3 Retrieve

This subsection describes how users can retrieve information from IoT devices of the SMARTIE system using authorization mechanisms based on secure tokens and policies.

Components involved in this interaction:

- XACML/JSON (Resolve the A.C. decision)
- DCapBAC (Capability Token issue and parsing from the sensor)
- Distributed Kerberos
- Processing Flow Optimization
- IoT Broker
- Configuration Management with Geographic Discovery

5.3.1 Retrieve with Kerberos-based tokens

For lightweight sensors or additional security against server compromise, the distributed Kerberos component can be integrated and used to generate the capability tokens. This goes along with two changes

- The cryptographic primitive will be changed and similar to the Kerberos protocol be based on a symmetric block cipher. This brings a reduced computational load to the sensor or actuator, but comes along with a higher complexity in key management compared to public key cryptography such as ECC.
- The distributed server follows the design decision that there should be no single entity able to create capability tokens. To successfully create a token, at least two instances need to cooperate. This protects against server compromise and a high security can be reached, if the two servers are distinct in hardware, operating system and software. The distributed server can also be used with the ECC-based capability tokens. However we will describe the Kerberos-based solution in this section.

The scenario considered is again that a user intends to contact a device directly, e.g. to retrieve information from a sensor or cause actuation at an actuator. The user knows the deviceID of the device to be contacted, and goes to the platform to obtain a Capability token to get authorization and send it along with the request to the device. The user sends the token request to one of the distributed Kerberos servers that will then act as a proxy. The changed interaction can be seen in the backend as shown in Figure 13.

KerberosServer 1 will evaluate the request and compute a DCapBAC token. It generates then a garbled circuit to authenticate the token and sends the circuit to KerberosServer 2. The server will also evaluate the request to compute the capability token, too. If successful, KerberosServer 2 contacts KerberosServer 1 for the required secret input parameters (that will be transmitted by oblivious transfer). Then the garbled circuit will be evaluated and the result (the authenticated capability token) is sent back to the first server who acts as a proxy to the user.

A detailed view of the communication between the two servers when running a full Kerberos protocol is given in Figure 14. Each user and each device hold a symmetric key. The second copy of this key is cryptographically shared between the two Kerberos servers. Therefore, the two servers need to cooperate to perform cryptographic operations with this key, such as signing a token. In the case of computing a Kerberos token, the garbled circuit is computing four cryptographic operations. They are related to the authentication of the user and can be replaced easily be other means of user authentication. The fourth computation “Create Token” is responsible for authenticating the token, e.g. a capability token as described previously.

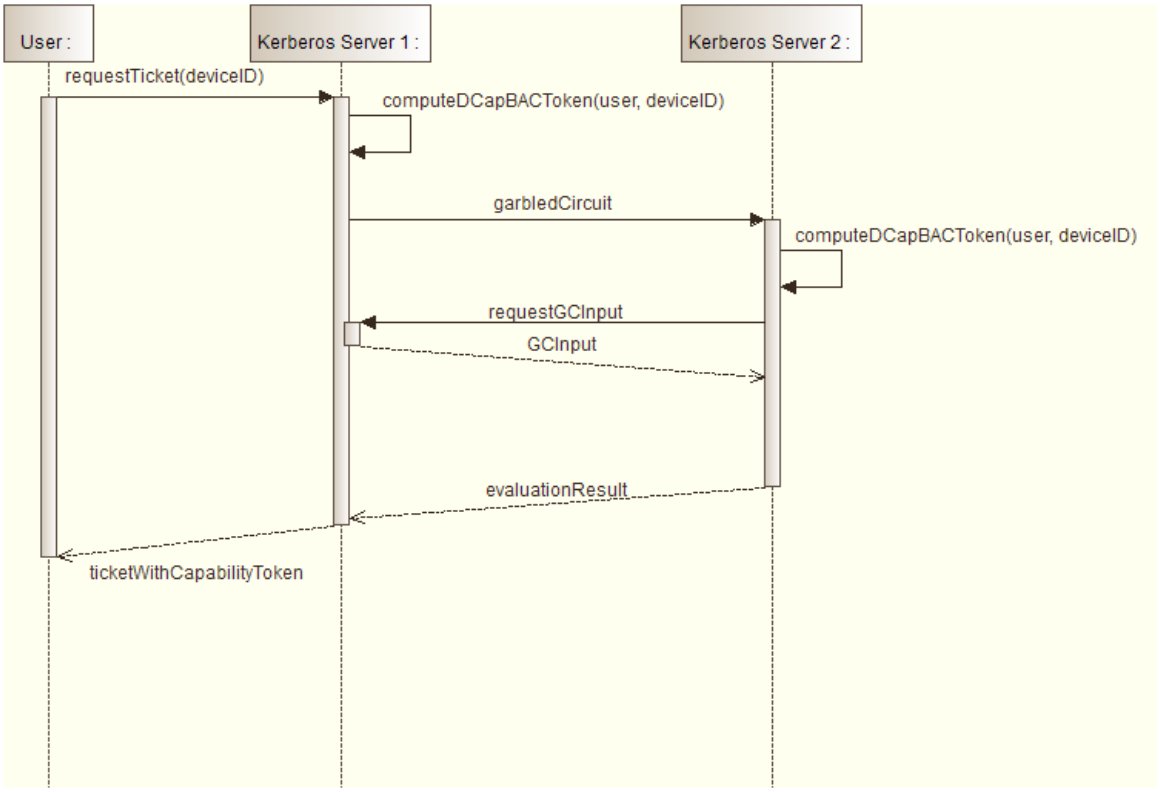

Figure 13: Kerberos-based DCapBAC token

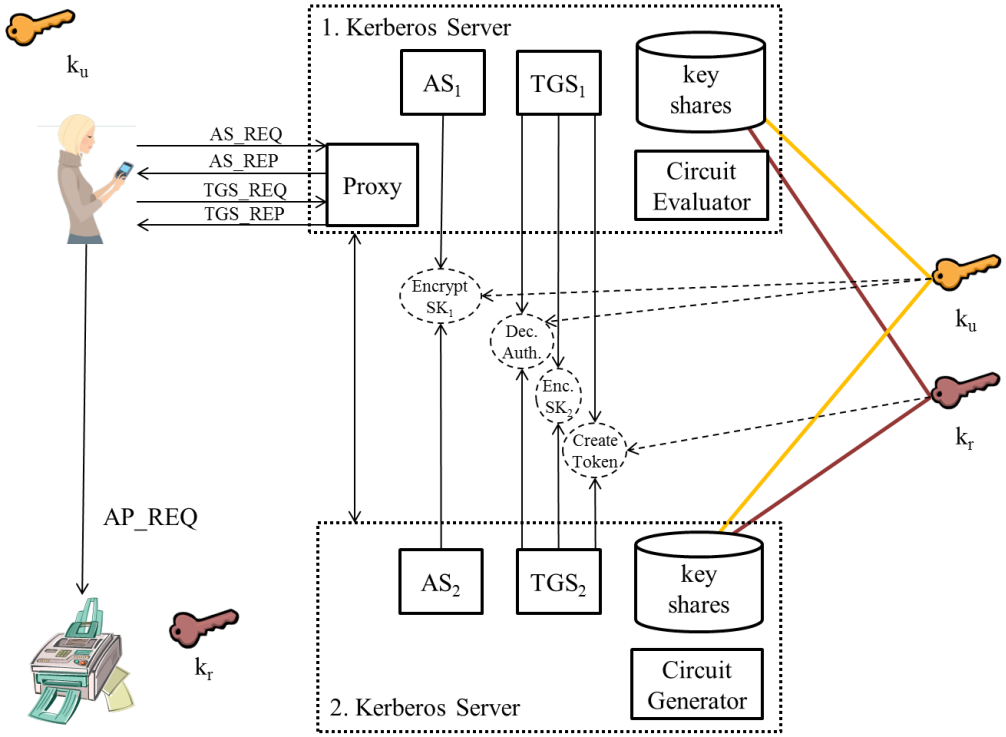

Figure 14: Detail view of the Kerberos servers and involved keys.

Once the user has the ticket embedded in the capability token, it interacts directly with the device in the same way as previously described. This interaction can be fused with the DCapBAC-XACML retrieve using Distributed Kerberos as Authentication and Authorization Entity (AA) or just Authentication Entity (AuthN).

5.3.2 Retrieve with IoT Broker

The IoT Broker acts as a single access point for a user to retrieve information from the system. However, there can be any number of IoT Broker instances running in the system, so it is not a single point of failure or a bottleneck with respect to overall scalability.

As shown in Figure 15 users can retrieve information synchronously using the NGSI-10 queryContext operation [5]. The operation has to specify what information the user is interested in, possibly based on the type of information, and a geographic scope in geographic coordinates can be provided.

The IoT Broker then uses the NGSI-9 queryContextAvailability operations [5] to query the ConfigurationManagement component for suitable information services, which can either directly, provide sensor information, or already processed information. The IoT Broker then uses the NGSI-10 queryContext operation to query the information from the identified information services. The IoT Broker then aggregates the information and returns it to the requesting user.

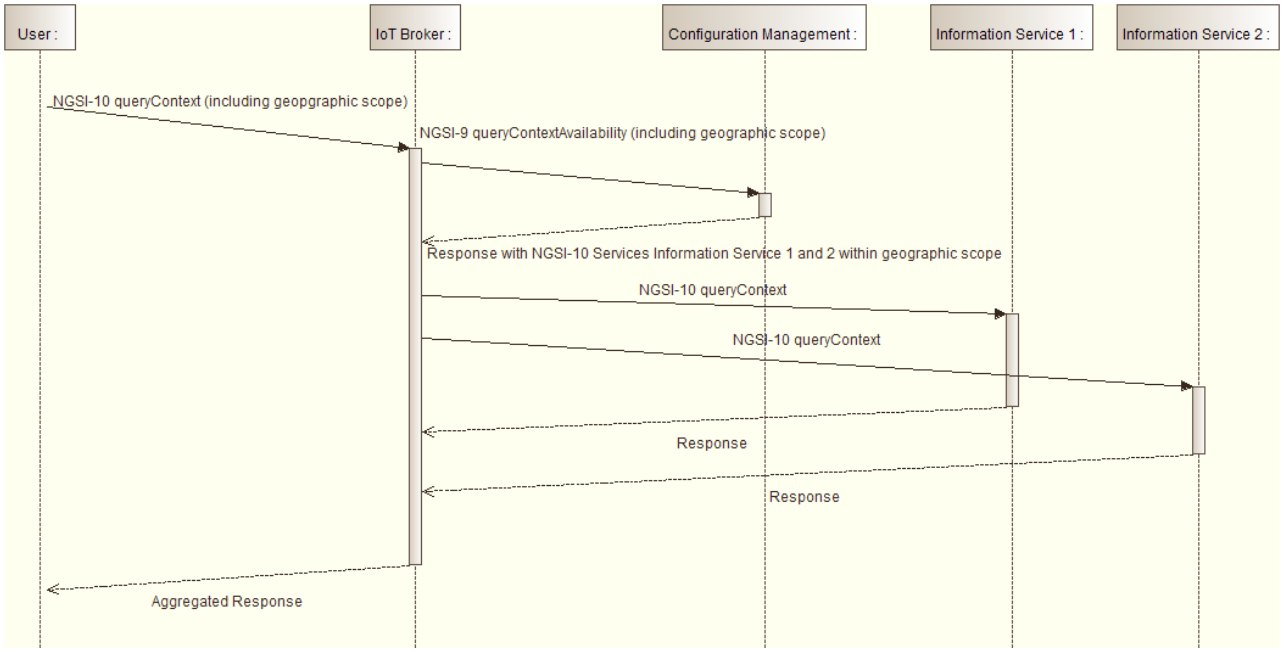

Figure 15: Retrieving information with the IoT Broker

5.3.3 Retrieve with IoT Broker and Processing Flow Optimization

Using IoT Broker, the interaction shown in Figure 16 starts in the same way as depicted in Figure 15, only that here, initially, no suitable information services can be found. In that case the IoT Broker contacts the Processing Flow Optimization to see if the requested result can be obtained by processing information from existing sources. If this is the case, suitable existing sources are selected and an information processor is deployed on the most suitable compute node, which could be in the cloud, on a dedicated server, on a gateway or even on a sensor node. In case, the setup is successful, the new information processor is returned as a suitable source to the IoT Broker. The IoT Broker then uses the NGSI-10 queryContext operation [5] to request the information, which is then returned to the user.

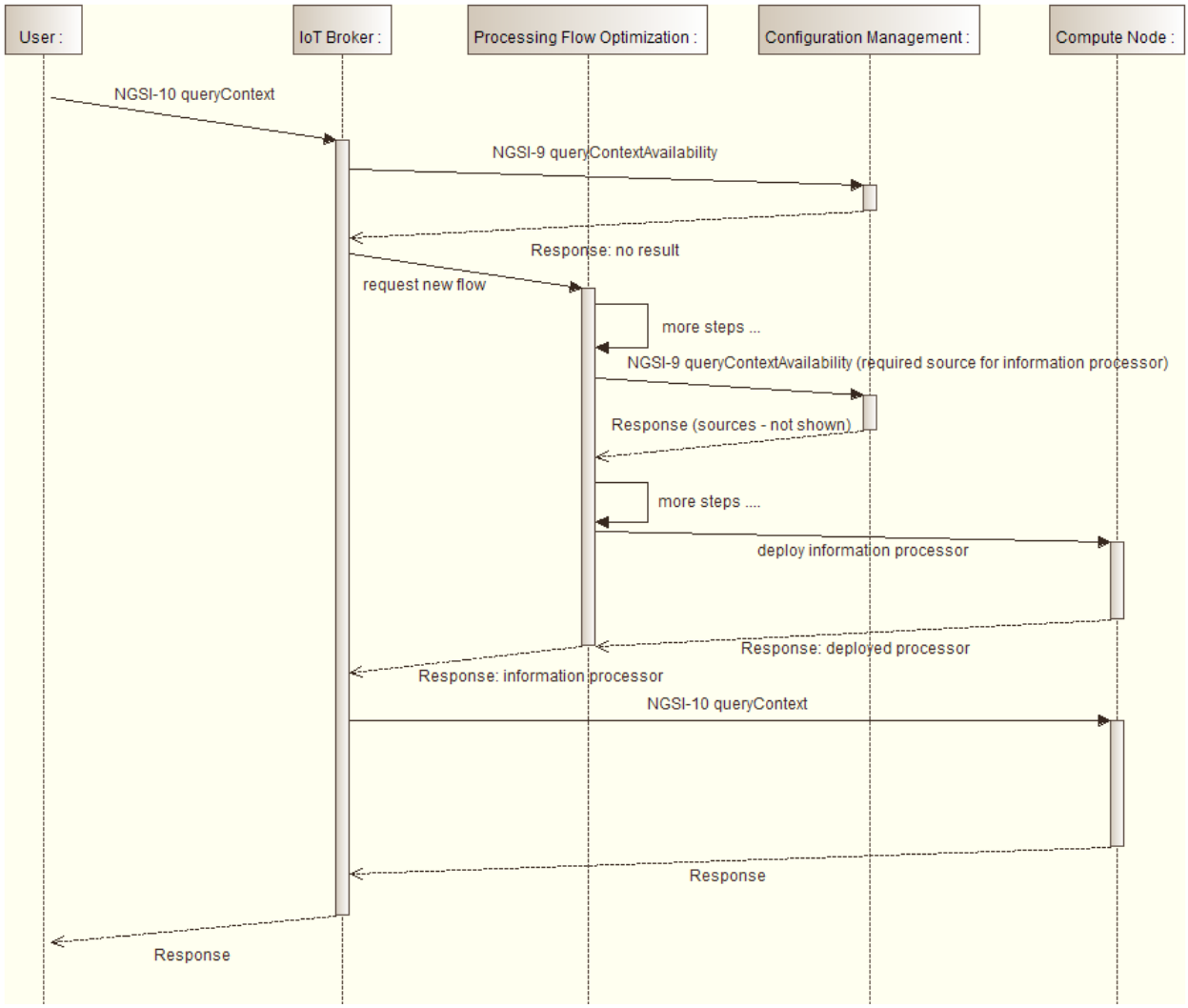

Figure 16: Retrieving information with the IoT Broker using the Processing Flow Optimization

5.3.4 Retrieve with DCapBAC

Here, the user intends to retrieve information from a sensor directly. At this point the user knows the sensor that is the source of information that he is looking for, so he tries to obtain a capability token to get authorization and send it along with the request to the sensor.

The user sends a query and the Distributed Kerberos manages to authenticate, and ask for authorization for the user, and then checks if the user is valid with the AuthN module. If successful, the next step is to confirm the user has the proper authorization to perform the actions requested. DCapBAC then asks XACML for the user's permissions, before issuing the capability token. This process is shown in the sequence diagram in figure 17.

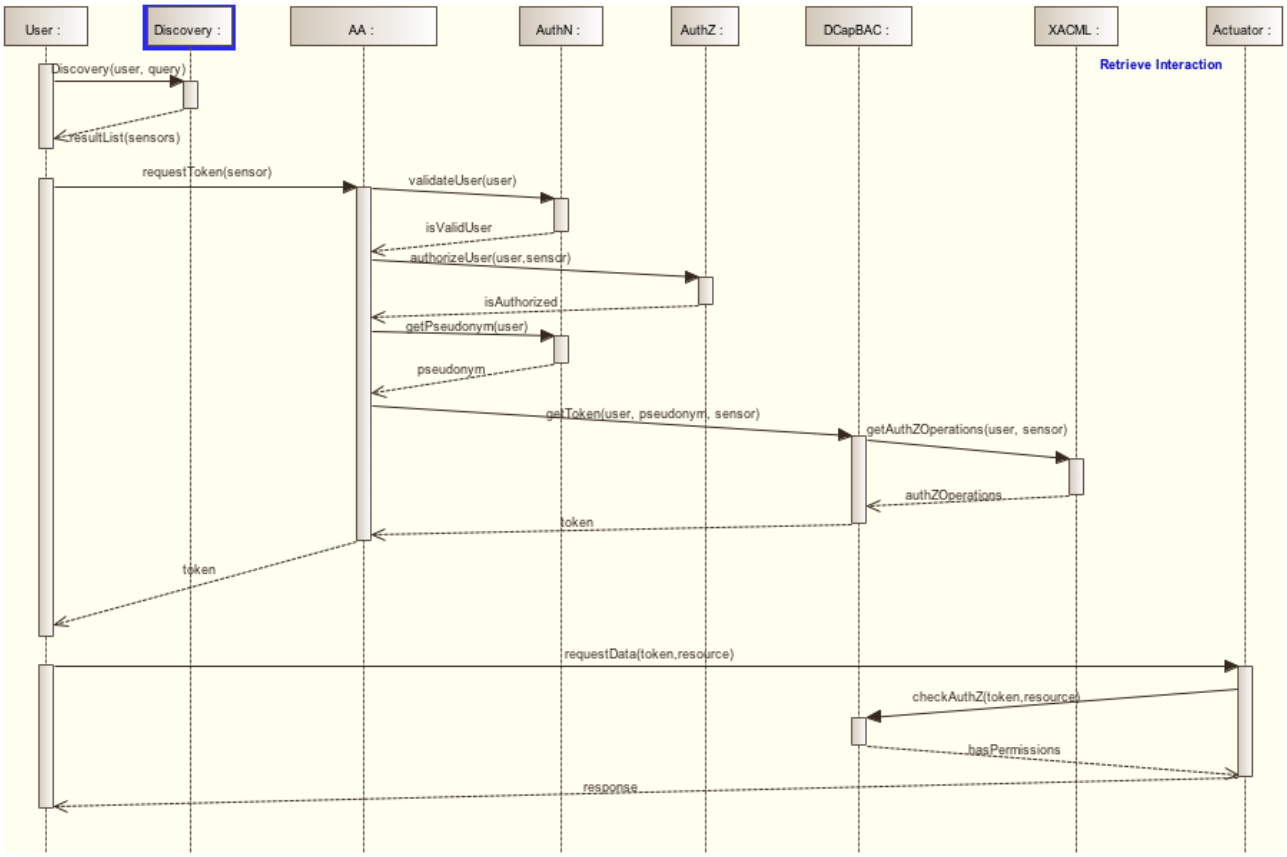

Figure 17: Retrieve Interaction with Capability Token

Once the user has the proper authorization embedded in the capability token it can interact directly with the sensor. In this sequence, the user sends a request along with the capability token obtained in the previous interaction. The sensor parses the request, checks the token, and if the verification is successful, the sensor, returns the information the user was requesting.

IoT Broker can act as a sensor in this interaction, acting as a source of information for the user, as shown in previous interactions.

In this interaction, there is a possibility of substituting the lifelines from Authentication and Authorization Entity (AA) to Authorization Entity (AuthZ) by Distributed Kerberos, leveraging its functionality and integrating both components into the retrieve interaction.

5.4 Subscribe

This subsection describes how users can subscribe to the SMARTIE system to obtain periodically information from IoT devices. This process requires authorization mechanisms based on secure tokens and policies, similar to the retrieve interaction.

Components involved in this interaction:

- XACML/JSON (Resolve the A.C. decision)
- CP-ABE (Secure and efficient distribution of information)
- Processing Flow Optimization
- IoT Broker
- Privacy-Preserving Geofencing

5.4.1 Subscribe with IoT Broker

As shown in Figure 18, users can subscribe to information through the IoT Broker using the NGSI-10 subscribeContext operation [5] to asynchronously receive notifications containing the requested information. The IoT Broker uses the NGSI-9 subscribeContextAvailability operation [5] to subscribe for suitable information sources (which may change over time, which would then require adaptations). The user gets back a subscription Id, so that incoming notifications can be attributed to requests. The IoT Broker receives notifications via the NGSI-9 notifyContextAvailability [5] with fitting information sources. The IoT Broker then subscribes to the information sources using the NGSI-10 subscribeContext operation [5]. On receiving notifications via the NGSI-10 notifyContext operation [5], the IoT Broker forwards them to the user.

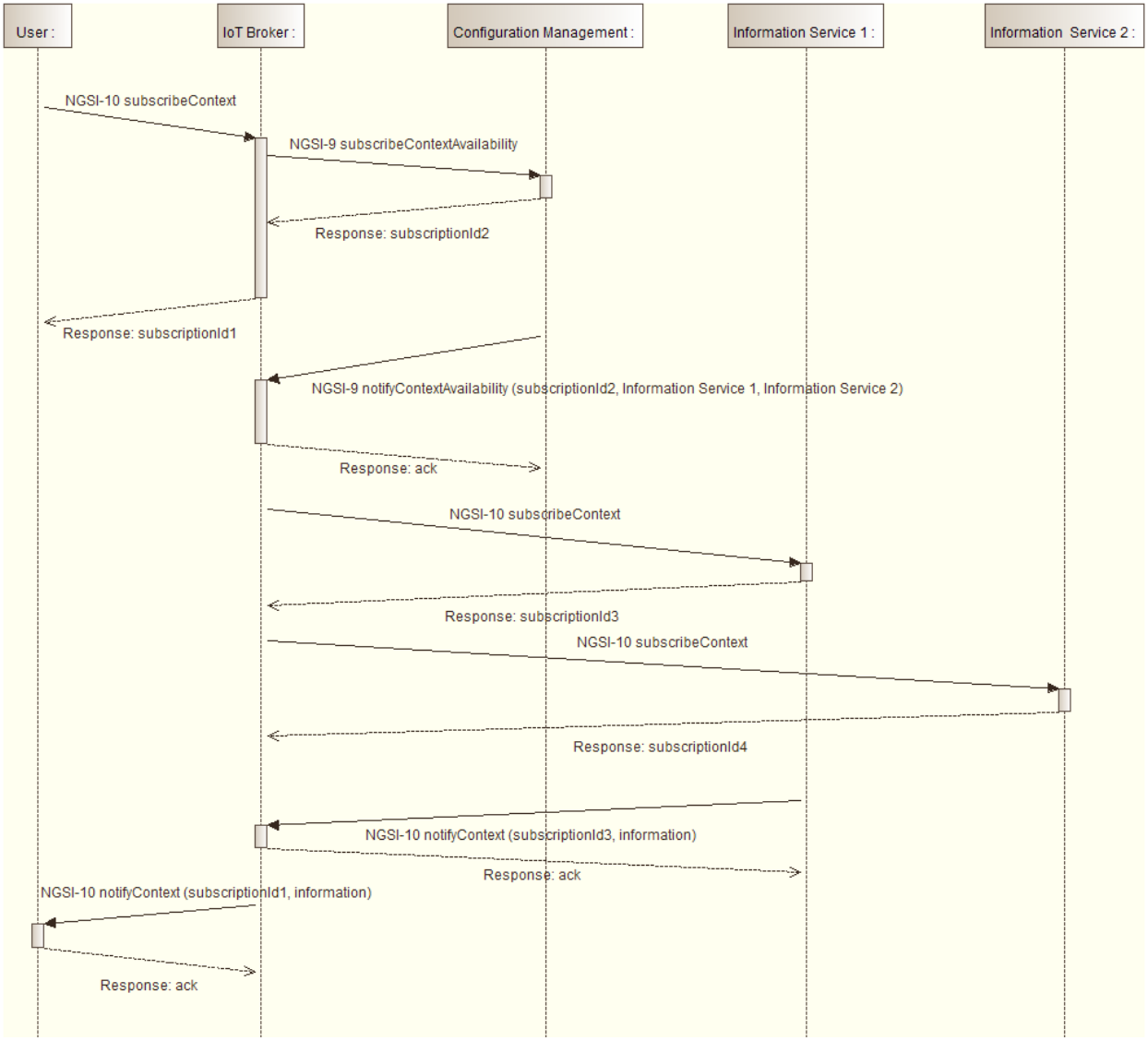

Figure 18: Subscribing to information with the IoT Broker

5.4.2 Subscribe with IoT Broker and Processing Flow Optimization

As shown in Figure 19, this interaction starts the same way as described in the previous section 5.4.1, only that here, initially, no suitable information services can be found. In that case the IoT Broker contacts the Processing Flow Optimization to see if the requested result can be obtained by processing information from existing sources. If this is the case, suitable existing sources are selected and an information processor is deployed on the most suitable compute node, which could be in the cloud, on a dedicated server, on a gateway or even on a sensor node. In case, the setup is successful, the new information processor is returned as a suitable source to the IoT Broker. The IoT Broker then uses the NGSI-10 subscribeContext operation [5] to subscribe for the requested information. When the IoT Broker receives a notification with the NGSI-10 notifyContext operation [5], it forwards the notification the user.

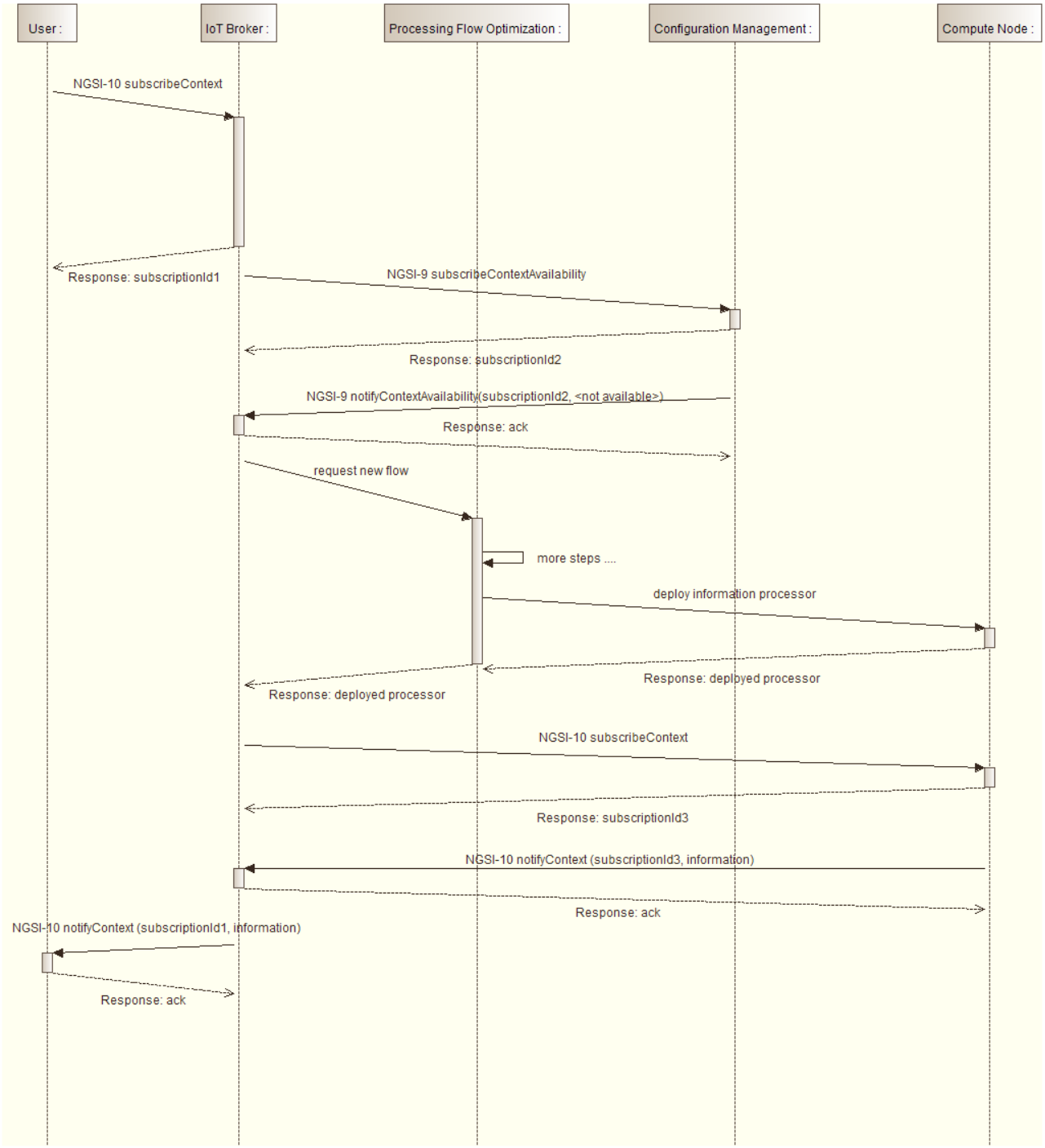

Figure 19: Subscribing to information with the IoT Broker

5.4.3 Subscribe with CP-ABE

This interaction explains how a user subscribes to a topic. A sensor is a source of information for that topic, and when new information is generated, the sensor sends that information to the platform, so the platform can distribute that information to the users that subscribed in an efficient and secure way by using CP-ABE.

In the first part of this process is the user requesting the information broker to subscribe to a given topic. To accomplish this, the information broker has to submit the information provided by the user to the authorization manager that will check with XACML if the user has the permission to subscribe. If this was

successfully completed the user will receive the response, that he is subscribed, and when new information is produced, he will receive that information accordingly. This interaction is shown in figure 20.

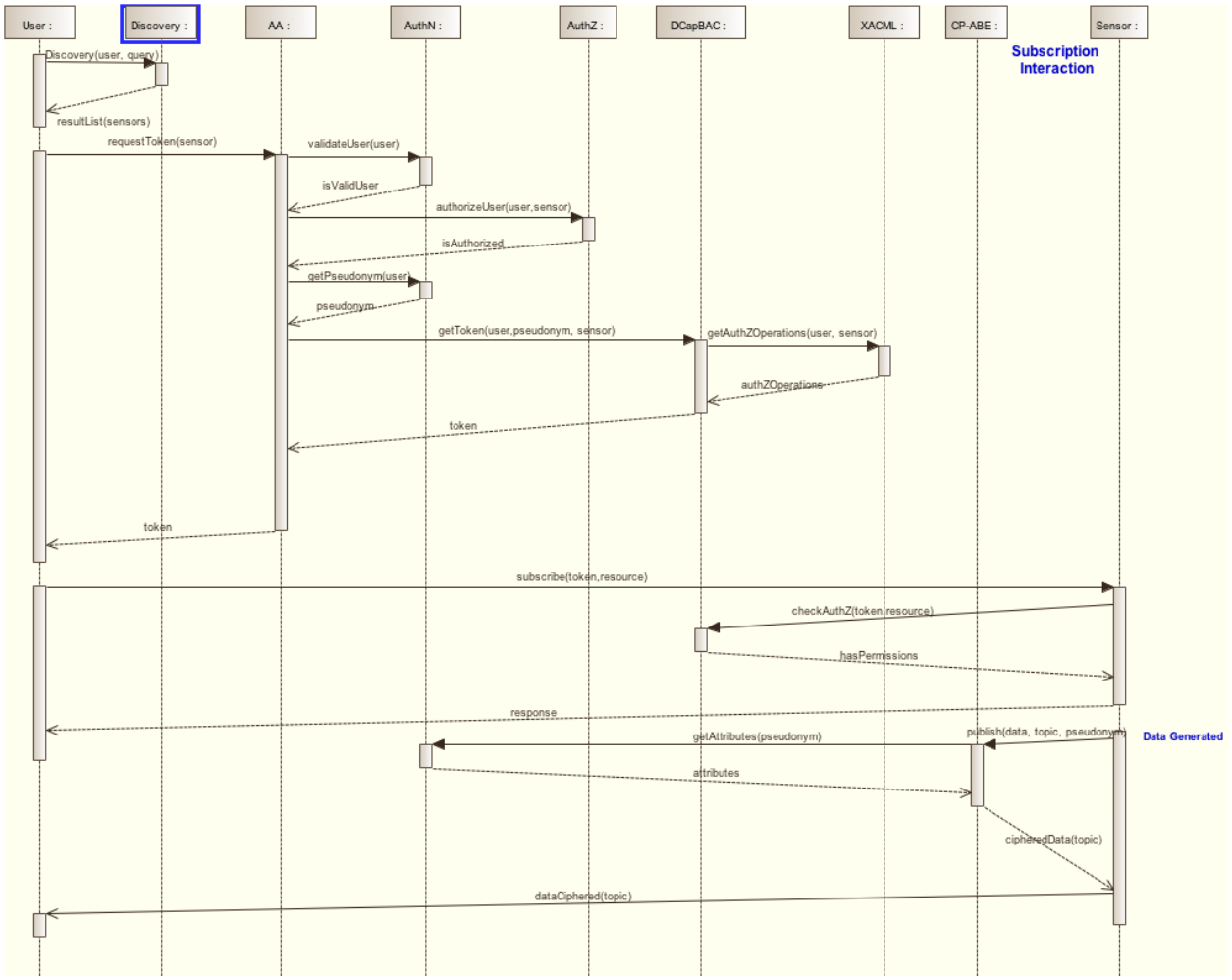

Figure 20: Subscription with CP-ABE

When new information is produced, the sensor sends that information to the information broker, and the information broker then processes that information to send it to all users subscribed to it. The processing of the information passes through checking the groups of users that subscribed to that topic, and then ciphering with CP-ABE the information, according to the policies that match the minimum number of groups that will be able to decipher the information.

The IoT Broker can act as a sensor in this interaction, as shown in previous interactions, acting as a source of information for the user. In this interaction, there is a possibility of substituting the lifelines from AA to AuthZ, by Distributed Kerberos, leveraging its functionality and integrating both components into the retrieve interaction.

5.4.4 Privacy-preserving subscription service

This process shows two components that provide a subscription service which will process encrypted data by means of PrivLoc. The authentication can be done as previously described based on capability tokens.

PrivLoc provides a proxy that scrambles or encrypts location information in order to protect the privacy for users or enterprises that rely on location-based services. In the following we describe the use of a geofencing service. The service gives users the possibility to subscribe to areas. Moving devices will regularly report their location to the service, and once a device enters or leaves a subscribed area, the respective subscriber will be notified.

A standard location-based geofencing service using a spatial database is assumed to be given. Three instances of this service are started. However, the plain use of this service would reveal the interest of subscribers and movement paths of devices to the service operator. Therefore the PrivLoc proxy is used in

the interaction between users, devices and service provider. Figure 21 shows a user subscribing to the service. The subscription is encrypted by the proxy and stored in all servers. Figure 22 shows a device reporting location updates. The reports need only to be processed by one server, thus having basically the same performance as the unencrypted evaluation of the report. The proxy will choose a suitable server out of the 3 instances and submit the encrypted report. If the report triggers an event (entering or leaving an area), or if an event is detected, the response is sent back to the user, forwarded by the proxy.

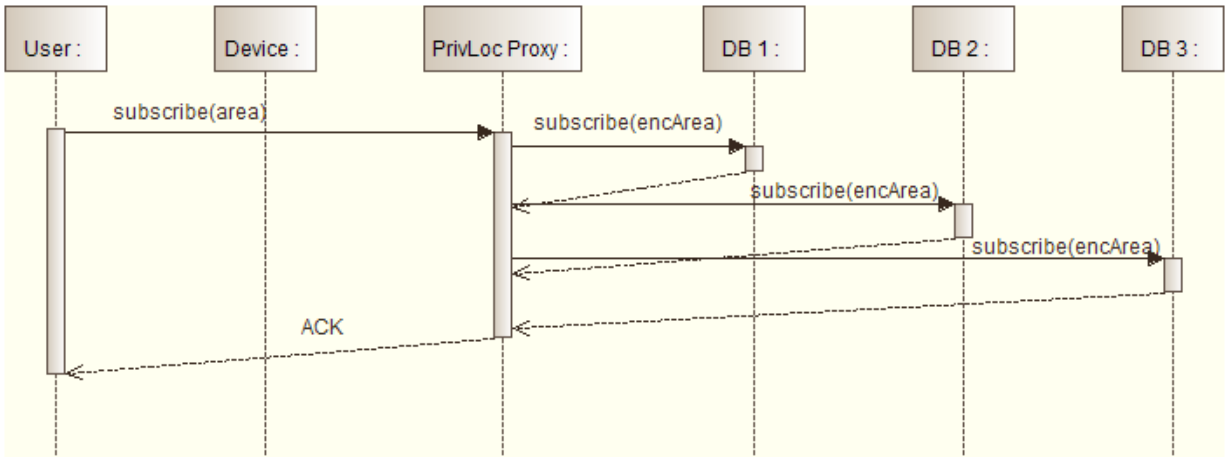

Figure 21: User subscribing to PrivLoc.

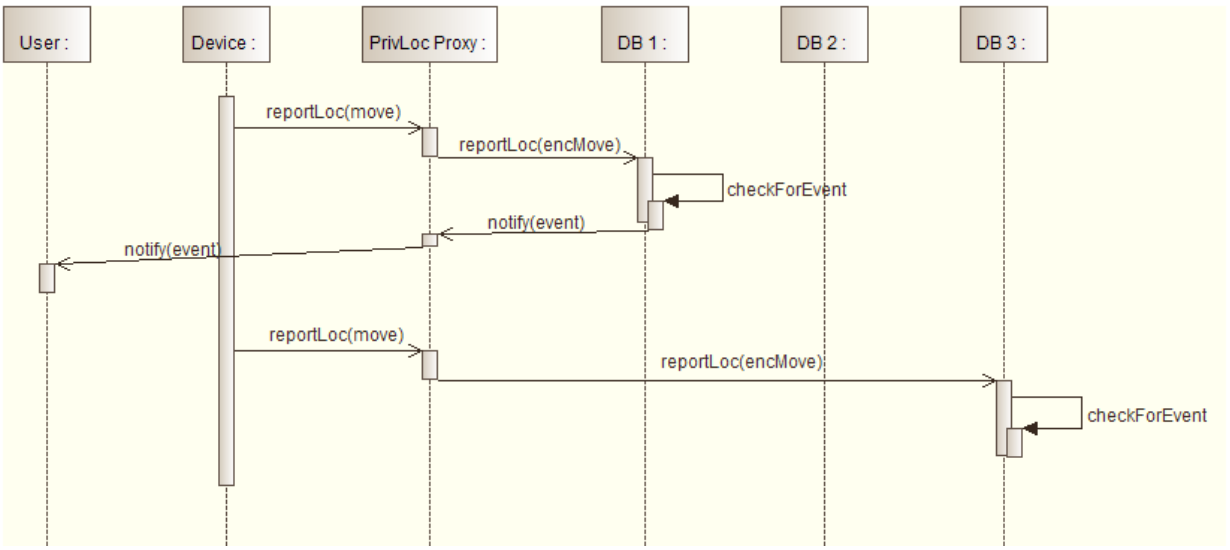

Figure 22: A device reporting its movements to PrivLoc.

5.4.5 Data upload from the sensor device

This process shows how the device (with associated sensors) will upload the data to the server side secure storage component utilising the secure data transfer protocol. The secure storage component is implemented as part of the Resource Directory module, but in this context it is used independently.

The IoT device is responsible for acquiring the information from the associated sensors which is then transferred to the server side secure storage using the secure CoAP based on the shortECC encryption scheme. The lwsCoAP component is located at both device and server side enabling the secure channel to be established between the two entities, as shown in figure 23.

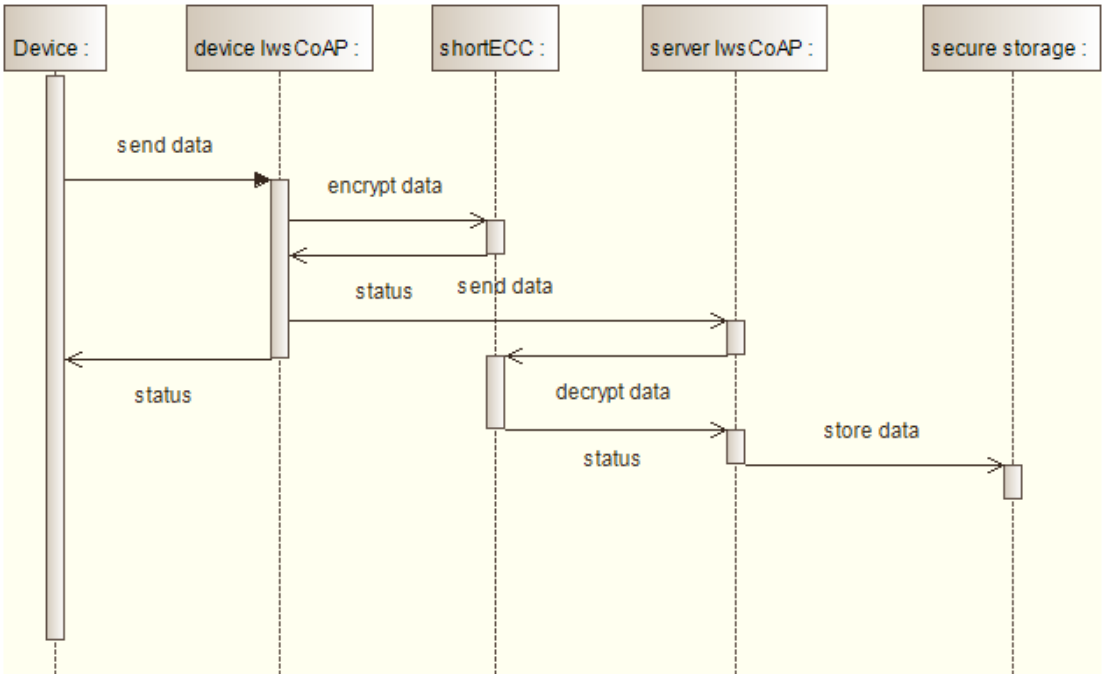

Figure 23: Data transfer from the IoT device to secure storage server side using the lwsCoAP based on shortECC encryption.

5.5 Actuation

This subsection describes how users can request actuations of IoT devices in the SMARTIE system. This process requires authorization mechanisms based on secure tokens and policies.

Components involved in this interaction:

- XACML/JSON (Resolve the Access Control decision)
- DCapBAC (Capability Token issue and parsing from the sensor)

5.5.1 Actuation with DCapBAC

This process is similar to the retrieve information phase, changing the values in the capability token. First, the user obtains the capability token of the AA entity in order to interoperate directly with the actuator. Then the user sends the action query to the actuator with the capability token, as shown in figure 24.

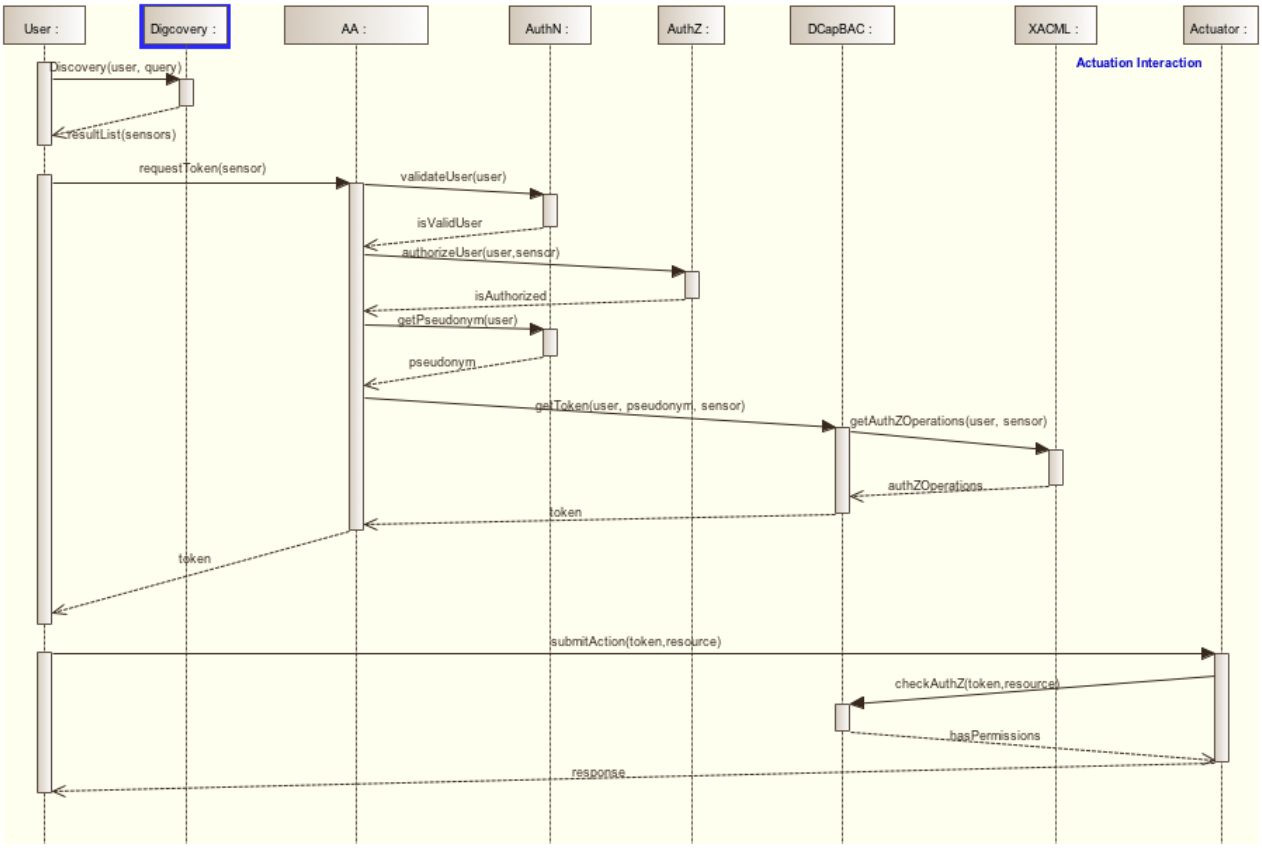

Figure 24: Request with Capability Token to Actuator

5.6 Event Detection

This process shows components that provide a secure distributed shared memory service with event detection. The authentication can be done as previously described based on capability tokens. Figure 25 presents the tinyDSM data sharing and storage and also event detection. It also shows tinyDSM interacting with the security library. Figure 26 shows the interaction of the tinyDSM and the entity requesting the data from the shared memory.

Components involved in this interaction:

- tinyDSM
- shortECC&lmRNG

5.6.1 Secure distributed shared memory and event detection service

The tinyDSM middleware provides distributed shared data memory for sensor nodes. The middleware on Device1 gets the sensor measurements (dataS1 in Figure 25) from the sensor on this device. As a next step the event detection is performed (detectEvent in Figure 25). It is checked if the predefined unusual situation related to dataS1 measurement occurs (what can change the behaviour of the network or part of it, e.g., causing changing the sampling rate or sending the latest measurements to a predefined sink). If it occurs, the authorized User gets the notification (notifyEvent in Figure 25). Additionally the tinyDSM data can be secured (encrypt in Figure 25) in order to prevent unauthorized access to the data. Then the encrypted data is stored in the shared memory area (store in Figure 25). The encrypted data can be shared with other devices (share in Figure 25) in order to provide data redundancy. In case the device with the sensor generating the measurements will be broken, it is always possible to get the latest measurements from the sensor nodes holding the copies of this data. The devices holding the copies of the data can also perform the event detection and send the notification to the authorized entities.

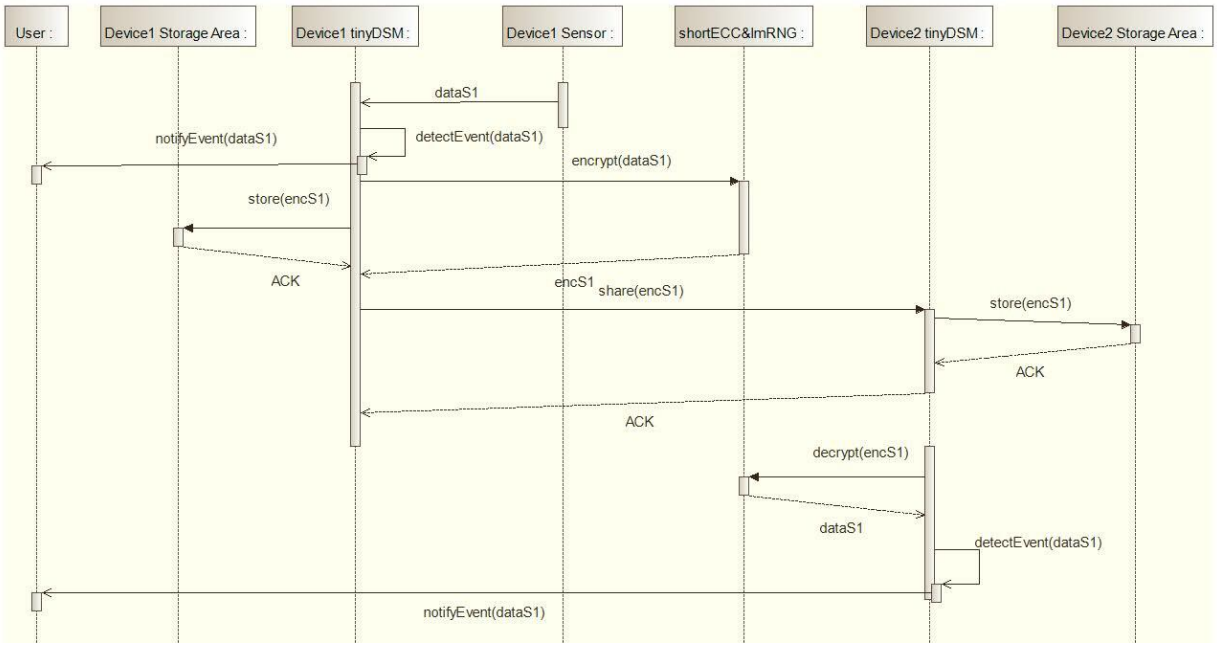

Figure 25:tinyDSM interacting with sensor and shortECC

The authorized user can request the data from the middleware (request in Figure 26). The middleware gets the data from the storage area (get in Figure 26), encrypts it if necessary and sends it back to the user.

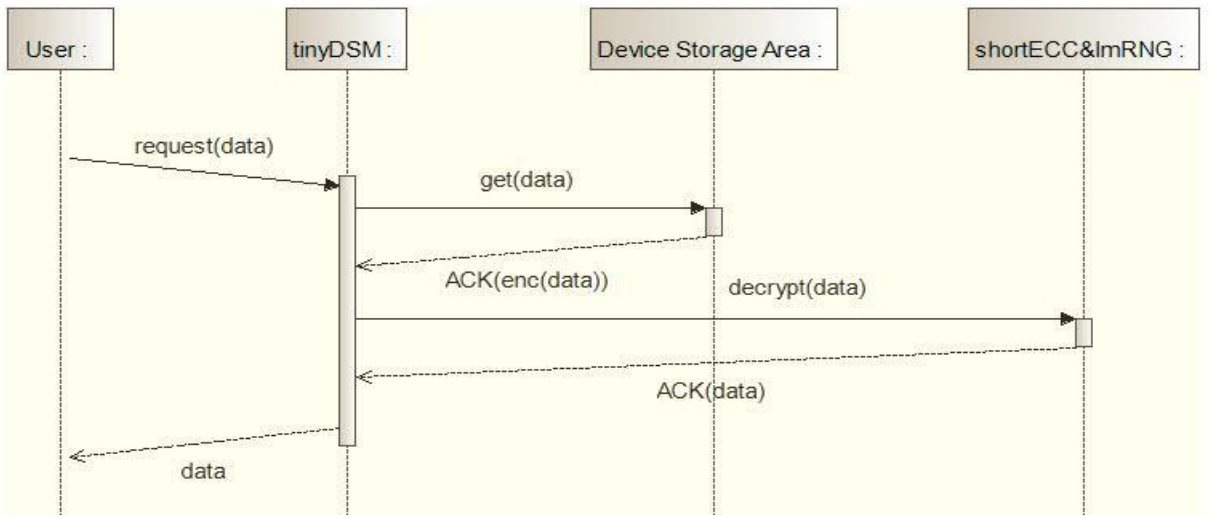

Figure 26: User requesting the data from tinyDSM

5.7 IDS Data Gathering and Storage

The IDS operation involves the two components IDS Data Gathering and Storage, regarded in this section, and IDS Data Distribution, regarded in Section 5.8. The IDS Data Gathering and Storage component monitors the network traffic and collects and stores security related events, whereas the IDS Data Distribution component distributes detected IDS events or IDS data to SMARTIE devices.

This section describes IDS interactions that are accomplished by the IDS component IDS Data Gathering and Storage that is responsible for security monitoring, which means data and event collection from different

sources and the storage of these data and events. The data and event collection is regarded in subsection 5.7.1 and the data and event storage in subsection 5.7.2.

Components involved in this interaction:

- IDS Data Generation and Storage
- lwsCoAP
- tinyDSM
- shortECC&lmRNG (ECC&RNG)

5.7.1 IDS Data Generation

This process describes the IDS data generation. The IDS Data Gathering and Storage component monitors the network traffic to detect security related or unknown events. The IDS collects, in addition, events from each of the following components lwsCoAP, Node Attestation, tinyDSM, and Privacy Preserving event detection and correlation. These events are then correlated by the IDS Data Gathering and Storage component, and, if necessary, an IDS event is generated. This process is shown in figure 27.

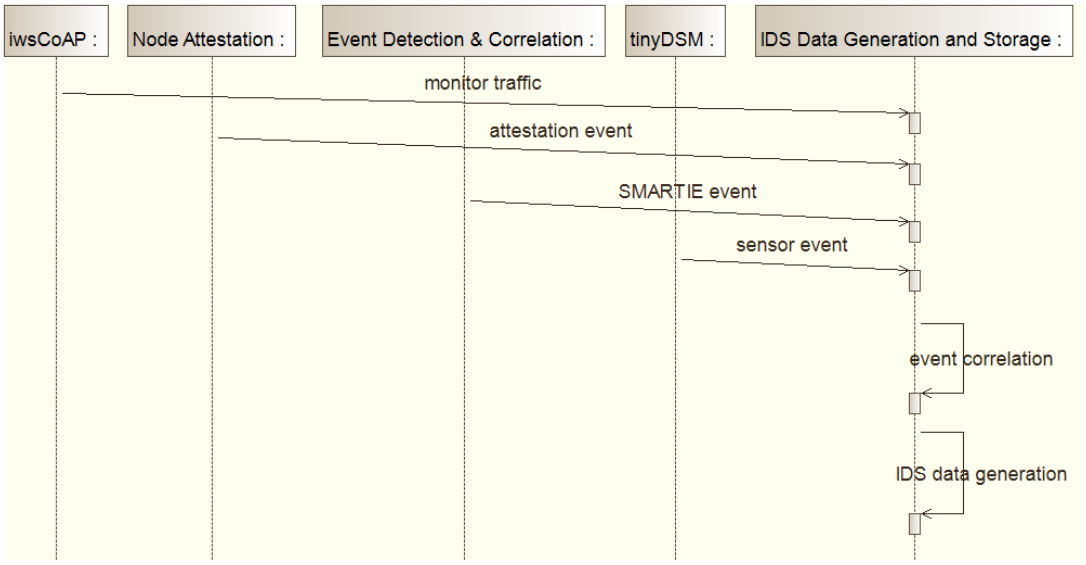

Figure 27: IDS data generation involving events created by other components

Data correlation

An example for event detection and correlation is a set intersection protocol. In this setting, two devices, e.g. components of the IDS, are submitting encrypted values to the correlation service. The encrypted values could for instance represent alerts observed by the device. If both devices submit the same alert, this constitutes a situation about which a user would like to be informed. The user subscribes therefore at the correlation service for the function f, in this case set intersection, applied to the data received by device 1 and device 2. The correlation processes all incoming encrypted data of those devices to find out if the function f applies. If this is detected, a notification about the event is sent to the user. Figure 28 shows the information flow of the subscription service offered by the correlation service.

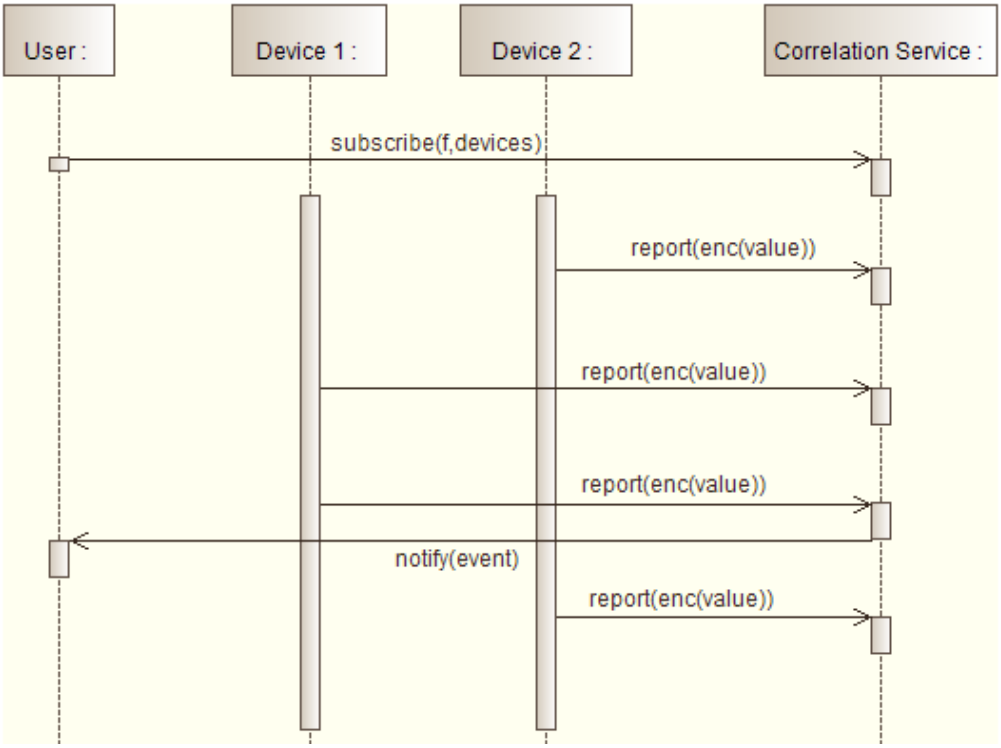

Figure 28: Event detection and correlation service with the example of private set intersection.

5.7.2 IDS Data Storage

This process describes the IDS data storage. Therefore, the IDS Data Gathering and Storage component either calls shortECC and lmRNG components for encryption / decryption of data before storing it locally or it uses the tinyDSM distributed data storage capabilities, which calls shortECC and lmRNG to secure the stored data as well. This process is shown in figure 29.

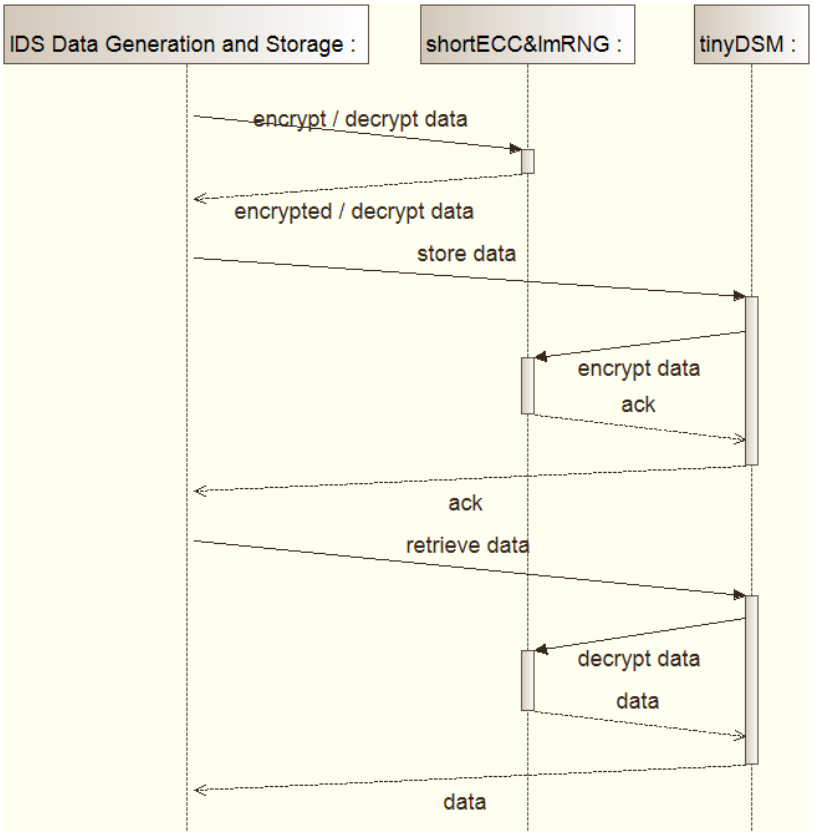

Figure 29: IDS data local storage (first interaction) and distributed storage (second and third interaction) using *tinyDSM*

5.8 IDS Data Distribution

The IDS operation involves the two components IDS Data Gathering and Storage, regarded in Section 5.7, and IDS Data Distribution, regarded in this section. The IDS Data Gathering and Storage component monitors the network traffic and collects and stores security related events [12][13], whereas the IDS Data Distribution component distributes detected IDS events or IDS data to SMARTIE devices.

This section explains the interactions that are accomplished by the IDS component IDS Data Distribution responsible for IDS data correlation and distribution among SMARTIE devices. There are two services in addition to the IDS Data Distribution component that are responsible for IDS service discovery and data retrieval (5.9), namely the IDS Service Announcement that is explained in subsection 5.9.1 and the IDS Data Distribution Service that is explained in subsection 5.9.2.

Components involved in this interaction:

- IDS Data Distribution
- shortECC&lmRNG
- SMARTIE platform

- Federation of Systems

5.8.1 IDS Event Distribution

This process describes the IDS event distribution. The IDS Data Distribution component first calls shortECC and lmRNG components for encryption/decryption of an IDS event before it is sent to the SMARTIE platform or to the Federation of Systems.

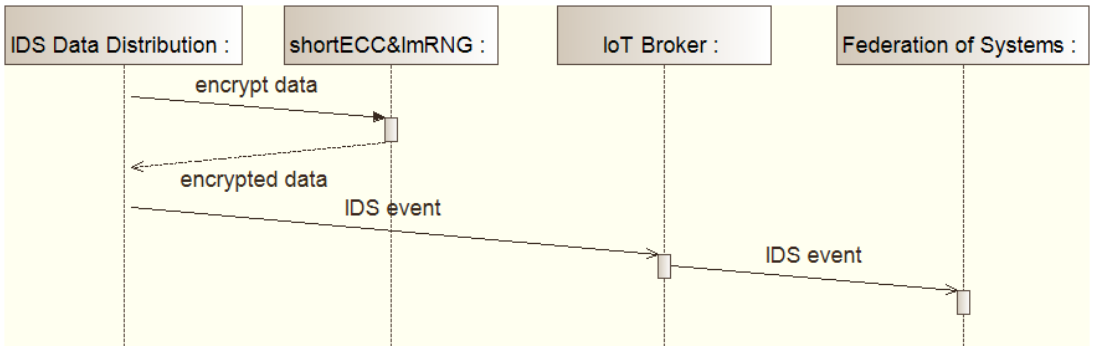

Figure 30: IDS Event distribution

5.9 IDS Service Discovery and Data Retrieval

This section explains the discovery of IDS services and how SMARTIE devices can retrieve data from these services. The IDS service discovery is split in two phases, the IDS service announcement, explained in 5.9.1, and the IDS data distribution service, explained in 5.9.2. The IDS service announcement makes the service known to the SMARTIE platform and the IDS data distribution service answers to requests from the SMARTIE platform and distributes IDS data and events among SMARTIE devices. The interactions of both services are described in the two subsections below.

Components involved in this interaction:

- IDS Data Distribution
- SMARTIE platform
- Federation of Systems

Services involved in this phase:

-
- Discovery (to form the Federation of Systems)
- Retrieve (to get the information from the services)

5.9.1 IDS Service Announcement

Figure 31 shows the interactions of the IDS service announcement. The IDS service announcement informs the SMARTIE platform about the IDS data distribution service. The SMARTIE platform registers the service for the Federation of Systems, so that devices/users can access it. Therefore it announces the service to the Service registration, from where it can be found and accessed by discovery mechanisms.

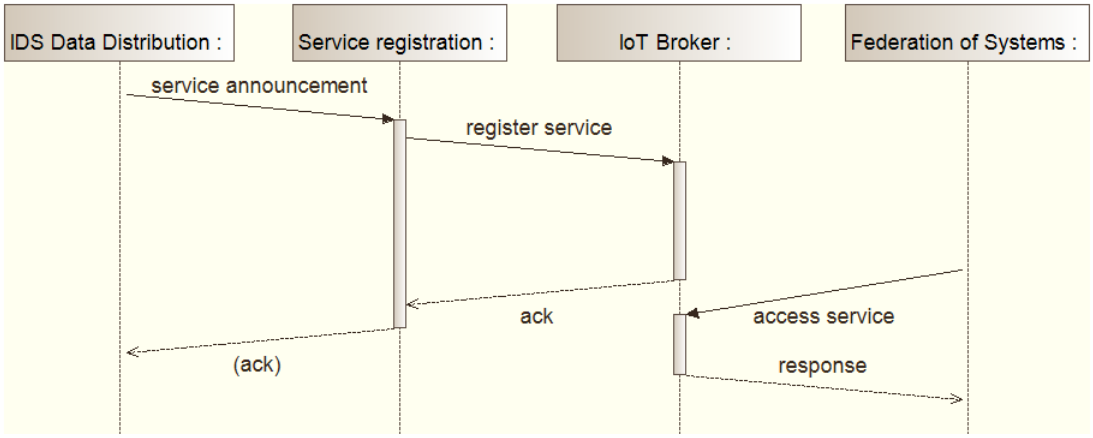

Figure 31: IDS service announcement and registration

5.9.2 IDS Data Distribution Service

Figure 32 shows the interactions of the IDS data distribution service. A device/user subscribes to receive IDS events at the SMARTIE platform that uses discovery services to locate matching IDS services on SMARTIE devices. The SMARTIE platform then sends all matching IDS events to the subscribed device.

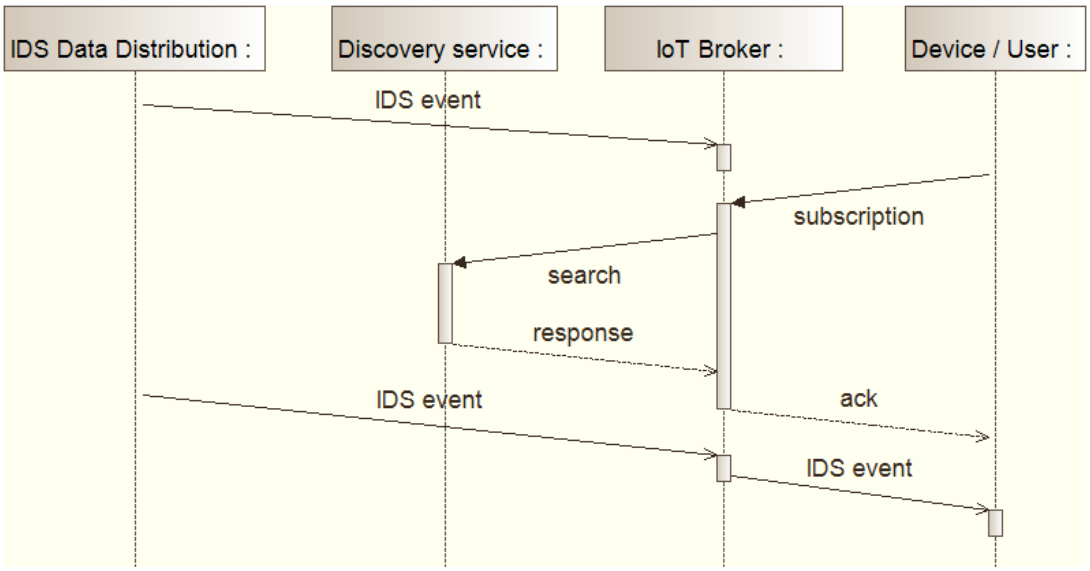

Figure 32: IDS service subscription and IDS data retrieval

5.10 Federation of Systems service

This interaction shows how the registered/authorized user using the Federation of Systems component can obtain the data from independent sub-systems. The Federation of Systems will process the request of the user. The appropriate services will be discovered and the necessary information or events will be retrieved.

Components involved:

- Federation of Systems
- Components involved in Discovery and Retrieve services

Services involved:

-
- Discovery (to form the Federation of Systems)
- Retrieve (to get the information from the services)

5.10.1 Federation of Systems to Retrieve Information

In this process, the user wants to use the Federation of Systems (FoS) in order to get the information about the temperature from all the temperature sensors available in the Smart City area, see Figure 33. First the Federation of Systems checks if the user is authorized to get the access to the requested data. Then as described in section 5.2 Discovery, the FoS looks for the devices fulfilling the requirements of the user and knowing the devices it can sent the requests to the sensors directly (as described in section 5.3 Retrieve).

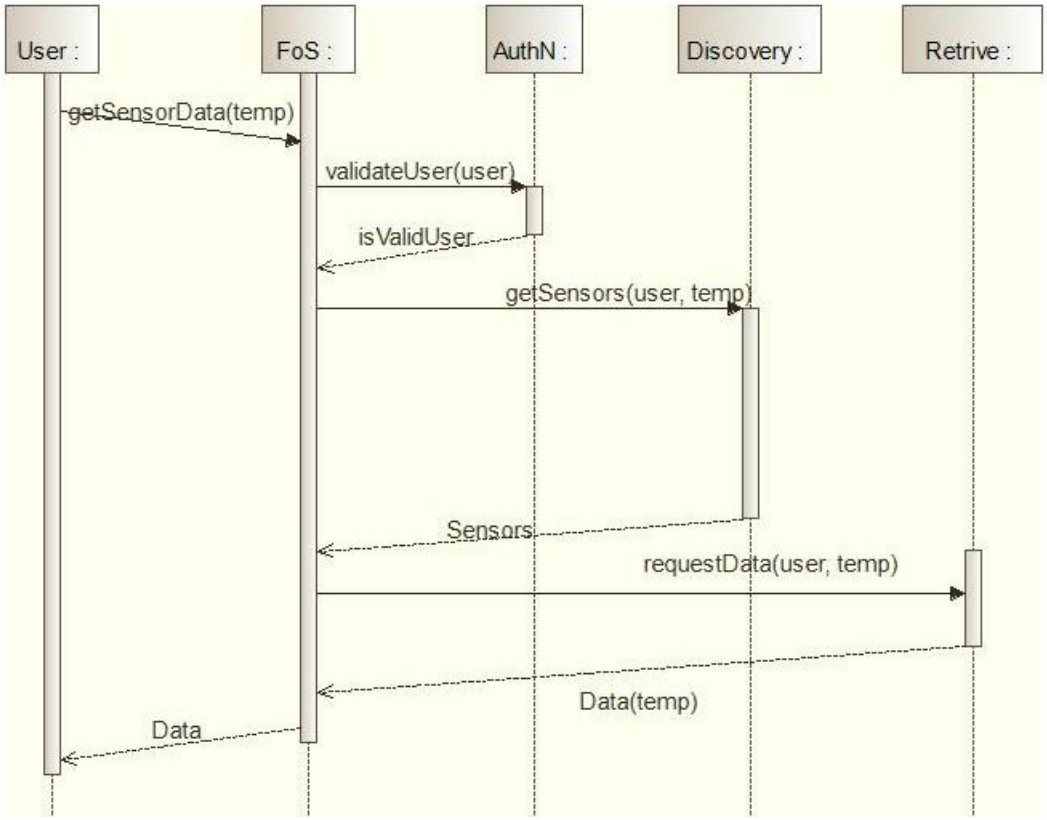

Figure 33: Federation of Systems authorizing the user, searching for appropriate devices and retrieving the data.

5.11 Node Attestation

This interaction allows detecting erratic or malicious behaviour to avoid misuse or abuse of the available resources by any unauthorized parties.

Components involved:

- Node Attestation (IMASC)

5.11.1 Node attestation and integrity measurements (IMASC)

The sensor devices are often the weakest link in the security chain. In order to increase the trust in the data delivered by the sensor, the IMASC component provides evidence that no one tampered with the software on the device. IMASC relies on a secure hardware anchor, a smart card, and uses the Linux integrity measurement architecture for measuring integrity of the software. Figure 34 shows the message flow of IMASC in request/response style data retrieval. The device submits integrity measurements to the smartcard when events are delivered by the integrity measurement architecture. In case of a data request by the user, the sensor device submits the response first to the smartcard. The smartcard includes attestation information and signs the response, in order to transfer the trust to the data.

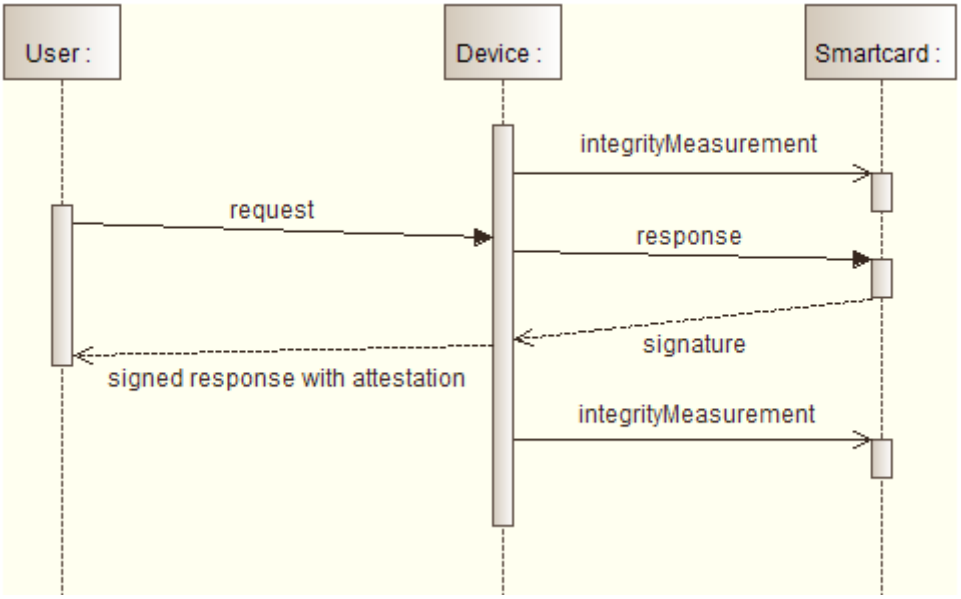

Figure 34: IMASC providing attestation combined with a data request.

6 Gaps, analysis and conclusions

This document describes the initial architecture of the SMARTIE platform and the interactions among all components in order to show the main functionalities of the system and to establish the basis for the final architecture. There are three main future works: finishing the implementation of some components, completing the definition of the interoperability and doing the integration of the interactions among the components.

The following subsection provides a review of the SMARTIE requirements (identified in the Deliverables 3.1 and 4.1) that are covered by each interaction presented in this document.

6.1 Requirements mapping

As seen in the deliverables D3.1 and D4.1 there is a mapping between the requirements and the components that are described in them. Here we will abstract those requirements to each general phase described in this document and show in a general way how those requirements are met in each phase.

Registration

The components that are involved in the Registration process support the requirements that are related with the management of information related with the services and capabilities of the sensors, and users. An important requirement accounted for by this component is the use of location information. Scalability is also an important requirement that these components provide, being the quantity of devices registered beforehand unknown but surely growing.

Discovery

The requirements related with the discovery phase supported also by the components present in that phase enforce the lookup using context information, being again the location information relevant to this service. The discovery phase has to enable look-ups according to preferences of the user and context information that is continuously changing, such as location or preferences. These kinds of data enable context-rich searches that give the possibility of dynamic results that are according to the dynamic and heterogeneous environments the IoT is conceived to be integrated in.

Retrieve

When dealing with the retrieval of information, privacy is an important matter mostly for the users of the platform, so enabling anonymity by means such as pseudonyms or unattached identifiers is something that the components involved in the retrieve phase must enforce. Furthermore, by design, IoT has some characteristics, that are not always desired such as the things might be unavailable. These situations are also dealt with in this phase.

Subscription

Subscription shares some ground in terms of requirements with Retrieve, as anonymity is also important. Also, since information that might be sensitive is sent, the information will be ciphered before it is sent. Furthermore, optimizations have to be made before sending the information, to avoid ciphering the information more times than necessary, gaining not only in performance, but also in efficiency.

Actuation

Actuation is a similar to the retrieve information phase where an actuator device can modify or alter the state of the system. In this process, the user obtains the capability token to be able to interact with the actuator.

Event Detection

This interaction provides a secure distributed shared memory service with event detection. Event detection represents no periodical information produced but unusual behaviours or possible misuse of the sensors and devices in the system.

Node Attestation

Part of the security requirements is to be able to prevent the misuse of the resources available in the system. This interaction takes care of detecting unusual behaviour that might point out to an intruder activity.

Federation of Systems

Following the goal of changing the view of IoT as individual vertical silos, (FoS) plays an important role as it enables the possibility of interacting between different subsystems exchanging information and making interoperable IoT system more feasible.

6.2 Conclusions

This document presents the initial architecture of SMARTIE project based on the architecture reference model (ARM) and the guideline of the IoT-A project. This document describes an initial approach of the SMARTIE architecture in order to establish the basis for the integration and interaction of the SMARTIE components.

Following the ARM methodology, the initial architecture of SMARTIE is provided as a baseline for designing a full IoT architecture to cover security and privacy requirements in IoT ecosystems. Using a subset of views from IoT-A ARM, we describe the typical interactions among SMARTIE components and define the communication exchanges of these components.

References

[1] FP7 257521 IoT-A Project, “Internet of Things Architecture, http://www.iot-a.eu/

[2] IoT-A Deliverable D1.5, “Final Architectural Reference Model for the IoT”.

[3] Bethencourt, J., Sahai, A., & Waters, B. (2007, May), Ciphertext-policy attribute-based encryption. In Security and Privacy, 2007. SP'07. IEEE Symposium on (pp. 321-334).

[4] K. Whitehouse, C. Sharp, E. Brewer, D. Culler: Hood: a Neighborhood Abstraction for Sensor Networks, In Proceedings of ACM International Conference on Mobile Systems, Applications and Services (MobiSys '04). Boston, MA, June, 2004. ACM Press.

[5] NGSI Context Management Specification, Open Mobile Alliance http://www.openmobilealliance.org/Technical/release_program/docs/NGSI/V1_0-20101207-C/OMA-TS-NGSI_Context_Management-V1_0-20100803-C.pdf (accessed Nov. 24, 2014), 2010

[6] The Constrained Application Protocol (CoAP), RFC 7252, https://datatracker.ietf.org/doc/rfc7252/

[7] Standards for Efficient Cryptography Group, http://www.secg.org/

[8] ISO/IEC 29192-2:2012, http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=56552

[9] Datagram Transport Layer Security Version 1.2, https://tools.ietf.org/html/rfc6347

[10] libcoap: C-Implementation of CoAP, http://sourceforge.net/projects/libcoap/

[11] CoRE Resource Directory, https://tools.ietf.org/html/draft-ietf-core-resource-directory-00

[12] L. Wallgren, S. Raza, and T. Voigt. "Routing Attacks and Countermeasures in the RPL-based Internet of Things." International Journal of Distributed Sensor Networks, 2013.

[13] A. Abduvaliyev, A. S. K. Pathan, J. Zhou, R. Roman, W. C. Wong. "On the Vital Areas of Intrusion Detection Systems in Wireless Sensor Networks." IEEE Communications Surveys and Tutorials 15.3, p. 1223-1237, 2013.

[14] BARKER, Elaine; KELSEY, John. Recommendation for the Entropy Sources Used for Random Bit Generation. Draft NIST Special Publication, 2012.

[15] RUKHIN, Andrew, et al. A statistical test suite for random and pseudorandom number generators for cryptographic applications. BOOZ-ALLEN AND HAMILTON INC MCLEAN VA, 2001

[16] SMARTIE Deliverable D2.1, “Use Cases” http://www.smartie-project.eu/download/D2.1-Use%20Cases.pdf

[17] Bethencourt, J., Sahai, A., & Waters, B. (2007, May), Ciphertext-policy attribute-based encryption. In Security and Privacy, 2007. SP'07. IEEE Symposium on (pp. 321-334).

[18] J. L. Hernández-Ramos, A. J. Jara, L. Marín, and A. F. Skarmeta, “Dcapbac: Embedding authorization logic into smart things through ecc optimizations,” International Journal of Computer Mathematics, no. just-accepted, pp. 1–22, 2014.

[19] Guide to Attribute Based Access Control (ABAC) Definition and Considerations - NIST Special Publication 800-162 –http://nvlpubs.nist.gov/nistpubs/specialpublications/NIST.sp.800-162.pdf

[20] JSON Profile of XACML 3.0 Version 1.0 – OASIS - http://docs.oasis-open.org/xacml/xacml-json-http/v1.0/csprd03/xacml-json-http-v1.0-csprd03.pdf