Access and privacy control enforcement in RFID middleware systems: Proposal and implementation on the Fosstrak platform

Wiem Tounsi, Nora Cuppens-Boulahia, Frédéric Cuppens, Guy Pujolle

World Wide Web Journal - DOI 10.1007/s11280-015-0325-5
The editor version is available at link.springer.com

Access and Privacy Control Enforcement in RFID Middleware Systems: Proposal and Implementation on the Fosstrak Platform

Wiem Tounsi[1,2] · **Nora Cuppens-Boulahia**[1] · **Frédéric Cuppens**[1] · **Guy Pujolle**[2]

Received: 10 March 2014 / Revised: 17 October 2014 / Accepted: 6 January 2015

Abstract Radio Frequency IDentification (RFID) technology offers a new way of automating the identification and storing of information in RFID tags. The emerging opportunities for the use of RFID technology in human centric applications like monitoring and indoor guidance systems indicate how important this topic is in term of privacy. Holding privacy issues from the early stages of RFID data collection helps to master the data view before translating it into business events and storing it in databases. An RFID middleware is the entity that sits between tag readers and database applications. It is in charge of collecting, filtering and aggregating the requested events from heterogeneous RFID environments. Thus, the system, at this point, is likely to suffer from parameter manipulation and eavesdropping, raising privacy concerns. In this paper, we propose an access and privacy controller module that adds a security level to the RFID middleware standardized by the EPCglobal consortium. We provide a privacy policy-driven model using some enhanced contextual concepts of the extended Role Based Access Control model, namely the purpose, the accuracy and the consent principles. We also use the provisional context to model security rules whose activation depends on the history of previously performed actions. To show the feasibility of our privacy enforcement model, we first provide a proof-of-concept prototype integrated into the middleware of the Fosstrak platform, then evaluate the performance of the integrated module in terms of execution time.

Keywords RFID, Access control · Privacy Policy · Middleware · EPCglobal · Provisional context

1 Introduction

Radio Frequency IDentification (RFID) has emerged as a low-cost technology due to the use of battery-less devices, known as passive tags. These devices can be attached or embedded in an item to be identified and are read when they enter a RFID reader's antenna field. The use of this technology in healthcare services has grown in importance with the need of assisting people (e.g., elderly individuals and patients) in their everyday life [24]. In this vein, the Auto-ID Center [34] created the concept of a unique RFID code called Electronic Product Code (EPC). On the one hand, it was shown that this technology significantly improves the visibility and accuracy of the logistic operations [28], for example, critical errors such as prescribing wrong medicines can be avoided. On the other hand, RFID technology helps providing patients some quality of care. For example, they

[1] Telecom Bretagne
2 Rue de la chataigneraie, 35510 Cesson Sevigné
E-mail: {*name.surname*}@telecom-bretagne.eu

[2] LIP6/UPMC - University of Paris 6
4 Place Jussieu, 75005 Paris
E-mail: {*name.surname*}@lip6.fr

can be applied to ensure a reminder system for elders, or to permanently assist patients suffering from chronic diseases, who must travel regularly for minor typical examinations [24]. Although the RFID technology offers several benefits, its human-centric deployment requires treating the data in a privacy-conscious manner. In fact, once RFID tags carry more than just an identifier, i.e, a bearer's age or illnesses correlated with other information, bearer's privacy may be violated. RFID systems consist typically on tags, readers, and backend applications such as middlewares and databases to manage and store the collected data. Here, we are interested in the backend side, particularly the middleware component. This middleware sits between the reader and database applications. It is in charge of collecting, filtering, aggregating and grouping the requested events from heterogeneous RFID environments, then compiling them into well-formatted data prior to send them to business application for usability e.g., typically by external applications, like the EPC Discovery Service (EPCDS), in EPCglobal technology. Large amount of data handled by the RFID middleware makes it attractive for malicious users to manipulate the parameters and access sensitive data. In this article, we reuse the middleware properties and integrate additional filtering and aggregation methods in the generated RFID data, for privacy goals and without interfering with existing standards.

The privacy of people focuses today on who has access to what information, rather than what information is accessed [42]. A major portion of previous work on privacy have been aligned on anonymizing user information or on preventing personal information from disclosure, e.g, by encryption means [8]. These works treat some issues of privacy but do not completely handle the different situations where the data owner (i.e., the user who specifies a set of privacy preferences on the usability of his data) chooses to share information with others. In addition, dealing with encrypted data in the middleware level makes the query process expensive, since not all the collected data are sensitive, rather, the aggregation of some information may need to be protected.

Numerous reasons motivate the support of privacy issues in the RFID middleware. First, treating the privacy in the middleware helps to master the collection configuration and the data view before interpreting and storing these data in upper layer applications. For example, let us consider two distinct applications wanting to receive data about a patient. The first application is only allowed to view the total tag count depending on the requester purpose, while the second one is allowed to view the identity of tags but only of product of type "GTIN". To treat the applications requests, we propose to handle the privacy policy before the events are generated and transmitted to database applications, since this minimizes, as soon as possible, the risk of unauthorized disclosures. Second, filtering data for privacy issues in the middleware stage has the advantage to relax the events to be collected in this stage, leading to an appropriate adaptation of the queries executed by the reader on the tag over the air interface. This also allows readers to use efficiently the limited bandwidth, e.g., to target only a particular tag population or switch off completely some readers. Finally, as RFID communication protocols use dedicated privacy enhancing features [40], the RFID middleware will also need to support their use. For instance, the RFID middleware must provide the right *kill*-password to the right reader to apply it on the tag as specified in EPC standard.

Our contributions in this paper can be summarized as follows. First, we address some privacy concerns in the RFID middleware systems, particularly those implementing the EPCglobal specifications [13] which is currently one of the predominant standardization efforts of the RFID community. Second, we propose a fine-grained access and privacy control that we integrate into the Filtering and Collection middleware. The approach we propose is policy-driven. It extends our previous work [39] and ensures the following features: (i) enforces a privacy policy without interfering with the standardized middleware interface, (ii) uses the PrivOrBAC privacy-aware model [4] to store and manage privacy policy preferences and (iii) takes into account the principles of declared purpose, consent and accuracy (i.e., collection limitation) as privacy requirements, in addition to the provisional context, to manage the history of performed actions in the system. To show the feasibility of our approach, we first provide a proof-of-concept prototype that we apply on the open-source software for track and trace Fosstrak [17], then, evaluate the performance of our integrated module in terms of execution time.

Paper organization. Section 2 discusses related work. Section 3 describes some background about the EPCglobal middleware and interfaces. Section 4 details the privacy principles that can be enforced in the middleware. Section 5 exposes our privacy-enhanced middleware solution and introduces the model we choose to specify the privacy requirements. Section 6 describes our privacy-enhanced approach applied on Fosstrak. Section 7 concludes the paper.

2 Related work

In the first levels of the RFID architecture, three types of data deserve to be considered with security and privacy aspects. We note the reader and tag events [40], the middleware events and the EPCIS data, after events have been captured and stored (cf. Figure 1 shows a summary view of the EPCglobal architecture). We consider that the reader interface is part of the middleware interface as the two roles are in the "middle" of the architecture and relatively play the same role with different levels of abstraction [10].

While security research in the context of RFID has mainly focused on reader to tag communication [16,22,**?**] and RFID-based data repositories [3,18,35] and more global communication services such as the RFID discovery service [25,26] (cf. Figure 1), the middleware level has not received much attention so far. Most of the middleware implementations available today are commercial-based [21]. There are also some middlewares which have been developed for research purposes, such as [17] and [20]. However, most widely deployed middleware products do not fully consider the security requirements when processing sensitive data. For example, several implementations from software vendors [19,30] and specialized companies [32] and open source initiatives [17] offering RFID middleware functions, manage the access control to historical events stored into final databases. However, they do not provide treatments to enhance privacy policies (e.g., by access control techniques) in accordance with international regulations, expected to be done at the middleware level [14]. The same issue is observed in some research works for securing the middleware level. For instance, authors in [37] provide an Application Level Events Service Security Component (ALE-SSC) to strengthen security and trust in the ALE-based service middleware. The proposed mechanism is only to protect transported data between the middleware and its clients. In [36], the authors provide techniques to support context-aware access control service in the middleware. The context information, in this case, is meant to be the reliability of the requester and the environment.

Regarding the EPCglobal standard, most existing implementations support its use. In case they include security mechanisms, they explicitly provide it in the EPCIS (EPC Information Service) level [19,30,32] relying mostly on role based access control and encryption models. Some of these aforementioned products include security enhancements into their middleware implementations [19,30]. However, few works focus on privacy problems. An exception is the work [20], where the authors propose to support the EPCglobal middleware via a tool called *Privacy Framework Tool* plug-in. This tool includes privacy friendly practices and audits to be applied to the proposed middleware. To the best of our knowledge, there have been neither public communication related

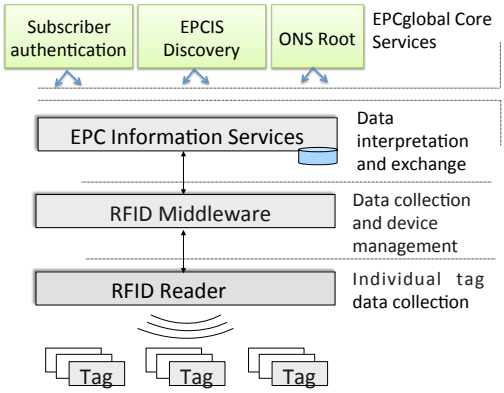

Fig. 1 Main levels in the EPCglobal architecture

to the used privacy policy nor information about the plug-in implementation. We also note the work in [5] that provides security and privacy features included in the middleware, which is referred as *data processing layer*. In this work, the authors enforce the middleware privacy by policy based management. According to the authors, the privacy policy specifies whether an application has the right to access RFID tag data, track it over time, and use it to generate events. It is not mentioned that the privacy policy has considered the privacy principles of declared purpose, accuracy and consent, issued from known regulations.

To conclude, RFID interfaces that govern the middleware today propose a role based access control as a security approach. However, they fail to address consumer privacy concerns by supporting appropriately known regulations and guidelines [2]. A possible reason for this issue is that most enterprises run the RFID middleware in an internal network [15]. In addition, as explained in [7], the efforts to achieve security and privacy exist in the middleware but not as an architectural incorporation for RFID middlewares; rather as domain specific security rules, implemented up to compliance to specific solutions. Finally, in terms of performance, there is no standard metric that can be used to study individual middleware implementation and compare the existing solutions among each other. This is due to the limited available information on the performance parameters of these solutions, especially commercial middleware systems. On the other hand, the unavailability of real world data for a large-scale RFID deployment affects the analysis of research middleware systems with real world application scenarios.

In this paper, we propose the use of the PrivOrbac service [4] to implement the role based access control proposed by the EPCglobal standard specifications. EPCglobal has already specified in the middleware level, an API of access control. The proposed model is Role Based Access Control (RBAC) to define rules for granting or denying access to methods or resources proposed by other APIs. The security mechanism of the access control API limits the access to critical methods (e.g., subscription to capture tag data). Nevertheless, it does not cope with data aggregation details and filtered and combined reports in a same request. In addition, the privacy principles of purpose, accuracy and consent, issued from known regulations are not considered, as PrivOrbac does. We believe that a fine-grained security on these collected data for privacy concerns has to be handled in the middleware level to prevent applications, even in the same organization, from collecting data that undermine people's private lives or reputations.

In the sequel, we show how it is possible to integrate a privacy controller on the top of a middleware solution to satisfy the basic principles of privacy, without interfering with the existing specifications of the EPCglobal standard. We evaluate our proposal in terms of execution time and compare it with the original middleware solution implemented in Fosstrak.

3 The EPCglobal middleware and its interface

The EPCglobal network, initially proposed by the MIT's Auto-ID Center [34] and further developed by members of the joint-venture EPCglobal, is currently one of the predominant standardization efforts of the RFID community. The EPCglobal network is a set of global technical standards aiming at enabling automatic and instant identification of individual items and sharing this information throughout the supply chain. The Filtering & Collection (F&C) middleware is one of the main components of the EPCglobal network architecture [6]. It uses a single interface to a large number of distributed readers and a large number of capturing applications that may be interested in the collected data. This interface is called the Application Level Event (ALE) (cf. Figure 2). ALE 1.1 comprises five standard APIs, namely Reading, Writing, Tag Memory, Logical Reader and Access Control APIs. In this paper, we are interested in the Reading API which provides a means for clients (i.e., Capturing applications) to specify in a high-level, declarative way, what tag data they are interested in and gives the corresponding reports in variety of ways.

It is undisputed that the act of reading out one or more RFID tags constitutes a data collection. This means that existing privacy regulations and laws (cf. Section 4.1) also apply to the communication involving the RFID middleware responsible for configuring RFID readers for data collection. In the sequel, we present the main information that an EPC tag carries and give in

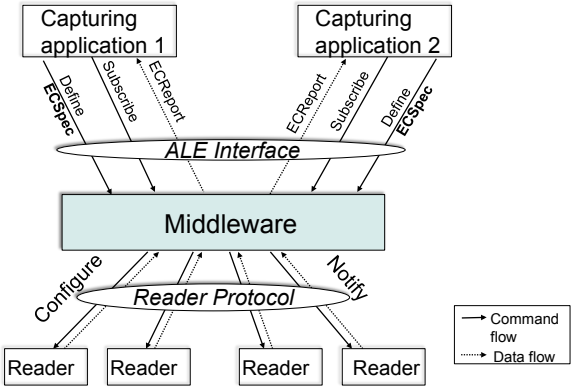

Fig. 2 EPCglobal middleware

more detail, the role played by the Reading API of the middleware interface. Subsequently, we present the limits of this interface regarding privacy controls.

3.1 Information handled by the RFID/EPC tag

Electronic Product Code is an example of RFID code. It is a numbering scheme that provides a unique identification for physical objects and assemblies. The information about these numbers is not stored in the code, but serves as a reference to Internet-based information. There are many situations where the code stored in an RFID tag is considered as a sensitive information, e.g., in a context where people fear to be tracked or refuse that their belongings are read. Other information may be stored in the additional memory of the tag (i.e., depending on the tag type). They are extracted by the tag Memory API of the ALE.

The EPC is mostly a 96-bit code [9] divided into four fixed partitions, cf. Figure 3. First, the 8-bit header defines the number, type and length of subsequent data partitions. Second, the manager code defines the manufacturer of the item. The next 24-bits defines the object type code. Finally, the 36-bits unique object defines the identification (or serial) number.

Since the EPC was made to integrate existing numbering schemes, different headers can be used in the EPC technology, e.g., the well known Global Trade Identification Number (GTIN) that is widely used in the retail industry. A special version of GTIN (i.e., UPC Version B) supported by EPC, originally intended to handle the National Drug Code and National Health Related Item Codes. Since the meaning of this code may reveal personal information, it presents some privacy concerns when read without the consent of the tag holder, i.e., the patient. This unauthorized reading can be handled in the middleware via the ALE interface. Let us see the ALE main actions and data types.

3.2 Application Level Events (ALE) interface

One or more clients (named also capturing applications) can request the ALE interface [10] via a set of methods. Each request causes the ALE engine to take an action and return synchronously

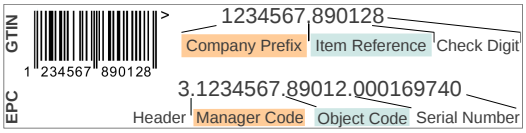

Fig. 3 EPC composition and GTIN integration

or asynchronously a result. To receive asynchronous results, the ALE clients have to subscribe to an already defined specification *ECSpec* (see next paragraph) and to specify a Uniform Resource Identifier (URI). This URI describes the client address to which the information is delivered, e.g., *subscribe(ECSpec, URI)*.

3.2.1 Primary data types

The primary data types associated with the Reading API of ALE are the Event Cycle Specification (ECSpec) and the Event Cycle Reports (ECReports). ECSpec is an XML type used to specify how an event cycle is to be elaborated and reported. ECReports is an XML type containing one or more reports which are generated from one execution of an ECSpec. ECSpec can be triggered in two ways: (1) A standing ECSpec is performed using the *define* method. Subsequently, one or more clients subscribe to that ECSpec using the *subscribe* or the *poll* method. (2) An immediate execution of the ECSpec is submitted via the *immediate* method.

3.2.2 Event Cycle Specification (ECSpec)

ECSpec describes an event cycle specification and defines the generated view that could result in a privacy threats (see Section 4.2). It contains three main parts :

- Logical readers or a list of readers (*ECLogicalReaders*). Each member of this list is either a name of a single reader or a composite reader used to read tags.

- Boundary Spec. A specification of how the boundaries of event cycles are to be determined (*ECBoundarySpec*). They specify the start and stop conditions or the duration or the repetition period of the event cycle.

- Reports specifications (*ECReportSpec*). Each report specifies one output to be generated from the event cycle and to be included in the list of reports. Its main fields are:

- include/exclude patterns to filter what tags are to be included in the final report. A single EPC pattern is a URI-formatted string that denotes a single EPC or a set of EPCs.
- grouping pattern defines how filtered tags are grouped for reporting. This parameter separates tags into different groups and is only used when some output format are set.
- report set is an enumerated type that specifies the set of tags to be considered for the output. i.e., EPCs read in the *current* event cycle, *additions* or *deletions* from the previous event cycle.
- output format specifies how the final set of EPCs is to be reported. If *includeCount* is true, the report includes also a count of the EPCs in the final set for each group.

3.3 Privacy control limitations in the middleware

EPCglobal has proposed an ALE API of access control to use the functionalities of the F&C middleware. This API allows administrative clients to define the access rights of other clients when using the methods and resources proposed by other ALE APIs (e.g., reading, writing to the tag memory). The model is role based access control [33] (RBAC). A role maps to one or more permissions to a particular feature of the ALE API. Two kinds of permissions exist: the *function permissions* and the *data permissions*. The first one grants the right to use a particular method of the ALE API (e.g., define, subscribe, unsubscribe), whereas the latter grants the right to use a particular resource or data (e.g., the right to govern a particular reader). The security mechanism of the access control API limits the accesses to critical methods (i.e., subscription to capture tag data) but does not cope with data aggregation details and filtered/combined reports in a request specification. For this aim, we propose to add a new layer on top of the middleware component that controls the collection performed by the middleware clients. Before delving into the details of our privacy controller proposition, we deal with privacy principles provided by relevant regulations, specifically those that can be handled in the middleware level.

4 Introducing the privacy principles

We present the privacy principles that can be handled in the middleware level, then cross them with each field of the ECSpec to extract those we have to focus on to enforce privacy in the middleware level.

4.1 Privacy principles in the middleware

Global System 1 has developed the EPC/RFID Privacy Impact Assessment (PIA) tool to help companies uncover the privacy risks and perform their own privacy assessments when implementing their RFID applications [12]. These privacy recommendations are defined in accordance with relevant privacy and data protection laws and regulations [14]. Based on these directives, we define the main privacy principles in RFID environments, as (1) Purpose specification (2) Collection limitation (3) Data owner access (4) Notification of the data owner (named also openness) (5) Explicit consent (6) Security mechanisms (7) Collection relevance. Some of these principles can be expanded in the middleware level while others are dropped to suit its specifications. These latter could be rather enforced in upper levels. We summarize in Table 1 the privacy requirements that can be achieved in the F&C middleware. Here, we are interested in the explicit consent, the purpose and the accuracy principles.

4.2 Privacy principles in ECSpec

Table 2 shows the privacy principles that can be treated in each ECSpec field. Note that it is better to specify the purpose of the data collection apart from the ECSpec (cf. Section 5). The configuration of one reader or a list of readers, named *logical readers*, can directly influence the received reports. The use of a non-authorized reader can result in collecting non-authorized state information (e.g., monitoring the number of items in a defined area) or in providing a localization service (e.g., applied to cases of smart rooms). The use of a list of readers is mainly viewed in scenarios of tracking an item or a person. For privacy reasons, the configuration of this field should be treated to obtain accurate results. The specification of the *boundaries* is important from a privacy point of view, e.g, for *start/stop trigger* conditions, if the person is moving from/to an assumed private place, this event could be a trigger to start/stop tracking the person. The *duration* declares the period of time during which the EPC tags are authorized to be identified. For privacy reasons, the configuration of the *temporal consent* is required. Finally, the specification of the *reports* fields affects the data view. For accurate results, the process of reading can be treated by including and excluding some EPC tags using some regular expressions [10] or by only reporting a set of tags grouped by an EPC code field. In the sequel, we present our solution and the privacy policy model we use to specify our rules.

5 Privacy enforcement in the RFID middleware

The privacy controller module integrated in the EPCglobal middleware is depicted in Figure 4. Relying on the ALE interactions for reading tags (cf. Section 3.2) and the privacy principles

Table 1 Privacy Principles in the Middleware Level

Privacy Principle	Middleware support
(1) Purpose specification	to announce the valid purpose associated with the request.
(2) Collection limitation (Accuracy)	to ensure that only necessary information is collected by holding the accuracy and anonymity specifications in the ECSpec filter.
(5) Explicit consent	to ensure that the data is collected with the knowledge and consent of the data owner.
(6) Security mechanisms	to protect personal data from unauthorized access or disclosure (e.g., encryption, anonymity).

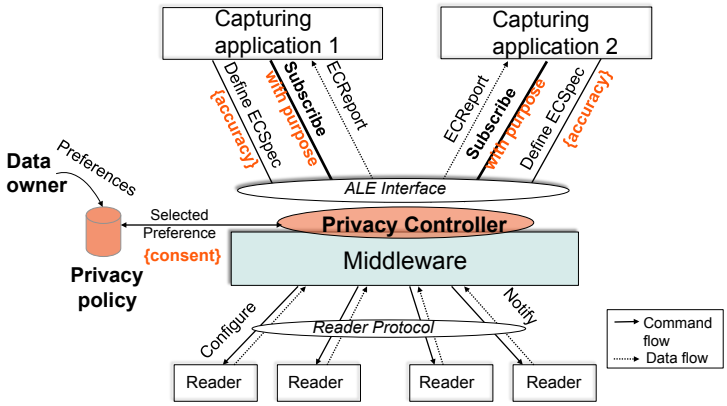

Fig. 4 EPCglobal middleware with a privacy controller module

required at this level (cf. Section 4.1), we consider that the subscription to an ECSpec is the action concerned with the privacy issues (the same for the other request modes). In fact, the collection of data is only triggered by this action, since the defined ECSpec is not executed before the subscription is made. The other activities (e.g., the definition of an ECSpec) could be controlled by classical access control policies. Therefore the privacy controller is triggered once a subscription request is received from the client, with a declaration of the collection purpose.

Regarding the *purpose* declaration, it can either be added to the ECSpec content or as a parameter with the *subscribe* request. We add the *purpose* to the *subscribe* request for the following reasons. First, this way, the ECSpec can be used by many capturing applications and with possibly different purposes. Second, adding the purpose parameter to the subscription request avoids distorting the ECSpec and leaves it consistent with the EPCglobal standard specification. When triggered by the client subscription, the privacy controller calls the privacy policy database, where the data owner preferences are stored. It compares these entered preferences and the content of the ECSpec request and should be able to decide whether to grant or deny the collection of the

Table 2 ECSpec fields and privacy threats

ECSpec Field	**Privacy threats**		**Privacy Principle**
Logical readers	represents a privacy threat to localize or identify an item. The association of readers can be used to trace the item or a person		(2) accuracy specification / (5) spatial consent
Boundary specification	- Start/Stop Trigger - Duration - Repetition Period	represents a privacy threat if the definition of each of one is not in accordance with the purpose of use and the data owner preference	(5) temporal/spatial consent
Reports	Include/exclude pattern	represents a privacy threat if some EPCs have not to be included in the final report. This filter field is useful for excluding private EPC codes	(2) accuracy specification
	Grouping pattern	is useful to know the quantity of each object type rather than the object's serial number	
	Report set	is a way to represent the collected data. It does not present a privacy issue	No privacy treatment
	Output Format	the format of the set of EPCs to be reported does not present a privacy threat as the filter is performed earlier	

specified data or to update the ECSpec (cf., Section 5.3.2). The accuracy principle is rather handled when focusing on the content of the ECSpec request. We begin by presenting the model we have chosen, to specify the privacy rules that we apply to a motivating scenario. Then, we delve into the details of our privacy controller module and its interations.

5.1 Privacy policy specification

There are several models of privacy in the literature. These models are used to model and integrate privacy requirements into a security policy [4, 27, 29, 43]. The security policy is generally specified according to an access control model to simplify the upgrade of existing information systems. Here, we detail the privacy principles of the PrivOrBAC model [4], which extends the Organization-Based Access Control model [23] (OrBAC) by reusing most of its implemented mechanisms, e.g., managing the security policy by specifying the contexts as complex conditions to define privacy control requirements. The choice of PrivOrBAC relies on its capacity to handle most of the privacy principles appearing in the guidelines and recommendations previously mentioned, e.g., principles of consent, accuracy, purpose of data collection; but of course it can be substituted with another privacy model without disrupting the whole process.

5.1.1 The OrBAC model

The OrBAC model is a generic and expressive access control model that extends the Role Based Access Control [33] (RBAC) model. OrBAC provides a set of concepts to express the security policy and enables making distinction between an abstract policy specifying organizational requirements and its concrete implementation in a given information system. Abstract organization privileges, such as *permission*, are expressed through the predicate *Permission(org, r, a, v, c)*. It means that the organization *org* grants a permission to role r to realize the activity a on the view v in the context c.

When declaring a context, a subject obtains some specific permissions and possibly some obligations or prohibitions. For instance, a subject empowered in the role *nurse* may be permitted to declare that she is performing an *inspection*. By doing so, this subject will get the permission to have an access to the tag data that are in relation with her service. PrivOrBAC reuses these concepts and specializes them in specific contexts to handle the required privacy principles.

5.1.2 Privacy contextual management

PrivOrBAC models the explicit consent as a context, the purpose as a user declared context and the different views as an accuracy of the objects, which could be defined by the data owner as preferences. The data owner preferences are included into the security policy of the organization.

The Consent Principle: We define (i) the consent preference view and (ii) the consent context. For (i), users store their consent preferences in the consent preference view (*cp*). Each object in this view corresponds to a particular data owner preference and has four attributes: *Dataowner*, *Recipient* (who receives the data related to the object), *Target* (the requested object), and *Need-Consent* (a Boolean parameter, which is set to true when the data owner consent is needed). (ii) The consent context takes into account the data owner preferences and/or notifies him when his personal information is accessed. Two cases are identified. The first case is when the consent is needed (*NeedConsent = true*). The data owner response is modeled by a built-in predicate *Consent_response*. That is, if *org* is an organization, s is a subject, *do* is a data owner, $resp \in \{accept, deny\}$, then *Consent_response(org, do, s, cp, resp)* is the response returned by the data owner to the organization. The second case is when the consent of the data owner is not required before revealing his private data to the recipient. In this case, the *NeedConsent(cp)* attribute is *false*. The access decision can be made without waiting for the *Consent_response*. By this means, the *Dataowner* chooses which view the *Recipient* can access.

The Purpose Principle The purpose is modeled as a user-declared context. Each data owner can create purpose objects to specify the purposes for which access to his personal objects are

allowed. The purpose objects belonging to a finite set of PO are grouped and inserted in a purpose view. Purpose values range over the domain PV (e.g., Medical_research, Inspection). Each purpose object has two attributes:

- *Recipient*, which is a predicate over domains PO `x` S defining who takes advantage of the declared purpose. That is, if *po* is a purpose object belonging to PO and *s* is a subject, then *Recipient(po,s)* means that *s* is the subject who takes advantage of the declared purpose *po*.

- *Declared_purpose*, which is a predicate over domains PO `x` PV, associating a purpose value with the declared purpose object. That is, if *po* is a purpose object and *pv* is a purpose value, then *Declared_purpose(po, pv)* means that *pv* is the purpose value associated with the declared purpose *po*.

By inserting a *po* in his purpose subview, a data owner declares that another subject (a *Recipient*) will perform some activity in a given context.

The Accuracy Principle Privacy enforcement requires the use of different levels of accuracy depending on the purpose and the subject requesting the collection of the private data. This principle is consistent with the privacy directive of collection limitation. Private objects of each data owner may have different levels of accuracy [4]. We can consider a hierarchy between the root view of a data owner that groups the initially collected objects and their sub-views. These sub-views group the derived objects that have different accuracies. We recall that *Sub_view* is a relation over domains *Org* `x` V`x` V, if *org* is an organization, and *v1* and *v2* are views, then *Sub_view(org, v1, v2)* means that in organization *org*, view *v1* is a sub-view of *v2*. Each data owner defines his own private data hierarchy, composed of different data views, e.g., a view of the EPC codes or a view of the EPC quantity. The data owner can then specify different privacy and security rules for each view.

5.1.3 Provisional context

Provisional context applies after specifying the privacy policy. It is used to model security rules whose activation depends on the history of previous actions performed by the subscriber. For this purpose, we first assume that the information system manages a *log*, that stores data about previous activities of the users in the system. This is modeled by a view called *Log*. Objects belonging to view *Log* have six attributes: *actor*, *action*, *target*, *activity*, *context* and *date* that respectively represents the subject (*actor*) who is performing an action (*action*) on an object (*target*) within an activity (*activity*) in a context (*context*) at a given date (*date*).

In the sequel, we specify our privacy rules using the PrivOrBAC model applied to a motivating scenario.

5.2 Motivating scenario

This section illustrates a scenario in a Hospital Ho for a remote monitoring system. To define the OrBAC organizational policy of Ho , we consider the following entities.

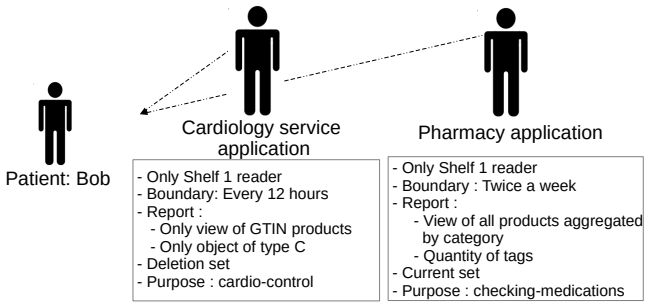

Fig. 5 Scenario of monitoring applications

The elderly patient Bob is remotely monitored at home from different services of the Hospital (cf. Figure 5). Bob suffers from hypertension and from alzheimer. The medications he daily takes are put in his medication shelf and are identified with EPC codes, i.e., every one pill is related to one EPC code. The nurse of the cardiology service has to monitor Bob twice a day. Every time he takes his hypertension medication from the shelf, the total number of pills decreases. The nurse has not to track the tags that are related to the alzheimer disease. This latter information is considered as private with respect to the patient preferences. In the other side, the Pharmacy application has to monitor the medication shelf twice a week, to check if it contains the required number of medications. Thus, it only needs a quantity information about the medications rather than their entire EPC code. It is also possible for the Pharmacy application to have access to the patients number suffering from the alzheimer's disease, if their names are unknown.

Based on these assumptions, the nurse of cardiology service has only the right to access objects of type C, i.e., the control is done over the field *object code* of the EPC code (cf. Figure 3), describing the reference of the hypertension medication. The ECSpec is configured to only consider *deletions_set* of tags, i.e., tags deleted from the previous event cycle. Finally, the pharmacist is only permitted to view the tags quantity depending on their *manager code*. Exceptionally, he has the permission to know the number of patients suffering from an alzheimer, without knowing their names. For this aim, he can specify in his request a fixed EPC *manager* and *object codes* related to this disease. Since this information is considered as private if both the disease and the patient name are learned by the pharmacist, the provisional context will also be triggered to check for historical requests.

OrBAC specification:

The privacy rules corresponding to the motivating scenario are expressed in the OrBAC model as follows:

(1) the nurse related to the cardiology application is only permitted to receive data of patients suffering from hypertension, with the context *Consent_nurse* and the view *TagC_type* in the user-declared context *patient_nurse*:

$Rule_1$ = *Permission(Ho, NurseApp, Subscribe, TagC_type, Consent_nurse)* where the context *Consent_nurse* is specified as,

$Rule_{consent\_1}$: $\forall$ *Ho* $\in$ *Org*, $\forall$ *s* $\in$ *S*, $\forall\alpha \in A$, $\forall$ *o* $\in$ *O*, $\forall$ *cp* $\in$ *O*, *Hold(Ho, s,* α*, o, Consent_nurse)* $\leftarrow$ *Use(Ho, cp, Consent_preference)*$\wedge$ *Recipient(cp,s)* $\wedge$ *Target(cp, o)* $\wedge$ *Dataowner(cp, do)* $\wedge$ ($\neg$ *NeedConsent(cp)* $\vee$ *Consent_response(Ho, do, s, cp, accept))*.

That is, the *Consent_nurse* context holds if there is an object *cp* belonging to the *Consent_preference* view which has the attributes *s* and *NeedConsent(cp)*.

$Rule_2$= *Permission(Ho, NurseApp, Subscribe, TagC_type, User_declared(patient_nurse))* where the context *patient_nurse* is defined as follows:

$\forall$ *Ho* $\in$ *Org*, $\forall$ *s, s'* $\in$ *S*, $\forall\alpha \in A$, $\forall$ *po* $\in$ *PO*, $\forall$ *pv* $\in$ *PV*, *Hold(Ho, s,* α*, po, User_declared(patient_nurse))*$\leftarrow$ *Use(Ho, po, do-purpose)* $\wedge$ *PatientID(po,s')* $\wedge$ *Recipient(po,s)* $\wedge$ *Declared_purpose(po,pv)* $\wedge$ *Nurse(s',s)*.

That is, in the organization *Ho*, a subject *s* performs an action α on the purpose object *po* in the context *patient_nurse* if there is a purpose object *po* used in the subview *do-purpose* of the data owner (associated with his defined objects) by *Ho* such that *s* is the recipient associated with *po* (represented by the application-dependent predicate *Recipient(po,s))* and *s* is the nurse of the patient *s'*. cf. Section 5.1.2, for details.

(2) the pharmacy application is only permitted to receive data of the patient, with the context *Consent_pharmacist* and the view *AllTagNumber* in the user-declared context *patient_pharmacy*:

$Rule_3$= *Permission(Ho, PharmacyApp, Subscribe, AllTagNumber, Consent_pharmacist)* where the context *Consent_pharmacist* is specified by:

$Rule_{consent\_2}$: $\forall$ *Ho* $\in$ *Org*, $\forall$ *s* $\in$ *S*, $\forall\alpha \in A$, $\forall$ *o, cp* $\in$ *O*, *Hold(Ho, s,* α*, o, Consent_pharmacist)* $\leftarrow$ *Use(Ho, cp, Consent_preference)*$\wedge$ *Recipient(cp,s)* $\wedge$ *Target(cp, o)* $\wedge$ *Dataowner(cp, do)* $\wedge$ ($\neg$ *NeedConsent(cp)* $\vee$ *Consent_response(Ho,do,s,cp,accept))*.

That is, the *Consent_pharmacist* context holds if there is an object *cp* belonging to the *Consent_preference* view, which has the attributes *s* and $NeedConsent(cp)$.
$Rule_4$ = *Permission(Ho, PharmacyApp, Subscribe, AllTagNumber, User_declared(patient_pharmacist))* where the context *patient_pharmacist* is defined as follows:

$\forall$ *Ho* $\in$ *Org*, $\forall$ *s, s'* $\in$ *S*, $\forall\alpha \in A$, $\forall$ *po* $\in$ *PO*, $\forall$ *pv* $\in$ *PV*, *Hold(Ho, s,* α*, po, patient_pharmacist)*$\leftarrow$ *Use(Ho, po, do-purpose)* $\wedge$ *PatientID(po,s')* $\wedge$ *Recipient(po,s)* $\wedge$ *Declared_purpose(po,pv)* $\wedge$ *Pharmacist(s',s).*

That is, in the organization *Ho*, a subject *s* performs an action α on the purpose object *po* in the context *patient_pharmacist* if there is a purpose object *po* used in the subview *do-purpose* of the data owner such that *s* is the recipient associated with *po* and *s* is the pharmacist of the patient s'.

It is assumed that the patient (i.e., the data owner) has already declared the contexts *patient_nurse, patient_pharmacist, Consent_nurse and Consent_pharmacy.*

Finally, the pharmacist has the permission to learn the number of patients who are suffering from the alzheimer disease, without specifying their names in the same request or in a previous executed request. To model this rule, we define a provisional context called $Alzeimer - statistics$ as follows:

$\forall$ *s* $\in$ *S*, $\forall\alpha \in A$, $\forall$ *o, (Hold(Ho,s,* α*,o, Alzeimer* $-$ *statistics)* $\leftarrow \exists$ *l, Use(Ho,l,Log)* $\wedge$ *actor(l,s)* $\wedge\neg$*activity(l, Consult-NameMemoryField)* $\wedge$ *(context(l, Checking*$-$*patients))*

That is, in *Ho*, subject *s* performs action α on object *o* in provisional context $Alzeimer - statistics$ if there is not activity $Consult\text{-}NameMemoryField$ that was logged in view *Log* in context of *Cheking-patients* with subject *s* as an actor.

Note that the roles nurse and pharmacist are managed by the access control API of the ALE middleware interface. In the following section, we describe our new privacy control module and the related algorithms.

5.3 Privacy controller activities

5.3.1 Privacy controller interactions

To subscribe to an event cycle and receive related reports, the client calls, via the ALE interface, an entity responsible for reports generation, referred here as *CentralEntity*, (cf. Figure 6). We define the *PrivacyController* as the entity handling the compliance of the subscription requests with the predefined privacy policy.

Figure 6 depicts the relation between the two defined entities (i.e., *CentralEntity* and *PrivacyController*). The *CentralEntity* instantiates the *PrivacyController* with the specification ECSpec, the recipient address URI and the purpose to verify the properties of the request. The *PrivacyController* outputs a Grant or Deny of access to the specified data after calling the privacy policy, otherwise updates the ECSpec for accurate results. The methods supported by the *PrivacyController* entity are explained in Section 5.3.2.

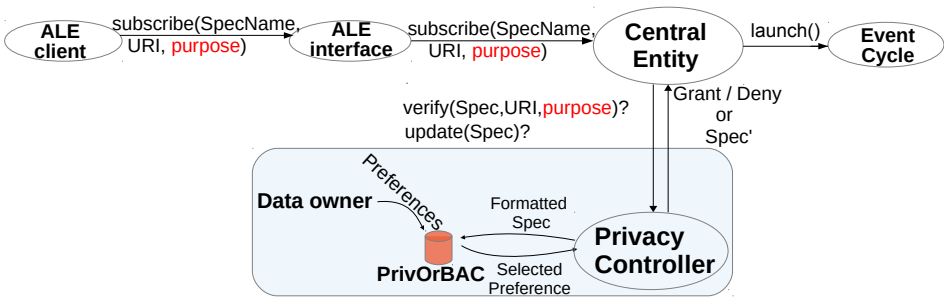

Fig. 6 Privacy controller activities

Regarding the privacy preferences, the data owner specifies them using the PrivOrBAC model [4] depending on the privacy principles enforced in the system. These preferences are introduced in a predefined ontology. As shown in Figure 7, each data owner *preference* is associated with a *recipient* address and a *purpose* on which the preference is defined. This preference turns around a set of targeted objects (*targets*) that the *recipient* aims to access. A *decision* attribute, related to each *target*, defines whether the data owner gives his consent to access the data or not. In the positive case, the *decision* attribute designates the accuracy in which the data will be disclosed. Note that the final decision has to be in accordance with general privacy rules of relevant regulations, which are initially specified in PrivOrBAC.

As an illustration, we apply our Privacy controller methods to the motivating example, presented in Section 5.2. Figure 8 represents the preferences of the data owner as implemented in PrivOrBAC. Recall that for the cardiology nurse, the patient Bob decides to disclose only his medications of type C in clear (i.e., the entire EPC code) for the purpose *Cardio_control*. Regarding the pharmacist application, the patient Bob allows the collection of all the tags information but with a numbering view depending on their EPC manufacturer field. In the two cases, the purpose should be declared along with the subscription. In more details, according to the privacy policy, if the ECSpec of the nurse application, only contains the type C medications as a *target*, and the specified duration is in accordance with the time constraints, the collection of the data is authorized with a valid *purpose*. If the Nurse requests for more EPC objects as *targets*, these latter are updated in accordance with the data owner preference, by changing the defined pattern for this field. For example, the pattern entered in the ECSpec : `urn:epc:pat:sgtin-96:0000389.*.*`, is replaced by `urn:epc:pat:sgtin-96:0000389.1234.*`, to specifically designate an object of type `1234`. The same update is done over the duration field. Note that the representation of the EPC objects is identically defined in both PrivOrBAC framework and ECSpec.

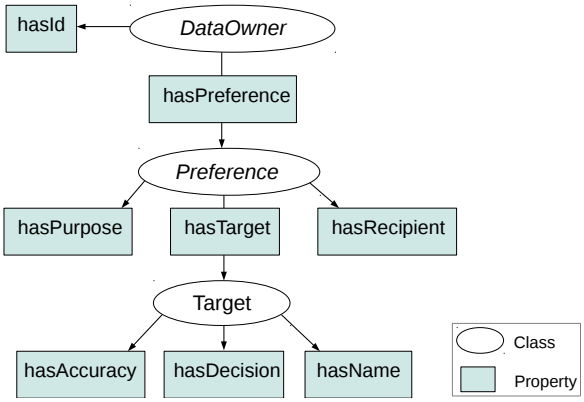

Fig. 7 Privacy preferences ontology

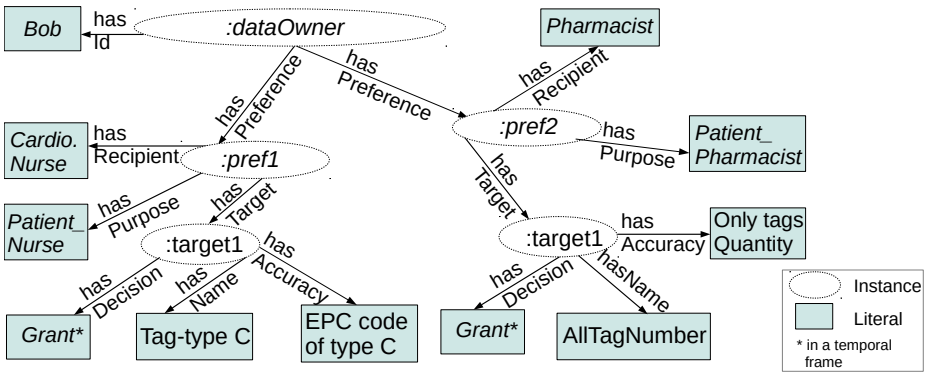

Fig. 8 Privacy preferences in the motivating scenario

Algorithm 1 Subscribe to an ECSpec

Input: ECSpec `spec`, purpose `purp`, recipient `URI`

```
1: NewSubscriber ← URI
2: if (PrivacyController.verify(spec, URI, purp)) then
3:    ECSpec spec' ← PrivacyController.update(spec)
4:    if spec' != Null then
5:       spec ← spec'
6:       SubscribersLIST.put(URI)
7:    else
8:       ALEMessage ← No update to be performed
9:    end if
10: else
11:    ALEMessage ← The URI has not the right to subscribe to ECSpec with the purpose purp
12: end if
```

5.3.2 Privacy control for subscriptions

The *PrivacyController* is defined as an independent entity that *CentralEntity* calls for new subscriptions. Our *PrivacyController* entity uses two main methods: *Verify* to check the subscription parameters, i.e., the purpose, the ECSpec name and the logical reader field and *Update* to change the ECSpec, when needed.

The *Verify* method returns a boolean:

- *False* when the entered parameters are not correct, e.g., the specified purpose and logical reader are not adequate for this access,
- *True* when the subscription to the ECSpec is entirely allowed (e.g., already existing in a defined database) or when it is allowed but with possibly, further filtering for privacy reasons, e.g., more EPC *targets* than allowed are specified. In this latter case, the *Update* method is called.

The *Update* method updates the ECSpec content with the corresponding data owner preferences. It compares the ECSpec fields entered by the recipient and the preferences of the data owner, then updates each field of the ECSpec, except the logical readers, which could be rather updated using the the logical reader API, cf. Section 3. At the end, the *Update* method checks for the correctness of the resulted ECSpec `spec'`. In the case `spec'` is not correct, the method outputs `Null`, meaning that no changes are performed on the ECSpec, and an empty report is generated.

Algorithm 1 shows the *subscribe* method that could be included in the *CentralEntity*. Steps 2 and 3 verify the existence of an already subscribed recipient, identified by its URI address with the same ECSpec. Step 5 verifies the permission of the recipient to subscribe to the report notifications with the entered ECSpec and purpose. In the positive case (Step 3), a call to the *Update* method is performed. If the output of the *Update* method is True (cf. Step 4), then the original ECSpec is required to be updated with a new filtered ECSpec. Step 6 adds the address URI to the list of validated subscribers in case of permission. Note that for checking the correctness, some general criteria should be satisfied by any updating algorithm, to generate sound, maximum and secure results [41]. For example, if the period of reading tags specified by the subscriber is different from the period specified by the data owner, the minimum of the two values is considered. Thus, the subscription is not rejected and the criteria of maximality is applied.

6 Privacy controller integration in Fosstrak

To implement our privacy-enhanced module, we use the open-source Fosstrak platform [17]. Fosstrak implements the standardized roles and interfaces published by EPCglobal. Fosstrak provides a generic middleware solution for large-scale deployment using a modular approach. It is also viewed as the most well addressed middleware solution among the research based middleware in terms of description of the solution and different interactions, which serves as a basis for other new implementations [20]. The Fosstrak platform consists of three separate modules: the reader module, the filtering and collection middleware module, and the EPCIS module, which deals with interpretation of the captured RFID data in an application context. Each of the three modules

implements the corresponding roles in the EPCglobal network, as well as the interface specifications of the reader protocol, the ALE (version 1.1) and the EPC Information System Query and Capture interfaces. In Fosstrak, the interfaces to request the ALE and readers are modeled in the WebServices Description Language[1] (WSDL). To communicate with RFID readers, the Fosstrak ALE middleware uses the EPCglobal LLRP [11] (Low Level Reader Protocol). For readers that do not support LLRP, the ALE middleware uses the Fosstrak Hardware Abstraction Layer (HAL).

We frame our work in the middleware module, the key location where the configuration of events to be collected is performed and related reports are generated.

6.1 Fosstrak *EventCycle* and *ReportsGenerators*

An *EventCycle* entity in Fosstrak is constructed according to an event cycle specification, ECSpec. Whenever a client defines a new event cycle (ECSpec) through the ALE interface, a new *ReportsGenerator* will be also created along an *EventCycle*. Upon creation, the *EventCycle* entity aquires the readers from the logical reader API. As soon as a client subscribes to the *EventCycle*, a *ReportsGenerator* entity starts the *EventCycle*. Then, tags are read and data are returned to the *ReportsGenerator*.

In point of fact, a capturing application does not subscribe to an *EventCycle* to receive its specified events, rather the subscription is done through the associated *ReportsGenerator*. The *ReportsGenerator* is the central entity ensuring that the *EventCycle* is started/stopped and that subscribers receive the resulting tags information. Therefore, the *ReportsGenerator* acts as a gateway for clients to the *EventCycle*.

6.2 Privacy controller implementation in Fosstrak

We have modified the Fosstrak filtering and collection server (*fc-server*) version 1.2.0, implemented in Java language, as well as the Web-based client to support the *purpose* attribute as an entry along with the ECSpec and the URI address. In the *fc-server*, we have mainly changed the *ReportsGenerator* implementation and all the classes related to it. More specifically, we have added the *PrivacyController* class and its methods into the Fosstrak filtering and collection server and changed the `subscribe` method of the *ReportsGenerator* class. We have also changed the *fc-common* directory of the Fosstrak middleware to integrate exception treatments in relation with the right to collect the data. To take into account these changes, a compilation of the server as a Web application ARchive (WAR) file is needed. For more details about the modified Fosstrak code, please refer to [38].

6.3 Testing scenario and results

To test the whole process of the filtering and collection of EPC events, all roles and interfaces directly related to the EPCglobal middleware (cf. Figure 4) should be well configured and connected. We consider the EPCglobal Low Level Reader Protocol [11] (LLRP) to ensure the communication between the F&C middleware and the different readers. Readers should also support the same protocol. We use the LLRPCommander tool [2] that helps configuring and managing LLRP-compliant RFID readers. To simulate LLRP readers and generate tag data, we consider the reader emulator Rifidi[3]. The ALE clients (i.e., capturing application) are simulated by the middleware Web client application, which comes with the Fosstrak platform.

Regarding the management of the data owners preferences, the PrivOrBAC policy is stored in a dedicated server and users preferences are accessed via a web service [31] using the REpresentational State Transfer (REST) architecture.

[1] http://www.w3.org/TR/wsdl

[2] http://code.google.com/p/fosstrak/wiki/LlrpMain

[3] http://www.transcends.co/community

As for the provisional rules, since collected data are not meant to be stored in the middleware level, we need at least to store a log of representative number of requests [1]. The aim is to search for previously performed actions in the history, to decide whether to activate or not a present action (e.g., a permission to read a particular information). In our approach, we do not store the requests that are initially sent by the capturing application, as proposed in the last version of Fosstrak. Rather, we store the rewritten requests to apply our privacy control on the effective launched requests.

6.3.1 Sequence diagram

In order to generate the event cycle reports, an ALE Client should subscribe to a set of events specified in its entered event cycle specification ECSpec. To visualize the new messages flow related to the subscription activity, Figure 9 depicts a sequence diagram including the privacy controller module we have integrated into the Fosstrak platform. We assume that the client and the recipient of the report represent the same entity. The goal is to receive the reports corresponding to the client specification, when the context holds with the rules defined in the privacy policy. In the sequel, we describe the flow of messages in the Fosstrak platform applied to the motivating scenario (cf. Section 5.2).

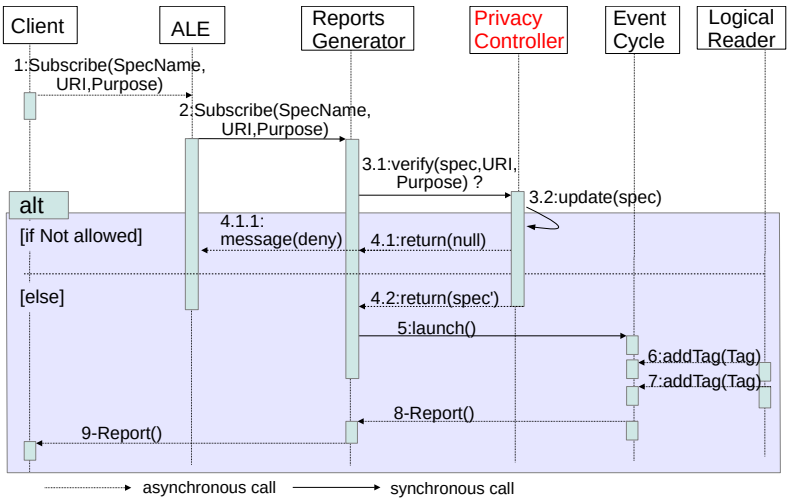

Fig. 9 Sequence diagram related to subscriptions

1: The Client subscribes to an *EventCycle*, by entering the ECSpec (assumed to be already created), the recipient address (URI), and the purpose (from a purpose list). Figures 12 and 13 in the Appendix, respectively illustrate the ECSpecs of the nurse and the pharmacist applications rendered into XML files. In the case of the nurse application, the EPC code partition `0000389` refers to the manager code for a given corporation, and the partition `000162` denotes the product code for hypertension medication (Type C), where `GTIN-96` tag encoding is used. The *reportSet* field is set to *DELETIONS*, to only detect deleted tags from the *shelf1* reader.

We assume that the nurse is only interested in the `epc` code, which matches the Type C tag, regardless of its serial number. In the second case, the pharmacist subscribes to an ECSpec considering only count view of tags grouped by their manager code. The subscription considers the current *DELETIONS* state of *shelf1*.
2: The ALE subscribes the Client to the *ReportsGenerator* that exclusively corresponds to the specified ECSpec (either for the nurse or the pharmacist applications).
3.1: The *ReportsGenerator* instantiates the *PrivacyController* to verify the content of the request depending on the privacy policy.
3.2: In the case, the purpose and the logical reader of the new ECSpec are authorized and an

updating process is handled to verify if the remaining fields of the ECSpec correspond to the privacy policy.
4.1/4.1.1: The *PrivacyController* checks the Client request. When the request does not meet the privacy policy (e.g., more than the Type C medication is specified for that URI Recipient), or if the updated ECSpec is not correct, a deny message is returned to the ALE interface stating that the privacy policy does not allow the present subscription.
4.2: The *PrivacyController* checks the request and possibly updates it. In the present case, the access is granted, and therefore, the *PrivacyController* allows the *ReportsGenerator* related to this event cycle to continue the processing.
5: The *ReportsGenerator* starts the *EventCycle*. The *EventCycle* now processes tags from the readers.
6/7: The LogicalReader(s) add(s) Tags to the *EventCycle*.
8/9: When the *EventCycle* reaches its boundaries, the reports are generated and sent back to the Client through the *ReportsGenerator*.

6.3.2 Obtained results

Regarding the nurse application, results (cf. Figure 14 in the Appendix) show the set of tags that are read by the *shelf1* reader. In the nurse specification (cf. Figure 12 in the Appendix), the nurse searched to collect either the EPC code *epc* in the `fieldname` attribute and a field named *age* in the memory content of the tag. This latter field has not been taken into account in the obtained report, as it is not specified in the data owner preferences. The report (cf. Figure 14) shows that only EPC tags of type C are collected and presented with their entire code. The `DELETION` set specifies that reports must include only the tags that are taken away from the shelf, compared to the last event cycle.

Concerning the pharmacy application, the outputs (cf. Figure 15 in the Appendix) show only the quantity of tags in the area of *shelf1*. This quantity is partitioned in a number of groups differentiated by the manager code. The last group shows the total tags that are in the frame of the filter specified in the pharmacist ECSpec (cf. Figure 13 in the Appendix). The `CURRENT` set of tags specified in the ECSpec shows the tags presented in the *shelf1*. In this case, the report is not updated since it corresponds to the preferences specified by the data owner.

7 Performance evaluation

We evaluate, in this work, the performance of our proposal in terms of execution time. The aim of our simulations is to measure the total time needed for the middleware to serve all the requests. For this, we vary the number of simultaneous subscriptions to the middleware from 1 to 50, then, we consider the maximum response time among all the launched requests. We also consider in this test that the number of patients managed by the PrivOrBAC server is equal to 100. We compare our results with the original Fosstrak middleware, i.e., without modifying the *ReportsGenerator* entity in the server and the corresponding Capturing application. Note that our simulations are performed on a PC with Ubuntu OS 64 bits, 2.4 GHz of CPU and 4.0 GB of RAM.

Three scenarios are considered: the first one is named Original scenario, where the privacy controller is disabled. The second one, named Verif. scenario, performs only a request verification to check the conformity of the declared purpose and the ECSpec fields with the data owner preferences (with a call to the PrivOrBAC entity). In this scenario, we assume that there is no need to update the ECSpec request. The third scenario, named Verif. & Update, performs both the verification and the update of the ECSpec request according to the privacy policy.

Figure 10 compares the total execution time of the three studied cases (i.e., Original, Verif., and Verif. & Update scenarios). As expected, we can notice that the total execution time needed in the privacy-enhanced middleware to handle all the requests is higher than the original case, with relevant differences when the number of simultaneous subscriptions (N_s, for short) is high. This increase is mainly due to the execution time when accessing the PrivOrBAC server, which grows linearly with N_s (cf. Figure 10). In addition, we can observe that the execution time in the

Verif. & Update scenario almost coincides with the execution time of the Verif. scenario when N_s is low (i.e., $N_s < 23$). The difference between the two scenarios becomes noticeable only when N_s is high. This means that the update time of the request in our privacy-enhanced middleware is prominent only at high load (i.e., when the number of simultaneous subscription and ECSpec updates becomes higher than 33).

Figure 11 further investigates the execution time of the three studied cases in the middleware without accounting for the time needed to access the PrivOrBAC server. We can see that the execution time increases with N_s for the three cases. In particular, for both privacy-enhanced scenarios, the execution time slightly increases compared to the original case when the number

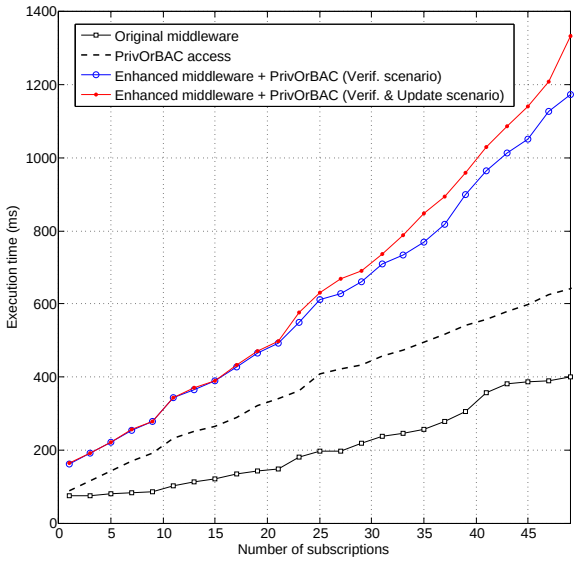

Fig. 10 Total execution time comparison with PrivOrBAC access (in Msec.)

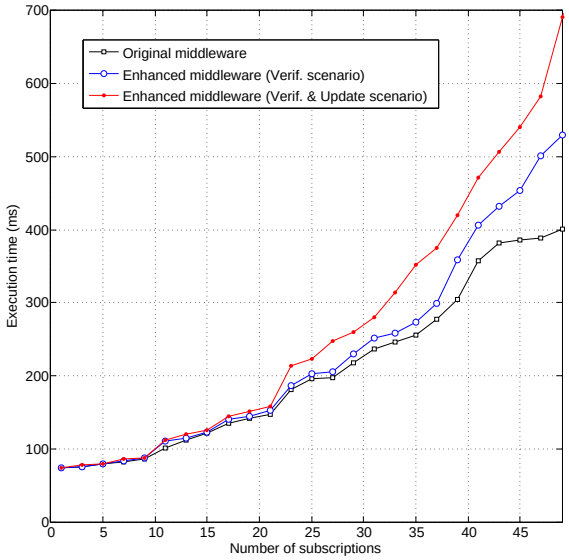

Fig. 11 Total execution time comparison without PrivOrBAC access (in Msec.)

of simultaneous subscriptions is low (i.e., $N_s < 23$). However, the gap becomes significant as N_s increases. Indeed, in the Verif. scenario (respectively, the Verif. & Update scenario), the execution time becomes clearly higher than the time for the original case when $N_s \geq 39$ (respectively, $N_s \geq 23$).

It is worth noting that the reported time measurements can be reduced when using a dedicated server with high computation capabilities, compared to the computer characteristics used for these simulations.

8 Conclusion

The human-centric deployment of RFID technology today poses several security and privacy concerns, which could adversely affect its wide adoption. We have shown the importance of introducing security solutions at the early stages of RFID data processing to prevent unintentional disclosures of sensitive information. In addition, it is undisputed that the act of reading out one or more RFID tags constitutes a data collection. This means that existing privacy regulations and laws also apply to the communication involving the RFID middleware, responsible for collecting data. In this paper, we first addressed some privacy concerns of the EPCglobal technology which is currently one of the predominant standardization efforts in the RFID community. Second, we provided a policy-driven approach to enforce privacy in such a technology with principles of declared purpose, accuracy and explicit consent. Also we have added the provisional context control to model security rules whose activation depends on the history of previously performed actions. Third, we provided a proof-of-concept prototype that shows the feasibility of our approach. Finally, we gave some performance evaluation in terms of execution time to compare our privacy enforcement solution to the Fosstrak original one. As future work, we aim to find a trade-off between the different correctness criteria of our updating algorithm, as in our approach the priority is given to the maximality criteria. Also, we intend to ensure an end-to-end privacy solution, to complete our approach, from the reader to the final back-end application.

Acknowledgements The authors would like to thank the anonymous reviewers and Dr. Langar for their valuable comments and suggestions to improve the quality of the paper. The authors also gratefully acknowledge the partial support received from the French FUI16 GINTAO project.

References

1. Common criteria for information technology security evaluation. https://www.niap-ccevs.org/Documents_and_Guidance/cc_docs/CCPART2V3.1R4.pdf (2012)
2. for Economic Co-operation & Development Council, O.: Recommendation of the Council Concerning Guidelines Governing the Protection of Privacy and Transborder Flows of Personal Data. OECD (1980)
3. Agrawal, R., Cheung, A., Kailing, K., Schonauer, S.: Towards traceability across sovereign, distributed RFID databases. In: 10th International Database Engineering and Applications Symposium, (IDEAS'06), pp. 174–184. IEEE (2006)
4. Ajam, N., Cuppens-Boulahia, N., Cuppens, F.: Contextual privacy management in extended role based access control model. Data Privacy Management and Autonomous Spontaneous Security pp. 121–135 (2010)
5. Ajana, M.E., Boulmalf, M., Harroud, H., Hamam, H.: A policy based event management middleware for implementing rfid applications. In: International Conference on Wireless and Mobile Computing, Networking and Communications, (WIMOB'09), pp. 406–410. IEEE (2009)
6. Architecture Review Committee: The EPCglobal Architecture Framework. Tech. rep., EPClobal (2010)
7. Chaudhry, M., Ahmad, Q., Sarwar, I., Akbar, A.H.: Comparative study of RFID middlewares-defining the roadmap to SOA-based middlewares. In: International Conference on Industrial Technology (ICIT'10), pp. 1386–1393. IEEE (2010)
8. Damiani, E., Vimercati, S., Jajodia, S., Paraboschi, S., Samarati, P.: Balancing confidentiality and efficiency in untrusted relational DBMSs. In: Proceedings of the ACM conference on Computer and communications security, pp. 93–102. ACM (2003)
9. EPCglobal.: EPC Radio-Frequency Identity Protocols Class-1 Generation-2 UHF RFID Protocol for Communications at 860-960 MHz. Tech. rep., Version 1.2.0, http://www.epcglobalinc.org/standards/ (2008)
10. EPCglobal. Inc: The Application-Level Events (ALE) Specification, version 1.1 - Part 1: Core Specification. Tech. rep., EPCGlobal (2008)
11. EPCGlobal Inc: Low Level Reader Protocol (LLRP). Tech. Rep. Version 1.1, EPCGlobal (2010)

12. EPCglobal. Inc: Public Policy. http://www.gs1.org/epcglobal/public_policy (2011)
13. EPCglobal Inc: The EPCglobal Website. http://www.gs1.org/epcglobal (2014)
14. of the European Communities, O.J. (ed.): Directive 95/46/EC of the European Parliament and of the Council on the protection of Individuals with regard to the processing of personal data and on the free movement of such data, no. 281 in 31. (1995)
15. Floerkemeier, C.: Integrating rfid readers in the enterprise it–overview of intra-organizational rfid system services and architectures. Academic publication of the Auto-ID Labs (2008)
16. Floerkemeier, C., Schneider, R., Langheinrich, M.: Scanning with a purpose–supporting the fair information principles in rfid protocols. In: Ubiquitous Computing Systems, pp. 214–231. Springer (2005)
17. Fosstrak: Project License. http://fosstrak.googlecode.com/svn-history/r2112/legacy_website/license.html (2009)
18. Grummt, E., Müller, M.: Fine-grained access control for epc information services. In: The Internet of Things, pp. 35–49. Springer (2008)
19. IBM Corp.: IBM WebSphere Premises Server. http://www-01.ibm.com/software/integration/sensor-events/ (2010)
20. INRIA: ASPIRE-Advanced Sensors and lightweight Programmable middleware for Innovative RFID Enterprise applications. www.fp7-aspire.eu/ (2009)
21. Ismael, A., Carlos, C., Jose, C., Rubén, H., Enrique, V.: Managing RFID Sensors Networks with a General Purpose RFID Middleware. Sensors **12**(6), 7719–7737 (2012)
22. Juels, A.: RFID security and privacy: a research survey. Journal of Selected Areas in Communications **24**, 381–394 (2006)
23. Kalam, A.A.E., Benferhat, S., Miège, A., Baida, R.E., Cuppens, F., Saurel, C., Balbiani, P., Deswarte, Y., Trouessin, G.: Organization based access control. In: POLICY. 4th IEEE International Workshop on Policies for Distributed Systems and Networks (2003)
24. Kartakis, S., Sakkalis, V., Tourlakis, P., Zacharioudakis, G., Stephanidis, C.: Enhancing Health Care Delivery through Ambient Intelligence Applications. Sensors **12**, 11,435–11,450 (2012)
25. Kerschbaum, F.: An access control model for mobile physical objects. In: Proceedings of the 15th ACM symposium on Access control models and technologies, pp. 193–202 (2010)
26. Kywe, S.M., Li, Y., Shi, J.: Attack and defense mechanisms of malicious epc event injection in epc discovery service. In: RFID-Technologies and Applications (RFID-TA), IEEE International Conference on, pp. 1–6 (2013)
27. Masoumzadeh, A., Joshi, J.: PuRBAC: Purpose-aware role-based access control. On the Move to Meaningful Internet Systems (OTM) pp. 1104–1121 (2008)
28. Motorola: RFID technology and EPC in retail. Tech. rep., Symbol Technologies (2004)
29. Ni, Q., Lin, D., Bertino, E., Lobo, J.: Privacy-Aware Role Based Access Control. In: 12th ACM symposium on Access control models and technologies, pp. 41–50. ACM (2007)
30. Oracle: Oracle Application Server Wireless. Tech. Rep. 10.1.2 (2005)
31. Oulmakhzoune, S., Cuppens-Boulahia, N., Cuppens, F., Morucci, S., Barhamgi, M., Benslimane, D.: Privacy query rewriting algorithm instrumented by a privacy-aware access control model. In: Annals of telecommunications (ANTE) (2013)
32. Prabhu, B., Su, X., Ramamurthy, H., Chu, C.C., Gadh, R.: WinRFID: A Middleware for the Enablement of Radiofrequency Identification (RFID)-Based Applications. Mobile, wireless, and sensor networks: Technology, applications, and future directions p. 313 (2006)
33. Sandhu, R.S., Coyne, E.J., Feinstein, H.L., Youman, C.E.: Role-based access control models. Computer **29**(2), 38–47 (1996)
34. Sarma, S., Brock, D.L., Ashton, K.: The networked physical world. Tech. Rep. White Paper MIT-AUTOID-WH-001, Auto-ID Center (2000)
35. Schapranow, M., Zeier, A., Plattner, H.: Security Extensions for Improving Data Security of Event Repositories in EPCglobal Networks. In: 9th International Conference on Embedded and Ubiquitous Computing (IFIP EUC'11), pp. 213–220. IEEE (2011)
36. Song, J., Kim, H.: The RFID middleware system supporting context-aware access control service. In: The 8th International Conference on Advanced Communication Technology, 2006. (ICACT'06), vol. 1. IEEE (2006)
37. Song, J., Kim, T., Lee, S., Kim, H.: Security enhanced RFID middleware system. World Academy of Science, Engineering and Technology **10** (2005)
38. Tounsi, W.: Security and Privacy Controls in RFID Systems Applied to EPCglobal Networks. Ph.D. thesis, Télécom Bretagne - Institut Mines-Telecom (2014)
39. Tounsi, W., Cuppens-Boulahia, N., Cuppens, F., Garcia-Alfaro, J.: Fine-grained privacy control for the rfid middleware of epcglobal networks. In: Proceedings of the Fifth International Conference on Management of Emergent Digital EcoSystems, MEDES '13, pp. 60–67. ACM (2013)
40. Tounsi, W., Cuppens-Boulahia, N., Garcia-Alfaro, J., Chevalier, Y., Cuppens, F.: KEDGEN2: A key establishment and derivation protocol for EPC Gen2 RFID systems. Journal of Network and Computer Applications **39**(0), 152 – 166 (2014)
41. Wang, Q., Yu, T., Li, N., Lobo, J., Bertino, E., Irwin, K., Byun, J.W.: On the correctness criteria of fine-grained access control in relational databases. In: Proceedings of the 33rd international conference on Very large data bases, pp. 555–566 (2007)
42. Westin, A.F.: Privacy and freedom. Washington and Lee Law Review **25**(1), 166 (1968)
43. Yang, N., Barringer, H., Zhang, N.: A purpose-based access control model. In: Third International Symposium on Information Assurance and Security (IAS), pp. 143–148. IEEE (2007)

Appendix: Specifications and results in XML Files

```
<?xml version="1.0" encoding="UTF-8"?>
<ale:ECSpec xmlns:ale="urn:epcglobal:ale:xsd:1" >
<logicalReaders>
        <logicalReader>shelf1</logicalReader>
</logicalReaders>
<boundarySpec>
<repeatPeriod unit="MS">48300000</repeatPeriod>
<duration unit="MS">3000</duration>
</boundarySpec>
<reportSpecs>
        <reportSpec reportName="report-cardio-nurse">
                <reportSet set="DELETION"/>
                <filterSpec>
                        <extension>
                                <filterList>
                                        <filter>
                                                <includeExclude>INCLUDE</includeExclude>
                                                <fieldspec>
                                                        <fieldname>epc</fieldname>
                                                </fieldspec>
                                                <patList>
                                                      <pat>urn:epc:pat:sgtin-96:3.0000389.000162.*</pat>
                                                </patList>
                                        </filter>
                                </filterList>
                        </extension>
                </filterSpec>
                <output includeTag="true">
                        <extension>
                                <fieldList>
                                        <field>
                                                <fieldspec>
                                                        <fieldname>age</fieldname>
                                                </fieldspec>
                                        <\field>
                                <fieldList>
                        </extension>
                </output>
        </reportSpec>
</reportSpecs>
</ale:ECSpec>
```

Fig. 12 ECSpec filtering to obtain Type C tags (nurse application)

```
<?xml version="1.0" encoding="UTF-8"?>
<ale:ECSpec xmlns:ale="urn:epcglobal:ale:xsd:1">
<logicalReaders>
        <logicalReader>shelf1</logicalReader>
</logicalReaders>
<boundarySpec>
<repeatPeriod unit="MS">302400000</repeatPeriod>
<duration unit="MS">3000</duration>
</boundarySpec>
<reportSpecs>
        <reportSpec reportName="report-pharmacist">
                <reportSet set="CURRENT"/>
                <groupSpec>
                        <pattern>urn:epc:pat:sgtin96:3.X.X.*.*</pattern>
                </groupSpec>
                <output includeCount="true"/>
        </reportSpec>
</reportSpecs>
</ale:ECSpec>
```

Fig. 13 ECSpec filtering to obtain tags numbers (pharmacist application)

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ale:ECReports totalMilliseconds="3000" terminationCondition="DURATION" specName="spec-nurse" date="2013-04-
03T17:31:27.913+02:00" ALEID="ETHZ-ALE-750360204"  xmlns:ale="urn:epcglobal:ale:xsd:1" >
        <reports>
                <report reportName="report-cardio-nurse">
                        <group>
                                <groupList>
                                        <member><tag>urn:epc:pat:sgtin-96:3.0000389.000162.1000</tag></member>
                                        <member><tag>urn:epc:pat:sgtin-96:3.0000389.000162.1001</tag></member>
                                </groupList>
                        </group>
                </report>
        </reports>
</ale:ECReports>
```

Fig. 14 ECReports for the cardiology (nurse) application

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ale:ECReports totalMilliseconds="3000" terminationCondition="DURATION" specName="spec-pharmacist"
date="2013-04-03T18:31:01.913+02:00" ALEID="ETHZ-ALE-750360204" xmlns:ale="urn:epcglobal:ale:xsd:1" >
<ale:ECReports xmlns:ale="urn:epcglobal:ale:xsd:1" >
        <reports>
                <report reportName="report-pharmacist">
                        <group name="urn:epc:pat:sgtin-96:3.0000389.*.*">
                                <groupCount><count>50</count></groupCount>
                        </group>
                        <group name="urn:epc:pat:sgtin-96:3.0000456.*.*">
                                <groupCount><count>4</count></groupCount>
                        </group>
                        <group>
                                <groupCount><count>60</count></groupCount>
                        </group>
                </report>
        </reports>
</ale:ECReports>
```

Fig. 15 ECReports for the pharmacist application