Fine-Grained Privacy Control for the RFID Middleware of EPCglobal Networks

Wiem Tounsi, Nora Cuppens-Boulahia, Frédéric Cuppens, Joaquin Garcia-Alfaro*
Institut Mines-Telecom, TELECOM Bretagne, France
*Institut Mines-Telecom, TELECOM SudParis, France
{*Name.Surname*}*@telecom-bretagne.eu ; joaquin.garcia_alfaro@telecom-sudparis.eu*

ABSTRACT

The Electronic Product Code (EPC) is a Radio Frequency IDentification (RFID) that offers a new way of automating identification. However, once RFID tags carry more than just an identifier, privacy may be violated. Treating the privacy in early stages helps to master the data view before interpreting 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, aggregating and grouping 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. We propose a privacy controller module that enhances the Filtering and Collection middleware of the RFID EPCglobal network. We provide a privacy policy-driven model using some enhanced contextual concepts of the extended Role Based Access Control model. To show the feasibility of our privacy-enhanced model, we provide a proof-of-concept prototype integrated into the middleware of the Fosstrak framework, an open-source implementation of the EPCglobal specifications.

Categories and Subject Descriptors

K.6.m [**Management of computing and Information Systems**]: Miscellaneous—*Security*; D.2.11 [**Software Engineering**]: Software Architectures; K.4.1 [**Public Policy Issues**]: Privacy

General Terms

Access Control, Privacy Assurance, Standardization

Keywords

RFID, Middleware, EPCglobal, Privacy policy

1. INTRODUCTION

Radio Frequency IDentification (RFID) technology offers a new way of automating the identification, the tracking and the storing of information contained on RFID tags. These tags can be attached or embedded in the item to be identified and are read when they enter a readers 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 [11]. In this vein, the Auto-ID Center [20] created the concept of a unique code called Electronic Product Code (EPC). With this technology, critical errors such as prescribing wrong medicines can be avoided, thanks to better monitoring operations. However, once tags carry more than just an identifier, i.e, a bearer's age or illnesses, bearer's privacy may be violated. RFID systems consist typically on tags, readers, and backend applications such as middleware and database to manage and store the collected data. Here, we are interested in the backend side, particularly the middleware component. The 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. The system, at this point, is likely to suffer from parameter manipulation and eavesdropping leading to privacy concerns. In this article, we use the properties of filtering and aggregation in the middleware for privacy goals, without interfering with existing standards.

The definition of not sharing information is a fundamental aspect of privacy [24], however, the privacy of people focuses today on who has access to what information, rather than what information is collected. A major portion of previous work on privacy has been aligned on anonymizing user information or on preventing personal information from disclosure e.g, by encryption means [3]. These latter works treat some issues of privacy but do not completely handle the different situations where the data owner choose to share information with others. In addition, dealing with encrypted data in the middleware level makes expensive the query process, since not all the collected data are sensitive, rather, the aggregation of some information may need to be protected. In this article, we name as data owner the user who specifies a set of privacy preferences. The preferences depend on the context and privacy dimensions supported by the system.

Numerous reasons motivate the support of privacy issues in the RFID middleware. First, treating the privacy in the middleware helps to master the reading configuration and the data view before interpreting and storing it 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 "GTIN". To treat the applications requests, we propose to handle the privacy policy before the events are generated, since this minimizes, as soon as possible, the risk of unauthorized disclosures. Second, filtering data for privacy issues at the middleware stage has the advantage to relax the event collection in this stage, leading to an appropriate adaptation of the queries executed by the reader over the air interface. This allows readers to use efficiently the limited bandwidth, e.g., target only a particular tag population or switch off

MEDES'13 October 28-31, 2013, Neumünster Abbey, Luxembourg

completely some readers. Finally, as RFID communication protocols support dedicated privacy enhancing features [22], the RFID middleware will also need to support their use.

Our contribution in this paper can be summarized as follows. First, we address some privacy concerns of the EPCglobal technology which is currently one of the predominant standardization efforts of the RFID community. Second, we propose a fined control over the privacy controller module [21] that we integrate into the Filtering and Collection middleware. The approach we apply in the EPCglobal middleware is policy-driven. It ensures the following features: (i) enforcing a privacy policy without interfering with the standard middleware interface, (ii) using an existing privacy-aware model[1] to store and manage privacy policy preferences, and (iii) taking into account the declared purpose, the accuracy and the consent dimensions as privacy requirements. To show the feasibility of our approach, we provide a proof-of-concept prototype that we apply on the open-source software for track and trace Fosstrak [7]. Fosstrak is a recommended implementation of the EPCglobal network interfaces.

Paper organization. Section II discusses related work. Section III describes some background about the EPCglobal middleware interfaces. Section IV details the privacy dimensions that can be enforced in the middleware. Section V introduces our privacy model and specification. Section VI describes our privacy enhancement solution applied on Fosstrak. Section VII concludes the paper.

2. RELATED WORK

Most widely deployed middleware products do not fully consider the security requirements when processing sensitive data. Several implementations from software vendors [8, 16] and specialized companies [18] 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 in accordance with regulations, expected to be done at the middleware level [15]. A possible reason for this issue is that most enterprises run the RFID middleware in an internal network.

Most existent implementations support EPCglobal network. Some of them explicitly include security solutions in the EPCIS (EPC Information Service) level [8, 16, 18] relying mostly on role based access control and encryption models. Although some of these aforementioned products include security enhancements into their middleware implementations [8, 16], few authors worked on privacy problems. An exception is the work in [9], 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 to the used privacy policy nor information about the plug-in implementation.

Table 1 summarizes some middleware security features of several platforms. It shows the version of the EPCglobal middleware interface (ALE) implemented in each tool and the proposed security and privacy approaches at this level.

Table 1: Middleware privacy in related work

	ALE version	Middl. Security	Middl. Privacy
IBM Websphere RFID [8]	1.1	Role based access control	-
Oracle sensor edge server [16]	1.0	Role Based user management	-
Fosstrak [7]	1.1	Authentication for ALE access	-
CHUK [13]	1.0	-	-
Aspire [9]	1.1	ALE Access Control API	-
WinRFID [18]	1.1	-	-

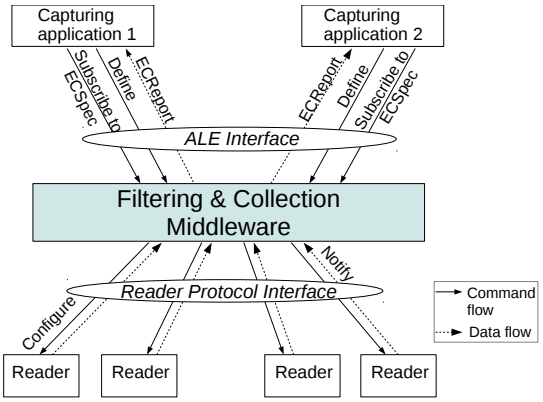

Figure 1: EPCglobal middleware

It is worth noting that the ALE 1.1 specification introduces several enhancements of the ALE 1.0, particularly a new API for controlling access to other ALE APIs.

So far, all the mentioned products propose a role based access control as a security approach in the middleware. However, they do not present a privacy control integrated solution. For example, the authors do not cite the notion of declared purpose to be applied to the definition of new *ECSpec*s. This notion is used in our proposition to extend the classical access control model by handling privacy requirements in accordance with known regulations.

3. THE EPCGLOBAL MIDDLEWARE AND ITS INTERFACE

The EPCglobal network, initially proposed by the MITs Auto-ID Center [20] 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 F&C (Filtering & Collection) middleware is one of the main components of the EPCglobal network architecture [2]. 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 Interface (ALE) (cf. Figure 1). 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. In the sequel, we present the information that an EPC tag carries. Then we give, in more detail, the role played by the Reading API.

3.1 Structure of the Electronic Product Code

The Electronic Product Code is a numbering scheme that provides a unique identification for physical objects, assemblies and systems. It serves as a reference to Internet-based information. There are many situations where the EPC stored in an RFID tag are considered as a sensitive information e.g., in contexts where product flows constitute valuable business intelligence. The EPC code elements are shown hereafter to determine the private data carried by the tag.

The EPC is mostly a 96-bit code divided into four fixed partitions, cf. Figure 2. First, the 8-bit header defines the number, type and length of subsequent data partitions. Second, the EPC manager defines the manufacturer of the item. The next 24-bits defines the object type code. Finally, the

36-bits unique object defines the identification number.

Since, the EPC was not made to replace but to integrate existing numbering schemes, different headers can be used. For example, the well known Global Trade Identification Number (GTIN) that is widely used in the retail industry is integrated into EPC technology. A special version of GTIN (i.e., UPC Version B) originally intended to handle the National Drug Code and National Health Related Item Codes.

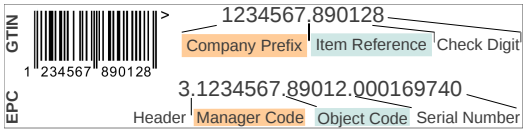

Figure 2: EPC composition and GTIN integration

Since this type of EPC carries personal information, it may present some privacy concerns when read without the consent of the holder, i.e., the patient. This unauthorized reading can be handled in the middleware ALE interface. For this aim, let us see the ALE main actions and data types.

3.2 Application Level Events Interface

One or more clients can request the ALE interface [4] via a set of methods. Each request causes the ALE engine to take an action and return synchronously a result. The ALE interface also enables clients to subscribe to events that are delivered asynchronously, by using a Uniform Resource Identifier. This URI describes the client address to which the information is delivered. e.g., *subscribe(ECSpec, URI)* is the method to subscribe to the ALE for an already defined *ECSpec*.

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 can result in 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 included in the list of reports. Its main fields are:

- include/exclude patterns (*ECFilterSpec*) 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 (*ECGroupSpec*) 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 (*ECReportSetSpec*) 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 (*ECReportOutputSpec*) 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

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 [19] (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). Nevertheless, it does not cope with the details of data aggregation and the use of filtered and combined reports within a request specification. We believe that a fine-grained security on these collected data for privacy concerns has to be handled to prevent applications, even in the same organization, to collect data that undermine people's private lives or reputations.

4. INTRODUCING PRIVACY DIMENSIONS

The information handled by the EPC code is used to identify assets of an individual or an organization. It could also be useful to know the illness of a person if the EPC identifies a drug. Even by hiding some parts, e.g., by encryption solutions, a partial part of the EPC code can be useful. For example, the combination of an EPC manager identifier and the object class is usually enough to determine the exact kind of object the tag belongs to. Thus, to treat the privacy issues for accurate results, the solution should let all the EPC code to be readable or to entirely filter it. Also, using the grouping pattern of the ECSpec specification can be a solution to prevent non-authorized disclosures. Before delving into the details of our proposal, let us present the privacy dimensions that can be handled in the middleware level then cross them with each field of the ECSpec to extract the fields we have to treat to enforce privacy.

4.1 Privacy into 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 [6]. These privacy recommendations are defined in accordance with relevant privacy and data protection laws and regulations [15]. Based on these directives, we define the main privacy dimensions 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

Table 2: Privacy Dimensions in the Middleware

Dimension	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 or consent of the data owner.
(6) Security mechanisms	to protect personal data from unauthorized access or disclosure (e.g., encryption, anonymity)

of these dimensions can be expanded in the middleware level while others are dropped to suit its specifications. These latter could be rather enforced in upper layers data processing. We summarize in Table 2 the privacy requirements that can be achieved in the F&C middleware. Here, we are interested in the explicit consent, purpose and accuracy dimensions.

4.2 Privacy Dimensions in ECSpec

Table 3 shows the privacy dimensions that can be treated in each ECSpec field. The purpose of the data request is better specified apart from the ECSpec (cf. Section 6.1).

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 to provide a localization service (e.g., applied to cases of smart 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 the regular expressions [4] or by only reporting a set of tags grouped by an EPC code field. In the sequel, we present the privacy policy model and specification of our solution.

5. 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 [1, 12, 14, 25]. The security policy is generally specified according to an access control model to simplify the upgrade of existing information systems. Here,

Table 3: ECSpec fields and privacy threats

EC-Spec Field	Privacy threats		Privacy Dimension
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 fileld 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	

we detail the dimensions of the PrivOrBAC model [1], which extends the Organization-Based Access Control model [10] (OrBAC) by reusing most of its implemented existing mechanisms, e.g., managing the security policy by specifying the contexts as complex conditions to define fine grained privacy control requirements. The choice of PrivOrBAC relies on its capacity to handle most of the privacy dimensions appearing in the guidelines and recommendations previously mentioned, e.g., dimensions of consent, accuracy, purposes of the data collection; but of course it can be substituted with another privacy model without disrupting the whole process.

5.1 Privacy Contextual Management

PrivOrBAC models the explicit consent as a context, the purpose as a user declared context and the different view levels as an accuracy of the objects. Regarding the data owner privacy preferences, they are included into the contexts of the security policy of the organization.

The Consent Dimension: 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 *NeedConsent* (a Boolean parameter whose value is true when the 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 subject can access.

The Purpose Dimension 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 Dimension 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 [1]. 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 ac-

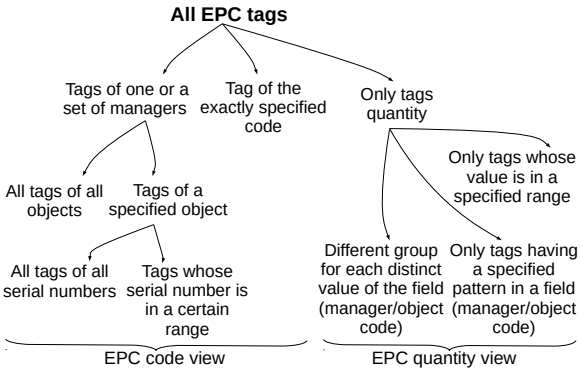

Figure 3: Accuracy levels of EPC tags data

curacies. 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*. Figure 3 shows the accuracy-based object hierarchy related to EPC data. 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 policy and access control for each view.

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.

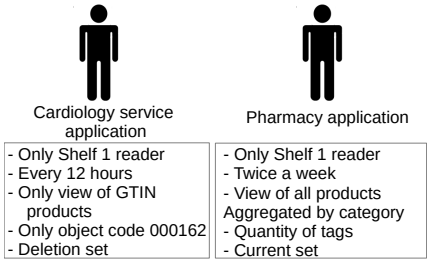

Figure 4: Scenario of monitoring applications

The elderly patient Bob is remotely monitored at home from different services of the Hospital (cf. Figure 4). 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.

Based on these assumptions, the nurse of cardiology service is only interested in objects of type C, i.e., the control is done over the field object of the EPC code (cf. Figure 2), 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 interested in tags quantity depending on their manager and object codes.

5.3 OrBAC Specification

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. For example, 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.

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, 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, AllTag Number, 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.*

The roles nurse and pharmacist are managed by the access control API of the ALE middleware interface.

6. PRIVACY CONTROLLER INTEGRATION INTO FOSSTRAK

To implement our solution, we use the open-source plateform Fosstrak [7]. Fosstrak provides an EPCglobal-certified EPCIS (EPC Information Service) Repository. It consists of three separate modules: the reader module, the filtering&collection middleware module, and the EPCIS module. These modules are implemented with the corresponding EPCglobal interfaces specifications e.g, the ALE interface. Here, we focus on the middleware module, the central point where begins the specification of events to be collected and reported.

Relying on the ALE interactions (cf. Section 3) and the privacy dimensions required at this level (cf. Section 4), we consider that the subscription to an ECSpec is the action concerned with the privacy issues i.e., since the defined ECSpec is not executed until the subscription method is performed, the collection of data is only triggered by this action. In the sequel, we consider the subscription request to test our Privacy Controller module. Let us first present the Privacy Controller activities when triggered by a subscription request, then detail our testing scenario.

6.1 Privacy Controller Activities & Algorithms

To subscribe to an event cycle and receive related reports, the clients subscribe to an entity called the ReportsGenerator [7] via the ALE interface. The ReportsGenerator is the entity responsible for ensuring that the event cycle is started/stopped and that subscribers receive the resulting reports. In addition, we define the PrivacyController as the entity handling the compliance of the requests with the predefined privacy policy.

6.1.1 Privacy Controller Activities

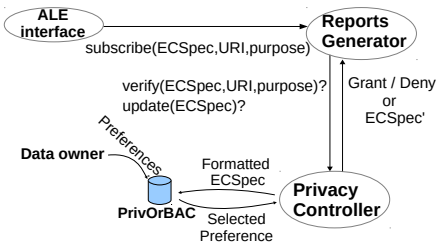

Figure 5: Privacy controller activities

Figure 5 depicts the relation between the two defined entities (i.e., ReportsGenerator and PrivacyController). The ReportsGenerator 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 access to the specified data or updates the ECSpec for accurate results.

Note that the purpose attribute, can either be added to the ECSpec XML content or to the *subscribe* request sent to the ALE middleware. We add this attribute to the *subscribe* request for the following reasons. First, this way, the ECSpec can be used by many ALE Clients and with possibly different purposes. Second, adding the purpose parameter to the request avoids distorting the ECSpec and leaves it consistent with the standard EPCglobal specification.

Regarding privacy preferences, the data owner specifies them using the PrivOrBAC model [1] depending on the privacy dimensions enforced in the system. Each privacy *preference* is associated with a *recipient* address and a *purpose* on which the preference is defined. This *preference* turns around a set of *target*s 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, and in the positive case, in which accuracy, the data will be disclosed.

Algorithm 1 Verifiy the subscription

Input: ECSpec `spec`, name `spName`, purpose `purp`, recipient `URI`
Output: Boolean

```
1: if URI.purposeList.contains(purp) then
2:   if Spec.consentList.contains(spName) then
3:     return True
4:   else
5:     if URI.allowedLR.contains(spec.logicalReader) then
6:       Verification of the entered spec Targets...
7:     else
8:       Msg←Not allowed to use logicalReader for this action.
9:     end if
10:  end if
11: end if
```

Algorithm 2 Subscribe to ECSpec

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

```
1: NewSubscriber ← URI
2: if (Subscribers.containsKey(URI)) then
3:   ALEExceptionMessage ← "The URI is already subscribed
     to this specification spec"
4: else
5:   if (PrivacyController.verify(spec, URI, purp)) then
6:     ECSpec spec' ← PrivacyController.update(spec)
7:     if spec' != Null then
8:       spec ← spec'
9:       Subscribers.put(URI)
10:    else
11:      ALEMessage ← The update is incorrect
12:    end if
13:  else
14:    ALEMessage ← The URI has not the right to subscribe
       to ECspec with the purpose purp
15:  end if
16: end if
```

6.1.2 Privacy Control Algorithms for Subscriptions

The PrivacyController is defined in the Fosstrak code as an independent class that ReportsGenerator calls for new subscriptions. Our PrivacyController entity uses two main methods: *Verify* for the verification of the subscription parameters, i.e., the purpose, the ECSpec name and the logical reader field and *Update* for the update of the ECSpec. The *Verify* method returns a boolean: *False* when the entered parameters are not coherent, e.g., the specified purpose and logical reader are not adequate for this access and *True* when the ECSpec is entirely allowed or if it is allowed but with a further filtering for privacy reasons. In this latter case, the *Update* method is called. This method updates the ECSpec with the corresponding data owner preferences (i.e., compares the ECSpec entered by the recipient and the preferences of the data owner, then updates each field of the ECSpec, excepting the logical readers). At the end, the *Update* method checks for the correctness of the resulted ECSpec. In the case it is not correct, the method outputs `Null`.

Algorithm 1 shows a snapshot of the *Verify* method. Step 1 verifies the purpose entered with the ECSpec in a list of allowed purposes for the specified recipient. Step 2 checks the name of the ECSpec. If it is a new *spName* (Step 4), it checks for the logical reader field in the given ECSpec. Step 8 shows the case of a not allowed logical reader specified in the ECSpec. In this case, no possible update is to be done and the subscription is refused.

Algorithm 2 shows the *subscribe* method included in the modified Fosstrak ReportsGenerator class. 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 reports notification with the entered ECSpec and purpose relying on the defined privacy policy. In the positive case (Step 6), a call to the *Update* method is performed. If the output of the *update* method is not `Null`, then the original ECspec is updated with a new filtered ECspec. Step 8 adds the address

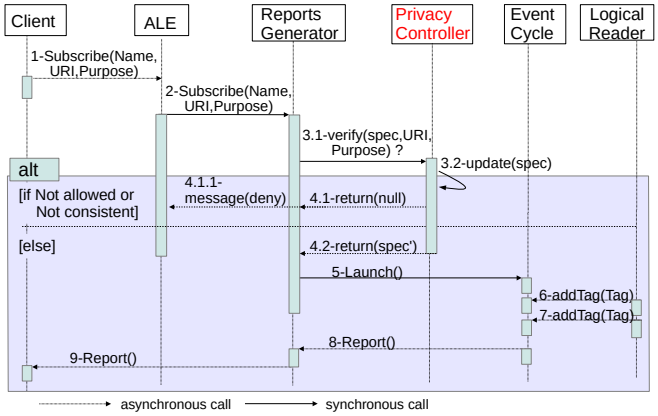

Figure 6: Sequence diagram related to subscriptions

URI to the list of validated subscribers in case of permission. The check for correctness avoids for example obtaining empty reports. Note that for checking the correctness, some criteria should be satisfied. In [23], the authors proposed three correctness criteria that should be satisfied by any query processing algorithm: (i) the *Soundness* criteria, when the rewritten ECSpec returns only correct answers, i.e., answers to the initial query. (ii) the *Maximality* criteria, when the rewritten ECSpec returns as much information as possible and, (iii) the *Security* criteria, if the result of ECSpec respects the security and privacy policy of the queried system.

As an example, if the period of tags detection specified by the requester is different from the period specified by the data owner, the minimum of the two values is considered. Thus, no empty reports are generated and the criteria of maximality is applied.

6.2 Testing Scenario and Results

To test the subscription scenario, all roles and interfaces shown in Figure 1 have to be well configured, and connected. We first give the implementation details that lead to our results. Then, we describe the messages flow related to the subscription activity applied on the F&C middleware.

Implementation details: Our privacy module written in Java, is added to the version 1.2.0 of the Fosstrak middleware. To connect the F&C middleware to different readers, we consider the EPCglobal Low Level Reader Protocol [5] (LLRP) as a reader protocol via the LLRPCommander tool. LLRPCommander provides a number of features to help configuring and managing LLRP-compliant RFID readers. To simulate LLRP readers and generate tag data, we use the reader emulator Rifidi[1]. The ALE Client (i.e., capturing application) we use is the F&C Web Client application which comes with the Fosstrak platform. Finally, the PrivOrBAC policy is stored in a dedicated server and user's preferences are accessed via a web service [17], using the REpresentational State Transfer (REST) architecture.

Sequence diagram and generated reports: Figure 6 depicts a sequence diagram visualizing the messages flow related to the subscription activity with the new 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.
1: The Client subscribes to an EventCycle, by entering the ECSpec, the recipient address (URI) and the purpose (from a purpose list). Figure 7 illustrates a snapshot of an ECSpec rendered into XML. In this example, the code is formatted in EPC tag URI: `0000389` refers to the manager code for

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

```
<ns2:ECSpec>
 <logicalReaders><logicalReader>Shelf1</logicalReader><\l
...
  <reportSpecs>
   <reportSpec reportName="report-nurse">
    <reportSet set="DELETIONS"/>
     <filterSpec>
       <filterList>
        <filter>
         <includeExclude>INCLUDE</includeExclude>
         <fieldspec><fieldname>epc</fieldname></fieldspec>
         <patList>
          <pat>urn:epc:pat:sgtin-96:3.0000389.000162.*</pa
         </patList>
        </filter>
       <filterList>
     <filterSpec>
     <output includeTag="true"/>
    <reportSpec>
   </reportSpecs>
...
 </ns2:ECSpec>
```

Figure 7: ECSpec Filtering to obtain Type C tags

```
<ns2:ECReports>
...
 <report reportName="report-nurse">
  <group>
   <groupList>
    <member>
     <tag>urn:epc:pat:sgtin-96:3.0000389.000162.1000</tag>
    </member>
   </groupList>
  </group>
 </report>
</ns2:ECReports>
```

Figure 8: ECReports for Cardiolology App.

```
<ns2:ECSpec>
 <logicalReaders><logicalReader>Shelf1</logicalReader><\
 ...
  <reportSpec reportName="report-pharmacist">
   <reportSet set="CURRENT"/>
   <groupSpec>
    <pattern>urn:epc:pat:sgtin-96:X.X.*.*</pattern>
   </groupSpec>
   <output includeCount="true"/>
  <reportSpec>
...
</ns2:ECSpec>
```

```
<ns2:ECReports>
  ...
 <report reportName="report-pharm">
  <group name="urn:epc:pat:sgtin-96:3.0000389.*.*">
   <groupCount><count>56</count></groupCount>
  </group>
  <group name="urn:epc:pat:sgtin-96:3.0039500.*.*">
   <groupCount><count>20</count></groupCount>
  </group>
</ns2:ECReports>
```

Figure 9: ECSpec/ECReports for Pharmacy App.

a given corporation, and `000162` denotes the product code for hypertension medication, where `GTIN-96` tag encodings are 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.
2: The ALE subscribes the Client to the ReportsGenerator that exclusively corresponds to this ECSpec.
3.1: The ReportsGenerator instantiates the PrivacyController to verify the request depending on the privacy policy.
3.2: In the case the purpose and the logical reader of the ECSpec are authorized, an update process verifies if the remaining fields of the ECSpec corrrespond 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 update it. In the present case, the access is granted, and therefore, the PrivacyController allows the ReportsGenerator related to that 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. Figure 8 shows a snapshot of the resulted ECReport for the nurse subscription, presented in XML, reporting the set of tags deleted from the previous event cycle.

Finally, Figure 9 shows the difference between reports views. It depicts a snapshot of the ECSpec and ECReports related to the pharmacy application. The pharmacist has to specify the quantity of tags, for each different manager code.

7. CONCLUSION

In this paper, we have shown the importance of introducing privacy solutions at an early stage of RFID data processing to prevent unintentional disclosures of sensitive information. The goal is to ensure that the communication line that extends from readers to middleware and enterprise applications responds to privacy requirements. First, we 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 dimensions of declared purpose, accuracy and explicit consent. Third, we provided a proof-of-concept prototype that shows the feasibility of our approach. Currently, we are measuring the execution time needed for our verification and updating approach to compare it with the standard case. This allows us to bring some insight on optimizing our privacy controller. In addition, we intend 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. Finally, when several middleware events go to one aggregation entity, a privacy-preserving problem could be raised, even if the privacy is handled separately in each middleware. As future work, we aim to find a trade off between data privacy preservation and secure data aggregation, in such centralized architectures.

8. REFERENCES

[1] N. Ajam, N. Cuppens-Boulahia, and F. Cuppens. Contextual privacy management in extended role based access control model. *Data Privacy Management and Autonomous Spontaneous Security*, pages 121–135, 2010.

[2] Architecture Review Committee. The EPCglobal Architecture Framework. Technical report, EPCGlobal, 2010.

[3] E. Damiani, S. Vimercati, S. Jajodia, S. Paraboschi, and P. Samarati. Balancing confidentiality and efficiency in untrusted relational DBMSs. In *Proceedings of the ACM conference on Computer and communications security*, pages 93–102. ACM, 2003.

[4] EPCglobal. Inc. The Application-Level Events (ALE) Specification, version 1.1 - Part 1: Core Specification. Technical report, EPCGlobal, 2008.

[5] EPCGlobal Inc. Low Level Reader Protocol (LLRP). Technical Report Version 1.1, EPCGlobal, 2010.

[6] EPCglobal. Inc. Public Policy, 2011.

[7] Fosstrak. Project License, 2009.

[8] IBM Corp. IBM WebSphere Premises Server, 2010.

[9] INRIA. ASPIRE-Advanced Sensors and lightweight Programmable middleware for Innovative RFID Enterprise applications, 2009.

[10] A. A. E. Kalam, S. Benferhat, A. Miège, R. E. Baida, F. Cuppens, C. Saurel, P. Balbiani, Y. Deswarte, and G. Trouessin. Organization based access control. In *POLICY*. 4th IEEE International Workshop on Policies for Distributed Systems and Networks, 2003.

[11] S. Kartakis, V. Sakkalis, P. Tourlakis, G. Zacharioudakis, and C. Stephanidis. Enhancing Health Care Delivery through Ambient Intelligence Applications. *Sensors*, 12:11435–11450, 2012.

[12] A. Masoumzadeh and J. Joshi. PuRBAC: Purpose-aware role-based access control. *On the Move to Meaningful Internet Systems (OTM)*, pages 1104–1121, 2008.

[13] Mobitec of CUHK IE department. CUHK RFID System 1.0. Technical report, Mobitec, 2007.

[14] Q. Ni, D. Lin, E. Bertino, and J. Lobo. Privacy-Aware Role Based Access Control. In *12th ACM symposium on Access control models and technologies*, pages 41–50. ACM, 2007.

[15] O. J. of the European Communities, editor. *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*, number 281 in 31, 1995.

[16] Oracle. Oracle Application Server Wireless. Technical Report 10.1.2, 2005.

[17] S. Oulmakhzoune, N. Cuppens-Boulahia, F. Cuppens, S. Morucci, M. Barhamgi, and D. Benslimane. Privacy query rewriting algorithm instrumented by a privacy-aware access control model. In *Annals of telecommunications (ANTE)*, 2013.

[18] B. Prabhu, X. Su, H. Ramamurthy, C.-C. Chu, and R. Gadh. WinRFID: A Middleware for the Enablement of Radiofrequency Identification (RFID)-Based Applications. *Mobile, wireless, and sensor networks: Technology, applications, and future directions*, page 313, 2006.

[19] R. S. Sandhu, E. J. Coyne, H. L. Feinstein, and C. E. Youman. Role-based access control models. *Computer*, 29(2):38–47, 1996.

[20] S. Sarma, D. L. Brock, and K. Ashton. The networked physical world. Technical Report White Paper MIT-AUTOID-WH-001, Auto-ID Center, 2000.

[21] W. Tounsi, N. Cuppens-Boulahia, F. Cuppens, and J. Garcia-Alfaro. Privacy-enhanced filtering and collection middleware in epcglobal networks. In *8th International Conference on Risks and Security of Internet and Systems (CRISIS)*, 2013.

[22] W. Tounsi, N. Cuppens-Boulahia, J. Garcia-Alfaro, Y. Chevalier, and F. Cuppens. KEDGEN2: A key establishment and derivation protocol for EPC Gen2 RFID systems. *Journal of Network and Computer Applications (In Press)*, 2013.

[23] Q. Wang, T. Yu, N. Li, J. Lobo, E. Bertino, K. Irwin, and J.-W. Byun. On the correctness criteria of fine-grained access control in relational databases. In *Proceedings of the 33rd international conference on Very large data bases*, pages 555–566, 2007.

[24] A. F. Westin. Privacy and freedom. *Washington and Lee Law Review*, 25(1):166, 1968.

[25] N. Yang, H. Barringer, and N. Zhang. A purpose-based access control model. In *Third International Symposium on Information Assurance and Security (IAS)*, pages 143–148. IEEE, 2007.