Modelling Contexts in the Or-BAC Model

Frédéric Cuppens[1] Alexandre Miège[2,3]
[1]GET/ENST Bretagne, BP 78, 2 rue de la Châtaigneraie, 35512 Cesson Sévigné Cedex, France
Email: cuppens@enst-bretagne.fr
[2]GET/ENST, 46 rue Barrault, 75634 Paris Cedex 13, France
Email: alexandre.miege@enst-bretagne.fr
[3]ONERA/DTIM Centre de Toulouse, BP4025, 31055 Toulouse Cedex 4, France

Abstract

As computer infrastructures become more complex, security models must provide means to handle more flexible and dynamic requirements. In the Organization Based Access Control (Or-BAC) model, it is possible to express such requirements using the notion of context. In Or-BAC, each privilege (permission or obligation or prohibition) only applies in a given context. A context is viewed as an extra condition that must be satisfied to activate a given privilege. In this paper, we present a taxonomy of different types of context and investigate the data the information system must manage in order to deal with these different contexts. We then explain how to model them in the Or-BAC model.

1. Introduction

Current computer infrastructure are becoming more and more complex and difficult to manage securely. It does no longer correspond to monolithic architecture but instead must manage a set of virtual communities that want to interoperate and share resources. A virtual community is a composition of heterogeneous and independently designed and managed sub-organizations. When setting up security policies of new communities, it is necessary to identify participant roles. Thus, a security policy model like Role Based Access Control (RBAC) model [21] provides concepts that are useful in this area.

However, such security policies must also be adapted to deal with new requirements; rules in these policies are no longer static but dynamic, depending on the context. They must be also self-adaptive with respect to temporal conditions, user's location, user's previous behavior, etc. Different organizations involved in a virtual community must be able to express their own policies. As a consequence, a suited security policy model must be able to manage these different policies within a single framework. Classical access control models [3, 13, 21] are not sufficiently flexible to specify such context-dependant requirements.

In this paper, we show how the Organization Based Access Control (Or-BAC) model [17] is useful to deal with some of these new requirements. In Or-BAC, the access control policy does not directly apply to subject, action and object. Instead, it defines permissions (or obligations or prohibitions) that applies within an *organization* to control the *activities* performed by *roles* on *views*. For instance, the policy might specify that role *physician* has permission to perform activity *consult* on view *medical record*. This fits to specify static permissions. However, the Or-BAC model also allows the administrators to specify more complex dynamic permissions since one can consider that each permission only applies in some given *contexts*.

Our objective is to further investigate this notion of context. To activate a given access control rule, the subject, the action and the object must separately satisfy some conditions. In the Or-BAC model, these conditions are that the subject must be assigned to a given role, the object must be used in a given view and the action must partake in some activity. Besides these conditions, there are extra conditions that must be satisfied to activate an access control rule. These extra conditions may be related to very different notions, such as temporal or spatial requirements. We call context, such extra conditions. In the following, we first investigate what kind of information a given information system must manage to provide means to deal with contextual conditions. Based on this analysis, we present several types of context – temporal, spatial, prerequisite, user-declared and provisional contexts – and explain how to model them in the Or-BAC model. As far as we know, this is the first time such different contexts are expressed within a unique access control model.

The remainder of this paper is organized as following. Section 2 recalls the Or-BAC model. Section 3 presents a taxonomy of different types of context and how to model them in the Or-BAC model. Section 4 provides a comparison with related work. Finally, section 5 concludes the paper.

2. Or-BAC model

2.1. Preamble

The final objective of an access control policy is to specify the privileges (*permissions*, *obligations* or *prohibitions*) that control the *actions* performed by *subjects* on *objects*. Using a logical notation, this might be represented by a set of facts having the following forms: $Is\_permitted(s, \alpha, o)$, $Is\_obliged(s, \alpha, o)$ or $Is\_prohibited(s, \alpha, o)$. These facts mean that a given subject s is permitted (resp. obliged or prohibited) to perform a given action α on a given object o. For instance, the fact $Is\_permitted(John, read, Paul\_medical\_record)$ specifies that John is permitted to read Paul's medical record.

However, enumerating all these facts is a quite fastidious and difficult to manage task. In particular, each time a new subject, or a new object, or a new action is created, it is necessary to explicitly insert new facts specifying the privileges associated with this new subject, object or action.

To simplify management of access control policy, rule based language have been proposed [16, 8, 12]. Using a rule based language, an access control policy is represented by a set of rules having the following forms:

- $\forall s, \forall \alpha, \forall o, (Condition \rightarrow Is\_permitted(s, \alpha, o))$
- $\forall s, \forall \alpha, \forall o, (Condition \rightarrow Is\_obliged(s, \alpha, o))$
- $\forall s, \forall \alpha, \forall o, (Condition \rightarrow Is\_prohibited(s, \alpha, o))$

meaning that, for every subject s, action α and object o, if a given condition is satisfied[1], then subject s is permitted (resp. obliged or prohibited) to perform action α on object o.

Let us now further analyze the structure of $Condition$ in the above rules. We suggest structuring $Condition$ as following:

$cond\_subject(s) \wedge cond\_action(\alpha) \wedge cond\_object(o) \wedge constraint(s, \alpha, o)$

where $cond\_subject(s)$, $cond\_action(\alpha)$ and $cond\_object(o)$ are respectively the conditions the subject s, the action α and the object o must separately satisfy so that the corresponding rule applies. $constraint(s, \alpha, o)$ is an additional condition that joins subject s, action α and object o. Satisfying the constraint is necessary to activate the rule.

For instance, let us consider the rule "a physician is permitted to consult his or her patient's medical record". In this case, $cond\_subject(s)$, $cond\_action(\alpha)$ and $cond\_object(o)$ respectively correspond to the conditions that s is a physician, α is an action of consulting and o is a medical record. $constraint(s, \alpha, o)$ is a condition that joins subject s and object o (in this example, action α is absent from the constraint), namely o must be a record of s's patient.

We shall now present Or-BAC and explain how to model $cond\_subject(s)$, $cond\_action(\alpha)$, $cond\_object(o)$ and $constraint(s, \alpha, o)$. For this purpose, we need first to present basic entities of the Or-BAC model.

2.2. Basic concepts of Or-BAC

The central entity in Or-BAC is the entity $Organization$. Roughly speaking, an organization can be seen as an organized group of subjects, playing some role or other. Notice that a group of subjects does not necessarily correspond to an organization. More precisely, the fact that each subject plays a role in the organization corresponds to some agreement between the subjects to form an organization.

In Or-BAC, subject, action and object are respectively abstracted into role, activity and view. A view corresponds to a set of objects that satisfy a common property. Similarly, an activity regroups actions that partake of the same principles. Finally, privileges only apply in specific $context$. Examples of context may be *Night*, *Working-Hours* or *Urgency* (see section 3 for further details about the context definition).

Therefore, in Or-BAC, there are eight basic sets of entities: Org (a set of organizations), S (a set of subjects), A (a set of actions), O (a set of objects), $\mathcal{R}$ (a set of roles), $\mathcal{A}$ (a set of activities), $\mathcal{V}$ (a set of views) and $\mathcal{C}$ (a set of contexts).

We assume that $Org \subseteq S$ (that is any organization is a subject) and that $S \subseteq O$ (that is any subject is an object).

Any entities in the Or-BAC model may have some attributes. This is represented by functions that associate the entities with the value of these attributes. For instance, if s is a subject, then $name(s)$ represents the name of s, $address(s)$ its address, etc.

2.3. Modelling the organization components

In Or-BAC, $cond\_subject(s)$, $cond\_object(o)$ and $cond\_action(\alpha)$ respectively correspond to conditions specifying that, in a given organization, a subject is empowered in a role, an object is used into a view and an action falls within an activity. This is represented by the following relationships:

- $Empower$ is a relation over domains $Org \times S \times \mathcal{R}$.

If org is an organization, s a subject and r a role, then $Empower(org, s, r)$ means that org empowers subject s in role r. Unlike the TMAC model [22] or the RBAC model [21] which consider binary relations between organizations and subjects or between subjects and roles, notice that our model consider a ternary relation between organizations, subjects and roles. This is useful to model situations where a given subject plays several roles but in different organizations.

- Use is a relation over domains $Org \times O \times \mathcal{V}$.

[1] Of course, this condition changes from one rule to another.

If org is an organization, o is an object and v is a view, then $Use(org, o, v)$ means that org uses object o in view v. This ternary relation is useful to characterize organizations that give different definitions to the same view. For instance, take the case of the view "medical record" defined in Purpan hospital as a set of Word documents and defined in Rangueil hospital as a set of tuples in a relational database.

- $Consider$ is a relation over domains $Org \times A \times \mathcal{A}$.

If org is an organization, α is an action and a is an activity, then $Consider(org, \alpha, a)$ means that org considers that action α falls within the activity a. Since $Consider$ is a ternary relation, different organizations may decide that the same action comes under distinct activities or that different actions come under the same activity. For instance, activity "consulting" corresponds, in Purpan hospital, to an action "read" that can be ran on data files whereas it corresponds, in Rangueil hospital, to action "select" that can be performed on relational databases.

2.4. Context definition

In section 2.1, we introduce the predicate $Constraint(s, \alpha, o)$ to model extra conditions a subject, an action and an object must satisfy to activate an access control rule. In Or-BAC, these extra conditions are modelled through the notion of *context*. Each context has a name and its definition depends on the organization. Notice that we use the term "context" in a broad sense since it actually corresponds to any constraint that joins a subject, an action and an object[2]. For instance, in the health care domain, the entity $Context$ will cover circumstances such as "urgency", "medical research", "attending physician", etc. Entities $Organization$, $Subject$, $Object$, $Action$ and $Context$ are linked together by the relationship $Define$:

- $Define$ is a relation over domains $Org \times S \times A \times O \times \mathcal{C}$.

If org is an organization, s a subject, α an action, o an object and c a context, then $Define(org, s, \alpha, o, c)$ means that within organization org, context c holds between subject s, action α and object o.

The conditions required for a given context to be linked, within a given organization, to subjects, objects and actions will be formally specified by logical rules. For instance, context *Attending_physician* may be defined as follows:

- $\forall s \in S, \forall \alpha \in A, \forall o \in O,$
  $(Define(H1, s, \alpha, o, Attending\_physician)$
  $\leftrightarrow name(o) \in patient(s))$

that is, in hospital $H1$, the context "attending physician" holds between subject s, action α and object o if and only if o is a record corresponding to a patient of subject s.

We also consider conjunctive, disjunctive and negative contexts. For this purpose, we introduce functions $\&$, $\oplus$ and $\bar{}$. If c_1 and c_2 are two contexts, then $\&(c_1, c_2)$ is a conjunctive contexts, $\oplus(c_1, c_2)$ is a disjunctive context and $\overline{c_1}$ is a negative context. We shall use the infix notations $c_1 \& c_2$ and $c_1 \oplus c_2$ in place of the prefix notations $\&(c_1, c_2)$ and $\oplus(c_1, c_2)$. These composed contexts are defined by the following rules:

- $\forall org \in Org, \forall s \in S, \forall \alpha \in A, \forall o \in O, \forall c_1 \in \mathcal{C}, \forall c_2 \in \mathcal{C},$
  $(Define(org, s, \alpha, o, c_1 \& c_2) \leftrightarrow$
  $Define(org, s, \alpha, o, c_1) \wedge Define(org, s, \alpha, o, c_2))$
- $\forall org \in Org, \forall s \in S, \forall \alpha \in A, \forall o \in O, \forall c_1 \in \mathcal{C}, \forall c_2 \in \mathcal{C},$
  $(Define(org, s, \alpha, o, c_1 \oplus c_2) \leftrightarrow$
  $Define(org, s, \alpha, o, c_1) \vee Define(org, s, \alpha, o, c_2))$
- $\forall org \in Org, \forall s \in S, \forall \alpha \in A, \forall o \in O, \forall c \in \mathcal{C},$
  $(Define(org, s, \alpha, o, \overline{c}) \leftrightarrow \neg Define(org, s, \alpha, o, c))$

Modelling various types of context will be further analyzed in section 3.

2.5. Policy definition

Using the materials presented in the previous sections, the $Condition$ expression introduced in section 2.1 corresponds in the Or-BAC model to formulas having the following form:

$$Empower(org, s, r) \wedge Use(org, o, v) \wedge$$
$$Consider(org, \alpha, a) \wedge Define(org, s, \alpha, o, c)$$

where s, o and α are variables corresponding respectively to a subject, an object and an action and org, r, v, a and c are constants corresponding respectively to an organization, a role, a view, an activity and a concept. For instance, the rule "in hospital H, a physician is permitted to consult his or her patient's medical record" may be represented by a rule having the following form:

- $\forall s \in S, \forall \alpha \in A, \forall o \in O,$
  $(Condition \rightarrow Is\_permitted(s, \alpha, o))$

  where, $Condition$ is the following formula:

  $Empower(H, s, Physician) \wedge$
  $Use(H, o, Medical\_record) \wedge$
  $Consider(H, \alpha, Consult) \wedge$
  $Define(H, s, \alpha, o, Attending\_physician)$

However, this is not exactly the way an access control policy is specified in the Or-BAC model. In Or-BAC, we go one step further by considering that the access control policy does not directly apply to subject, action and object. Instead, the access control policy is specified using the relationship $Permission$, $Obligation$ and $Prohibition$ defined as follows:

- $Permission$, $Obligation$ and $Prohibition$ are relations over domains $Org \times \mathcal{R} \times \mathcal{A} \times \mathcal{V} \times \mathcal{C}$

[2] Constraints do not apply systematically to all the parameters. For instance, we may have constraints that simply join subject and object or even constraints that are simply related to subject.

If org is an organization, r a role, a an activity, v a view and c a context then $Permission(org, r, a, v, c)$ (resp. $Obligation(org, r, a, v, c)$ or $Prohibition(org, r, a, v, c)$) means that in organization org role r is granted permission (resp. obligation or prohibition) to perform activity a on view v within context c.

The relationships $Permission$, $Obligation$ and $Prohibition$ enable a given organization to specify permissions, obligations and prohibitions between roles, activities and views in a given context. Now, triples that are instances of the relationship $Is\_permitted$ are logically derived from permissions granted to roles, views and activities by the relationship $Permission$. This is modelled by the following general rule[3]:

- $\forall org \in Org, \forall s \in S, \forall o \in O, \forall \alpha \in A, \forall r \in \mathcal{R}, \forall v \in \mathcal{V}, \forall a \in \mathcal{A}, \forall c \in \mathcal{C},$
  $Permission(org, r, a, v, c) \wedge$
  $Empower(org, s, r) \wedge Use(org, o, v) \wedge$
  $Consider(org, \alpha, a) \wedge$
  $Define(org, s, \alpha, o, c)$
  $\rightarrow Is\_permitted(s, \alpha, o)$

that is, if organization org, within context c, grants role r permission to perform activity a on view v, if org empowers subject s in role r, if org uses object o in view v, if org considers that action α falls within the activity a and if, within org, the context c holds between s, α and o then s is permitted to perform α on o.

2.6. Summary

There are several ways to specify an access control policy. A first way consists in explicitly enumerating a set of facts specifying the permissions, obligations and prohibitions of subjects to perform actions on objects. However, this approach is very difficult to manage. A second way is to specify a set of rules specifying the conditions that must be satisfied to derive that subjects are permitted, obliged or prohibited to perform actions on objects. However, specifying this set of rules is a complex and error prone task.

The Or-BAC model suggests an intermediary approach. The access control policy is specified by enumerating a set of facts corresponding to privileges. However, these privileges do not concern subject, action and object but role, activity and view. It is no longer necessary to specify a set of rules. There are simply three general rules that apply in every circumstance to derive what subjects are permitted, obliged or prohibited to perform actions on objects.

Notice that in [17], it is suggested to define hierarchies over roles but also organization, activity and view, and to associate permission inheritance with these different hierarchies. This is modelled as follows:

- Sub_role, is a relation over domains $Org \times \mathcal{R} \times \mathcal{R}$.

  If org is an organization, and r_1 and r_2 are roles, then $Sub\_role(org, r_1, r_2)$ means that, in organization org, role r_1 is a sub-role (also called senior role) of role r_2.
- Sub_view and $Sub\_activity$ are similarly defined as relations over domains $Org \times \mathcal{V} \times \mathcal{V}$ and $Org \times \mathcal{A} \times \mathcal{A}$.
- $Sub\_organization$ is a relation over domains $Org \times Org$.

3. Context in Or-BAC

3.1. The different contexts

As we have just seen, we use contexts to express different types of extra conditions or constraints that control activation of rules expressed in the access control policy. In this section, we investigate the following contexts (see figure 1) and show how we model them in Or-BAC:

- The *Temporal context* that depends on the time at which the subject is requesting for an access to the system,
- the *Spatial context* that depends on the subject location,
- the *User-declared context* that depends on the subject objective (or purpose),
- the *Prerequisite context* that depends on characteristics that join the subject, the action and the object.
- the *Provisional context* that depends on previous actions the subject has performed in the system.

We also assume that each organization manages some information system that stores and manages different types of information. To control context activation, each information system must provide the information required to check that conditions associated with the context definition are satisfied or not. The following list gives the kind of information related to the contexts we have just mentioned:

- A *global clock* to check the temporal context,
- the *subject environment* and the *software and hardware architecture* to check the spatial context,
- the *subject purpose* to check the user-declared context,
- the *system database* to check the prerequisite context,
- an *history* of the action carried out, to check the provisional context.

Figure 1 presents the correspondence between the contexts and the required data. If the information system does not provide some information in this list, then the corresponding context cannot be managed by the access control policy.

[3] Two similar general rules exist to derive instances of $Is\_Obliged$ and $Is\_prohibited$ from relationships $Obligation$ and $Prohibition$.

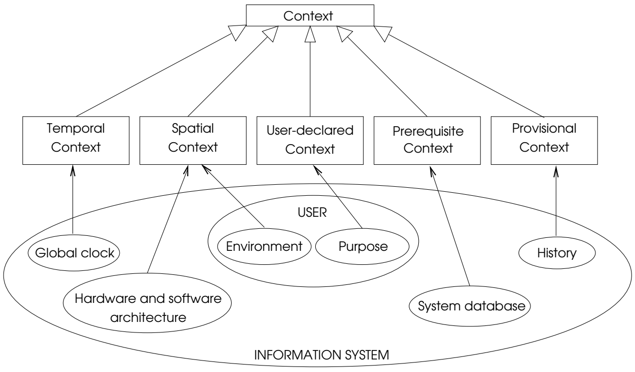

Figure 1. Context taxonomy and required data

3.2. Temporal context

Principle. With temporal contexts, it should be possible to express that a given action made by a given user on a given object is authorized only at a given time or during a given time interval. The temporal conditions can correspond to a day of the week, or to a time of the day, etc. For instance, a physician within an hospital may be allowed to access the medical record server only during the working hours, that is between 8:00AM and 19:00PM for example.

To validate a given query for an access, it is necessary to be able to evaluate the current time. Thus, we assume that the information system has a clock, and that this clock can be queried at any time to assess the temporal context of the query. We consider the entity clock as an object called $GLOBAL\_CLOCK$.

We associate the following attributes to $GLOBAL\_CLOCK$: $time$, day, $week$, $mounth$, $year$. The corresponding functions give the current time, the current day, the current week, etc.

Basic temporal contexts. We define two functions $before\_time$ and $after\_time$ that applies to the set of $Time$ and return a temporal context defined as follows:

- $\forall org \in Org, \forall s \in S, \forall \alpha \in A, \forall o \in O, \forall t \in Time,$ $(Define(org, s, \alpha, o, after\_time(t))$ $\leftrightarrow time(GLOBAL\_CLOCK) \geq t)$
- $\forall org \in Org, \forall s \in S, \forall \alpha \in A, \forall o \in O, \forall t \in Time,$ $(Define(org, s, \alpha, o, before\_time(t))$ $\leftrightarrow time(GLOBAL\_CLOCK) \leq t)$

We can similarly define two functions $before\_date$ and $after\_date$ that apply to the set of $Date$ and return a temporal context. We also consider a function on_day that applies to the set of Day and return a temporal context defined as follows:

- $\forall org \in Org, \forall s \in S, \forall \alpha \in A, \forall o \in O, \forall d \in Day,$ $(Define(org, s, \alpha, o, on\_day(d))$ $\leftrightarrow day(GLOBAL\_CLOCK) = d)$

Composed temporal context. Using the basic temporal contexts, we can define more complex temporal contexts, for instance:

- $night = after\_time(23:00) \oplus before\_time(8:00)$
- $weekend = on\_day(saturday) \oplus on\_day(sunday)$
- $working\_hours = after\_time(08:00)$ & $before\_time(19:00)$ & $\overline{weekend}$

Notice that context definition actually depends on the organization. For instance, in an organization where employees works on Saturday but not on Monday, context $working\_hours$ would be defined as follows:

- $working\_hours =$ $after\_time(08:00)$ & $before\_time(19:00)$ & $\overline{on\_day(sunday)}$ & $\overline{on\_day(monday)}$

Examples of permissions using temporal context. Let us consider the following rule: "In hospital $H1$, a physician is permitted to consult the medical record data base $MRDB$ during working hours". This is expressed by the following fact:

- $Permission(H1, physician, consult, MRDB,$ $working\_hours)$

Let us now assume that the role $cardiologist$ is also permitted to consult $MRDB$ on Sunday. This is expressed by the following fact:

- $Permission(H1, cardiologist, consult, MRDB,$ $working\_hours \oplus on\_day(sunday))$

If we assume that $cardiologist$ is a senior role of $physician$, it would be actually sufficient to specify that $cardiologist$ is permitted to consult $MRDB$ on Sunday, since the permission in temporal context $working\_hours$ will be inherited from $physician$.

3.3. Spatial context

Principle. Knowing the location from where the user makes the request can be useful to specify the access control policy. For example an hospital manager may be granted the right to read all employees' payrolls. But he must read those payrolls in his own office, and not anywhere else in the company. It thus reduces the possibility of curious employees being able to read their colleague's payrolls over the manager's shoulder. Spatial context is used to express this kind of condition.

We can distinguish two different types of spatial context. The physical spatial context and the logical spatial context. The first one corresponds to the physical location of the user, namely his office, a security area, a specific building, the country, etc. The logical spatial context corresponds to the "logical location" he stands in. For example, it can be the computer, the network or the sub-network, the cell in the case of radio communication such as in UMTS, etc.

In some cases, physical and logical spatial contexts are highly correlated. The network IP address from which a user is connected probably corresponds to a specific physical place such as a department area. Note that due to the expanding use of Global Position System (GPS) tools, it could be possible to locate a user or a terminal independently of the network.

If an organization allows its employee the use of Mobile IP, it is necessary to take into account from which network a request is emitted. A user will probably get reduced permissions if he is connected from a customer's office. Moreover the development of wireless technologies such as Wi-Fi motivates this work. The security policy must make it possible to take into account the fact that a user is connected through a wire network, a wireless network, or on which Wi-Fi access point he is attached.

Example of spatial contexts. Consider a company C that has a secured area SA in which specific security requirements are enforced. For example, there is no possibility of optical eavesdropping [19]. Users are allowed to consult certain documents on their laptop only in this area. If a given subnetwork address is allocated to this area, then the IP address of the terminal that is making a request is enough to locate it. Thus SA corresponds to the name of a specific subnetwork. We consider that the subject entity has the attribute $host\_IP$ which provides the IP address of the terminal on which a user is connected. The attribute IP_range is allocated to networks and subnetworks. In this example the context $in\_secured\_area$ can be defined as follows:

- $\forall s \in S, \forall \alpha \in A, \forall o \in O,$
  $Define(C, s, \alpha, o, in\_secured\_area)$
  $\leftrightarrow host\_IP(s) \in IP\_range(SA)$

A similar idea can be used in the case of Mobile IP. In this case, the local agent must manage the network where the mobile hosts are.

Let's consider another example. In a wireless network, some user is allowed to access a specific ressource from everywhere but only with his own laptop. The attribute boolean $host\_MAC$ is allocated to the entity user that indicates if the MAC address of the the packet recieved is really the MAC address of the user's laptop. The context on_own_laptop is then defined as follows:

- $\forall s \in S, \forall \alpha \in A, \forall o \in O,$
  $Define(C, s, a, o, on\_own\_laptop) \leftrightarrow host\_MAC(s)$

Notice that specific security mechanisms must be implemented to prevent a malicious user to bypass access control requirements by forging his own packets, and choosing the appropriate IP address or MAC address. However, we shall not further discuss these security issues in this paper.

3.4. User-declared context

Principle. In some circumstances, a subject according to the role he plays in the organization may be allowed to declare that he performs some activities in a given context. When declaring a context, a subject will obtain some specific permissions and possibly also some obligations or prohibitions. For instance, a subject playing the role medical researcher may be permitted to declare that he or she is performing an epidemiological analysis. By doing so, this subject will be permitted to have an access to some statistical database.

In our approach, user declared contexts are modelled as follows. We shall consider a view called $Purpose$. Objects belonging to the view $Purpose$ have an attribute $recipient$. If p is an object belonging to the view $Purpose$, then $recipient(p)$ represents the subject who takes advantage of the declared purpose. Notice that it is possible to consider sub-views of view $Purpose$ that may be associated with other specific attributes (see below for examples). user- The access control policy can specify that some roles are permitted to insert some objects in the view $Purpose$. Of course, the policy can also specify that the inserted objects must satisfy constraints, for instance conditions related to the attributes of the objects. This is useful to specify that a medical researcher is permitted to declare the epidemiological analysis purpose but not another purpose.

By inserting an object in the view $Purpose$, a subject will declare that another subject will perform some activity in a given context. Notice that in our model, there are two subjects involved in the process of context declaration: the subject who is declaring the context and the subject who takes advantage of this declaration. The policy can specify that these two subjects must be identical. For instance, in the above example, the medical researcher may be only permitted to declare a context that applies to himself or herself. However, it is also possible that the policy specifies that these two subjects may be different, provided that these subjects satisfy some constraints. For instance, a physician may be permitted to declare that his or her nurse will perform some activity in some given context. In this case, the subjects are different but the declarant must be a physician and the recipient must be the physician's nurse.

The definition of a user-declared context has three parts:

1. Definition of the context associated with objects belonging to the view $Purpose$
2. Specification of roles who are permitted to declare some given purpose.
3. Specification of roles that are permitted to perform some activities in the associated user-declared context.

Notice that activation of a user-declared context is often associated with provisional obligation (see section 3.6).

Example of user-declared context. Let us illustrate these three steps though the following example: In a given hospital $H1$, a user playing the role "medical researcher" is permitted to declare the context "epidemiological analysis". In this context, this user is permitted to have an access to some statistical database.

We consider a sub-view $Medical\_research$ of view $Purpose$. View $Medical\_research$ has two attributes: $recipient$ (inherited from view $Purpose$) and $topic$. If p is an object belonging to view $Medical\_research$, then $topic(p)$ represents the topic associated with this medical research (for instance, epidemiology, hemophilia, cancer, etc.).

First step: We define a context $Epidemic\_analysis$. This context is associated with objects belonging to sub-view $Medical\_research$ having topic equal to $epidemiology$. This is represented by the following rule:

- $\forall s \in S, \forall \alpha \in A, \forall o \in O,$
  $Define(H1, s, \alpha, o, Epidemic\_analysis) \leftrightarrow$
  $\exists p, (use(H1, p, Medical\_research) \wedge$
  $(recipient(p) = s) \wedge$
  $(topic(p) = epidemiology))$

that is, in hospital $H1$, subject s performs action α on object o in context $Epidemic\_analysis$ if there is an object p belonging to view $Medical\_research$ having s as a recipient and $epidemiology$ as a topic.

Second step: We specify that subjects playing role $Medical\_researcher$ is permitted to declare the purpose $Medical\_research$ that applies to themselves:

- $Permission(H1, Medical\_researcher, declare,$
  $Medical\_research, My\_purpose)$

In this $Permission$, $declare$ is an activity and $My\_purpose$ is a context defined as follows:

- $\forall s \in S, \forall \alpha \in A, \forall o \in O,$
  $Define(H1, s, \alpha, o, My\_purpose) \leftrightarrow$
  $\exists p, (use(H1, p, Purpose) \wedge (recipient(p) = s))$

that is a subject s is in context $My\_purpose$ if there is a purpose p having s as a recipient.

Third step: We specify that subjects playing role $Medical\_researcher$ are permitted to consult objects belonging to view $Statistic\_database$ in context $Epidemic\_analysis$:

- $Permission(H1, Medical\_researcher, consult,$
  $Statistic\_database, Epidemic\_analysis)$

Urgency as a user-declared context. Managing urgency is an important requirement of medical applications. However, it is a complex problem to characterize the conditions that, when satisfied, activate the urgency context. Actually, we argue that it would not be possible to give an exhaustive specification of such conditions. And even though this would be possible, it would be quite difficult for the information system to automatically check that one of these conditions is satisfied because most of them actually depend on the physician's judgement.

This is why we suggest modelling urgency as a user-declared context. Thus, we first define a sub-view $Urgent\_consultation$ of view $Purpose$. By inserting an object in this view, a physician is allowed to declare that he or she is consulting a given patient in urgency. View $Urgent\_consultation$ has two attributes: $recipient$ (inherited from view $Purpose$) and $declared\_patient$. If p is an object belonging to view $Urgent\_consultation$, then $declared\_patient(p)$ provides the name of the patient admitted in urgency.

Using view $Urgent\_consultation$, the associated context $Urgency$ is defined as follows:

- $\forall s \in S, \forall \alpha \in A, \forall o \in O,$
  $Define(H1, s, \alpha, o, Urgency) \leftrightarrow$
  $\exists p, (use(H1, p, Urgent\_consultation) \wedge$
  $(recipient(p) = s) \wedge$
  $(declared\_patient(p) = name(o)))$

that is, in the organization $H1$, subject s performs action α on object o in the context $Urgency$ if this subject declared a purpose of $Urgent\_consultation$ and o is an object related to the patient admitted in urgency.

We then assume that physicians are permitted to declare that they are consulting a patient in urgency:

- $Permission(H1, Physician, declare,$
  $Urgent\_consultation, My\_purpose)$

Finally, we have to specify the permission that applies to a physician in a context of $Urgency$. For instance, we can consider that a physician is permitted to consult the patient's medical record (even though this is not the attending physician of this patient):

- $Permission(H1, Physician, consult,$
  $Medical\_record, Urgency)$

3.5. Prerequisite context

Principle. In many cases, a permission (or an obligation or a prohibition) is granted to a subject, but only if some specific constraints are satisfied. For instance, let us turn back to the example presented in section 2.5. This example says that a physician is permitted to consult the patient's medical record. However, a specific constraint must be satisfied, that is this record corresponds to the physician's patient.

We assume that the information required to check this constraint, namely the set of patients attended by each physician, is stored into the *system database*. Thus, the evaluation of such a constraint is done by querying the database. This kind of constraints are called *prerequisite context.*

Example of prerequisite context. Let us consider the following example: "A nurse is granted the permission to consult a medical record in the context where the physician of the corresponding patient is absent". We consider the following function, $status$, that indicates the user's status (for instance $absent$).

The prerequisite context $absent\_physician$ is expressed as follows :

- $\forall s \in S, \forall \alpha \in A, \forall o \in O,$
  $Define(H1, s, \alpha, o, absent\_physician) \leftrightarrow$
  $(Use(H1, o, Medical\_record) \wedge$
  $\neg \exists s' \in S, (Empower(H1, s', physician) \wedge$
  $name(o) \in patient(s') \wedge status(s') \neq absent))$

that is subject s performs action α on object o if o is a medical record of some patient and there is no subject s' such that s' is the attending physician of this patient and s' is not absent.

Notice that we decide to define the context $absent\_physician$ as a prerequisite context, that is it is evaluated by querying the database to check if the attending physicians of a given patient are absent. This will be possible if the database actually stores such information.

If this is not the case, then another possibility would be to define the context $absent\_physician$ as a user-declared context. For instance, the nurse may be permitted to declare this context for a given patient. Of course, the two policies will not be identical since, in this second case, the nurse will be responsible for declaring the context $absent\_physician.$

This means that the fact that a given context will be defined as a prerequisite context or as a user-declared context strongly depends on the data stored in the system database.

3.6. Provisional context

Principle. The notion of provisional *obligation* was first introduced in [18, 15]. A provisional obligation is an obligation to perform some action that applies when some subject performs another action, generally in a given context (typically when the context is user-declared). In this case, the provisional obligation is automatically "fired" as a counterpart of the action performed by the subject.

We suggest modelling this notion using another type of context called provisional context. For this purpose, we shall first assume that the information system manages a log, that stores data about previous activity of users in the system. This is modelled by a view called Log. Object 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 ($date$).

Example of provisional context. To illustrate the approach, let us show how to model the following rule: "In the hospital $H1$, if a physician consult a given patient's medical record in a context of urgency, then this physician has a provisional obligation to send a medical report to the attending physician of this patient".

To model this rule, we first define a provisional context called $Urgent\_activity$ as follows:

- $\forall s \in S, \forall \alpha \in A, \forall o \in O,$
  $(Define(H1, s, \alpha, o, Urgent\_activity) \leftrightarrow$
  $\exists l, (Use(H1, l, Log) \wedge (actor(l) = s) \wedge$
  $(activity(l) = consult) \wedge (context(l) = Urgency))$

that is, in $H1$, subject s performs action α on object o in provisional context $Urgent\_activity$ if an activity $consult$ was logged in view Log in a context of $Urgency$ with subject s as an actor.

We then define a view $Medical\_report$ having three attributes: $addressee$ (the subject who is supposed to receive the report), $patient$ (the patient concerned by the report) and $content$ (the content of this medical report). Based on this view, we define a sub-view $Med\_report\_to\_attending\_physician$ as follows:

- $\forall o \in O,$
  $Use(H1, o, Med\_report\_to\_attending\_physician) \leftrightarrow$
  $(Use(H1, o, Medical\_report)$
  $\wedge patient(o) \in patient(addressee(o)))$

Using this provisional context and this view, we can then specify that a physician has a provisional obligation to send a medical report to the attending physician of this patient:

- $Obligation(H1, Physician, send,$
  $Med\_report\_to\_attending\_physician, Urgent\_activity)$

Other applications of provisional context. In [18], the authors consider two different types of provisional obligation: obligations that are fired *after* executing a given action (see the example in the previous section) and obligation that must be fulfilled *before* executing a given action.

Let us call "before obligation" the second type of provisional obligation and let us assume that a before obligation o_1 must be fulfilled before a permission p_1 is granted. In our approach, before obligation o_1 is actually modelled as a *permission*. If a given user performs the action corresponding to this permission, then a given provisional context is activated. In this provisional context, the (provisional) permission p_1 will be granted.

The advantage of this approach is that we only need to check (past) historical data to define provisional context. Notice also that a permission may be viewed as provisional (and not only an obligation). In the future, we plan to apply this notion of provisional permission to *workflow* access control. This will be useful to model that, in a workflow, permissions are granted as the user advances in a given task.

4. Related work

Several approaches have been proposed to model context management in access control and security policies. As we have seen earlier, the context makes it possible to express different kinds of constraint.

In the RBAC family models [21], $RBAC_2$ introduces the notion of constraints that control user-role assignment, permission-role assignment and session-role assignment constraints. Thus, $RBAC_2$ makes it potentially possible to establish security rules that depend on certain context information. Even though the constraints are not modelled in the initial RBAC model, some work have been done to formalize them [1, 2].

The contexts which are most obvious and most easily correspond to spatiotemporal information. [6] offers to broaden the role concept to the security requests environment. In the generalized model [7], the environment context is specified through a new type of role called the *environment role*. It is thus possible to express the constraints which are related to time intervals and trusted locations. Moreover the environment role hierarchy makes the contextual information management easier. In [4], the authors propose another extension of the RBAC model, called Temporal Role-Based Access Control (TRBAC). This work does not aim to express temporal conditions over the time of the action corresponding to the security request but offer means to activate roles periodically or thanks to a trigger. The concept of purpose was also suggested in several privacy model for instance the Privacy Model [9].

In most works related to context, the context corresponds to the ongoing activity in which the permissions are requested. In this case, controlling the workflow makes it possible to ”filter” the privileges granted to users. [11, 10] provide a model based on TMAC, called C-TMAC, in which users obtain permissions according to their role and the team in which they are involved. A context which defines the time, the location, etc, is associated to the team. These attributes are used to reduce permissions granted to each role. This model is particularly adapted to collaborative environments.

Through C-TMAC, contexts are used to take into account the need-to-know requirement, and the notion of just-in-time permission activation. A great number of works such as [14] are based on this idea.

The expression of a workflow environment through the context can also be considered as a means to respond to the least privilege principle. The option of expressing contexts is useful in the case of business and transaction activities. In such areas, the access control decision depends on specific sequences of events. For instance, [5] describes a context-sensitive access control model in which the rights are granted according to the actual task. The multiple tasks are defined in a global process. The context is activated through reference to this process definition.

If an access control model must allow the expression of contexts, it is also important to be able to evaluate this context. [14] shows how to apply a context-dependent access control mechanism on a commercial platform. Through the Antigone Condition Framework (ACF), [20] provides means to specify, implement and evaluate the context which is viewed as a set of external conditions.

These different models make it possible to capture information related to time, or to the user location, or to the team to which the user belong, or to the current workflow, etc. Even though these models are useful, they do not provide means to express a large number of different contexts within a single framework. It was precisely our objective in this paper.

5. Conclusion

In this paper, we present a model that includes an explicit expression of context and show how to use it to specify dynamic and flexible access control rules. We suggest a taxonomy of different types of context and model them in the Or-BAC model. Starting from elementary contexts, we also define conjunctive, disjunctive and negative contexts.

To control activation of a given context, the information system must store different data: temporal data to manage temporal context, user environment and system architecture representation for spatial context, and application data stored in the information base to define prerequisite context.

We also show that it is sometimes not possible to express all the possible conditions required to activate some contexts. For instance, if we consider the medical context of *urgency*, there are many different possibilities so that it is actually impossible to provide an exhaustive definition of such a context. In this case, we suggest defining the urgency context as a *user-declared* context: this is the responsibility of some authorized user to declare that this context is activated.

Thus, to activate a user-declared context, the user must be authorized to declare some objective (or purpose) of his or her activity. This is modelled by views, a given user-declared context being activated by inserting a given object in this view.

Finally, we define provisional context. This is used to model permissions, obligations or prohibitions that depend on previous actions performed by the user. To control activation of provisional context, the information system must store historical data of what happens in the system. Information systems generally provide such historical data through audit trail. Provisional obligation was already suggested in previous research work. Provisional permission may be also useful to model situations where users obtain permissions as their work proceeds. Similarly, provisional prohibition is another useful context to model situations where the user's previous activity leads to prohibition.

There were several other proposals to model contexts within an access control model but this is the first time that all the different contexts are expressed within a unique homogeneous framework. We are currently further investigating the notion of provisional context, in particular to model management of rights in workflow system. We are also applying this model in the framework of relational database administration.

Acknowledgement. For this work, Alexandre Miège is funded by France Télécom R&D and Frédéric Cuppens is partially funded by the MP6 RNRT project of the ministery of Research.

References

[1] Gail-Joon Ahn and Ravi Sandhu. Role-based Authorization Constraints Specification. *ACM Transactions on Information and Systems Security*, 3(4), 2000.

[2] Gail-Joon Ahn and Michael. E. Shin. Role-based Authorization Constraints Specification Using Object Constraint Language. In *Proceedings of the 10th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE 2001)*, Massachusetts, USA, 2001.

[3] D. E. Bell and L. J. La Padula. Secure Computer Systems: Unified Exposition and Multics Interpretation. Technical Report ESD-TR-75-306, MTR-2997, Rev. 1, MITRE Corporation, Bedfort, MA, March 1976.

[4] E. Bertino, P. A. Bonatti, and E. Ferrari. A Temporal role-based access control access model. *ACM transactions on information and System Security*, 2001.

[5] D. G. Cholewka, R. A. Botha, and J. H. P. Eloff. A Context-sensitive Access Control Model and Prototype Implementation. In *IFIP TC 11 16th Annual Working Conference on Information Security*, Beijing, China, August 2000.

[6] M. J. Covington, W. Long, S. Srinivasan, A. Dey, M. Ahamad, and G. Abowd. Securing context-aware applications using environment roles. In *Proceedings of the 6th ACM Symposium on Access Control Models and Technologies (SACMAT 2001)*, Chantilly, Virginia, USA, May 2001.

[7] M. J. Covington, M. J. Moyer, and M. Ahamad. Generalized role-based access control for securing future applications. In *Proceedings of the 23rd National Information Systems Security Conference, (NISSC)*, Baltimore, MD, USA, October 2000.

[8] N. Damianou, N. Dulay, E. Lupu, and M. Sloman. The Ponder Policy Specification Language. In *International Workshop, Policies for Distributed Systems and Neworks (Policy 2001)*, Bristol, UK, January 2001.

[9] S. Fischer-Hbner and A. Ott. From a Formal Privacy Model to its Implementation. In *Proceedings of the 21st National Information Systems Security Conference*, 1998.

[10] C. Georgiadis, I. Mavridis, and G. Pangalos. Context and Role Based Hybrid Access Control for Collaborative Environments. In *Proceedings of the Fifth Nordic Workshop on Secure IT Systems - Encouraging Co-operation (NORDSEC 2000)*, Reykjavik, Iceland, October 2000.

[11] Christos K. Georgiadis, Ioannis Mavridis, George Pangalos, and Roshan K. Thomas. Flexible team-based access control using contexts. In *Proceedings of the 6th ACM Symposium on Access Control Models and Technologies (SACMAT 2001)*, Chantilly, Virginia, USA, May 2001.

[12] Joseph Halpern and Vicky Weissman. Using First-Order Logic to Reason about Policies. In *Proceedings of the 16th IEEE Computer Security Foundations Workshop (CSFW'03)*, 2003.

[13] M. Harrison, W. Ruzzo, and J. Ullman. Protection in operating systems. *CACM*, 19(8):461–471, August 1976.

[14] R. Holbein, O. Morger, U. Nitsche, and S. Teufel. Realization of a Context-Dependent Access Control Mechanism on a Commercial Platform. In *IFIP 14th International Conference on Information Security (IFIP/Sec'98)*, Vienna-Budapest, Austria-Hungary, August 31 - September 4 1998.

[15] S. Jajodia, M. Kudo, and V.S. Subrahmanian. Provisional Authorizations. In editor A. Ghosh, editor, *E-commerce Security and Privacy*, pages 133–159. Kluwer Academic Publishers, 2001.

[16] S. Jajodia, P. Samarati, and V. S. Subrahmanian. A Logical Language for Expressing Authorizations. In *Proceedings of IEEE Symposium on Security and Privacy*, pages 31–42, Oakland, California, USA, 1997.

[17] A. Abou El Kalam, R. El Baida, P. Balbiani, S. Benferhat, F. Cuppens, Y. Deswarte, A. Miège, C. Saurel, and G. Trouessin. Organization Based Access Control. In *Proceedings of IEEE 4th International Workshop on Policies for Distributed Systems and Networks (POLICY 2003)*, Lake Como, Italy, June 2003.

[18] M. Kudo and S. Hada. XML Document Security based on Provisional Authorization. In *Proceedings of the 7th ACM Conference on Computer and Communication Security (CCS 2000)*, Athens, Greece, 2000.

[19] Markus G. Kuhn. Optical Time-Domain Eavesdropping Risks of CRT Displays. In *Proceedings of the 2002 IEEE Symposium on Security and Privacy*, Oakland, California, USA, 2002.

[20] Patrick McDaniel. On Context in Authorization Policy. In *Proceedings of the 8th ACM Sympoisum On Access Control Models and Technologies (SACMAT 2003)*, Como, Italy, June 2003.

[21] R. S. Sandhu, E. J. Coyne, H. L. Feinstein, and C. E. Youman. Role-Based Access Control Models. *IEEE Computer*, 29(2):38–47, February 1996.

[22] Roshan K. Thomas. TMAC: A primitive for Applying RBAC in collaborative environment. In *2nd ACM, Workshop on RBAC*, pages 13–19, FairFax, Virginia, USA, 1997.