Privacy-aware Role Based Access Control

Qun Ni
Purdue University, USA
ni@cs.purdue.edu

Alberto Trombetta
Insubria University, Italy
alberto.trombetta@uninsubria.it

Elisa Bertino
Purdue University, USA
bertino@cs.purdue.edu

Jorge Lobo
IBM T.J. Watson, USA
jlobo@us.ibm.com

ABSTRACT

Privacy has been acknowledged to be a critical requirement for many business (and non-business) environments. Therefore, the definition of an expressive and easy-to-use privacy-related access control model, based on which privacy policies can be specified, is crucial. In this work we introduce a family of models (P-RBAC) that extend the well known RBAC model in order to provide full support for expressing highly complex privacy-related policies, taking into account features like purposes and obligations. We also compare our work with access control and privacy policy frameworks such as P3P, EPAL, and XACML.

Categories and Subject Descriptors

C.2.0 [**Computer Communication Networks**]: General—*security and protection*; D.4.6 [**Operating Systems**]: Security and Protection—*Access Controls*; K.6.5 [**Management of Computing and Information Systems**]: Security and Protection

General Terms

Management, Security, Standardization

Keywords

Privacy, Role Based Access Control, Model, Purpose

1. INTRODUCTION

Privacy is today a key issue in information technology and has received increasing attention from consumers, companies, researchers and legislators. Legislative acts, such as Health Insurance Portability and Accountability Act (HIPAA) [25] for healthcare and Gramm Leach Bliley Act (GLBA) [26] for financial institutions, require enterprises to protect the privacy of their customers[1]. Although enterprises have adopted various strategies to protect customer privacy and to communicate their privacy policies to customers, such as publishing a privacy policy on websites [2, 9, 6] possibly based on P3P, or incorporating privacy seal programs (e.g. TRUSTe [24], ESRB, BBBOnline, and CPAWebTrust), in these approaches there are not systematic mechanisms that describe how consumer personal data is actually handled after it is collected. Privacy protection can only be achieved by enforcing privacy policies within an enterprise's online and offline data processing systems. Otherwise, enterprises' actual practices might intentionally or unintentionally violate the privacy policies published at their websites.

Conventional access models, such as Mandatory Access Control (MAC), Discretionary Access Control (DAC), and Role Based Access Control(RBAC) [11, 22], are not designed to enforce privacy policies and barely meet privacy protection requirements[12], particularly, purpose binding (i.e. data collected for one purpose should not used for another purpose without user consent), conditions and obligations. The significance of purposes, conditions, and obligations originates from OECD Guidelines [19] on the Protection of Privacy and Transborder Flows of Personal Data, current privacy laws in the United States, and public privacy policies of some well know organizations. The OECD guidelines are, to the best of our knowledge, the most well known set of private information protection principles, on which many other guidelines, data-protection laws, and public privacy policies are based. Purposes are directly applied in the OECD *Data Quality Principle*, *Purpose Specification Principle*, and *Use Limitation Principle*. Purposes are also widely used for specifying privacy rules in legislative acts and actual public policies. HIPPA[25] rules clearly state purposes. The majority of public privacy documents posted at well known sites also specify purposes.

[1]We use the term *customer* to refer to an individual who releases personally identifiable information to an enterprise. We will also use the term "data subject" with the same meaning.

SACMAT'07, June 20-22, 2007, Sophia Antipolis, France.

Obligations, that is, actions to be performed after an action has been executed on data objects, are necessary for some cases. For example, the OECD *Accountability Principle* states that "A data controller should be accountable for complying with measures which give effect to the principles stated above". A common approach to implement this principle in operating systems or DBMS is to log each data access as an event. Executing logging actions could be an obligation for the majority of privacy policies.

Conditions, that is, prerequisites to be met before any action can be executed, are critical in some cases. One of these cases is related to children information. One of the most important rules in COPPA is the so called *Verifiable Parental Consent*(VPC): before collecting, using or disclosing personal information from a child, an operator must obtain verifiable parental consent from the child's parent. The VPC is a condition that must be satisfied before collecting and accessing personal information related to children under thirteen.

However, existing access control technology can be used as a starting point for managing personal identifiable information in a trustworthy fashion [20]. A language used for privacy policies must be the same as or integrated with the language used for access control policies, because both types of policy usually control access to the same resources and should not conflict with one another [3]. Hence, we propose a family of Privacy-aware Role Based Access Control (P-RBAC) models that naturally extend classical RBAC models to support privacy policies.

We believe that an RBAC-based solution to the problem of privacy-aware access control may have a great potential. It could be easily deployed in systems already adopting RBAC and would thus allow one to seamlessly introduce access control policies specialized for privacy enforcement. The goal of the work reported in this paper is to extend the RBAC model in order to support privacy-aware access control. In our model, referred to as P-RBAC, privacy policies are expressed as permission assignments (PA); these permissions differ from permissions in classical RBAC because of the presence of additional components, representing privacy-related information. We also develop conflict analysis algorithms to detect conflicts among PA, thus avoiding the problems that EPAL[1] rules have because of its sequential semantics [5].

2. P-RBAC MODELS

In order to extend classical RBAC to support (possibly complex) privacy policies, consistently with the approach adopted for classical RBAC, we took the approach of defining a family of *Privacy aware RBAC* (P-RBAC) conceptual models (see Figure 1) characterized by different modeling capabilities. Core P-RBAC, the base model, is at bottom. There is a tradeoff when designing Core P-RBAC. On the one hand, Core P-RBAC should have sufficient expressive power for representing public privacy policies, privacy statements and privacy notices in Web sites, and policies based on privacy related acts, such as HIPPA [25], COPPA[10], and GLBA[26], in the US. On the other hand, conflicts detection in Core P-RBAC should remain tractable. Advanced models in the family extend Core P-RBAC with additional modeling constructs. Hierarchical P-RBAC introduces the notions of *Role Hierarchy*(RH), *Data Hierarchy*(DH), and *Purpose Hierarchy*(PH); it thus enhances Core P-RBAC with a hierarchical organizations for three important entities of Core P-RBAC. Conditional P-RBAC introduces *Permission Assignment Sets* and *Boolean Expressions*; its main goal is to provide a language for expressing conditions richer than the simple condition language provided by Core P-RBAC. Universal P-RBAC combines functionalities of both Conditional P-RBAC and Hierarchical P-RBAC.

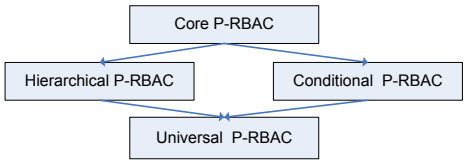

Figure 1: The family of conceptual P-RBAC models

2.1 The Core P-RBAC Model

Core P-RBAC is illustrated in Figure 2. The model includes several sets of entities: Users(U), Roles(R), Data(D), Actions(A), Purposes(Pu), Obligations(O), and conditions (C) expressed by using a customized language, referred to as LC_0.

A user in our model is human being, and a role represents a job function or job title within the organization with some associated semantics regarding the authority and responsibility conferred on a member of the role. Data in our model means any information relating to an identified or identifiable individual. An action is an executable image of a program, which upon invocation executes some function for the user. The types of action and data object that P-RBAC controls depend on the type of system in which they will be implemented.

The motivations for introducing purposes, conditions, and obligations in Core P-RBAC are illustrated in the introduction. Core P-RBAC directly models these notions. In Core P-RBAC, as in classical RBAC, permissions are assigned to roles and users obtain such permissions by being assigned to roles. The distinctive feature of Core P-RBAC lies in the complex structure of privacy permissions, which reflects the highly structured ways of expressing privacy rules to represent the essence of OECD principles and privacy acts. Therefore, aside from the data and the action to be performed on it, a privacy permission explicitly states the intended purpose, along with the conditions under which the

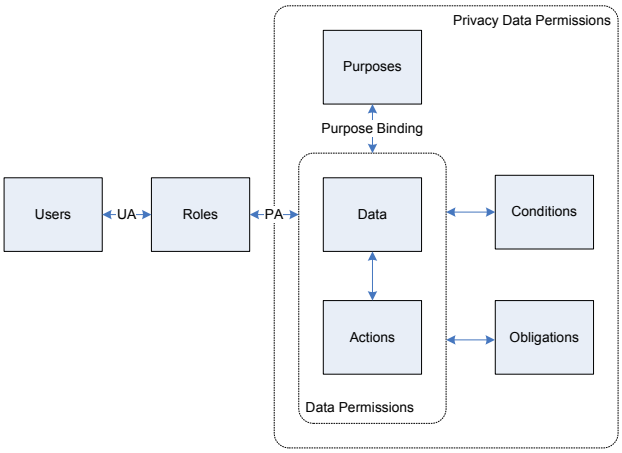

Figure 2: Core P-RBAC model

permission can be given, and the obligations that are to be finally performed.

Core P-RBAC conditions should not be confused with constraints, which are a very relevant component of the classic RBAC model. Constraints are a powerful mechanism for laying out higher-level organizational policies, while conditions are a mechanism to precisely define a permission. A common example of constraints is separation of duties. Handling separately privacy-related conditions and constraints allows us to focus on how to effectively and precisely model the necessary prerequisites for validating and enforcing privacy policies. We defer the treatment of constraints to our future work.

DEFINITION 2.1. The core P-RBAC model is composed of the following components:

- A set U of *users*, a set R of *roles*, a set D of *data*, a set Pu of *purposes*, a set A of *actions*, a set O of *obligations*, and a condition language LC_0.
- The set of *Data Permissions* $DP = \{(a, d) \mid a \in A,\ d \in D\}$.
- The set of *Privacy-sensitive Data Permission* $PDP = \{(dp, pu, c, o) \mid dp \in DP,\ pu \in Pu, c \text{ is an expression of } LC_0,\ o \in \mathcal{P}(O)\}$. $\mathcal{P}(O)$ is the power set of O.
- *User Assignment* $UA \subseteq U \times R$, a many-to-many mapping user to role assignment relation.
- *Privacy-sensitive Data Permission Assignment* $PDPA \subseteq R \times PDP$, a many-to-many mapping privacy-sensitive data permission to role assignment relation. □

In what follows we provide additional details on the condition language of Core P-RBAC and elaborate on various aspects concerning conflicts among permissions and obligations.

2.1.1 The Basic Condition Language LC_0

Core P-RBAC includes a simple language for expressing conditions; they are expressed using *context variables*. Such variables record privacy-relevant information that is to be taken into account when enforcing privacy permissions. Even though the LC_0 condition language has limited expressive power, it is able to model several conditions usually found in privacy permissions. The conditions that can be expressed by LC_0 are defined in what follows.

DEFINITION 2.2. Let X be a set of context variables; each variable $x \in X$ has a finite domain of possible values, denoted as D_x; every domain is equipped with a pair of corresponding relational operators $=$ and $\neq$. An atomic condition ac defined over X has the form $(x\ op_r\ v)$ where $x \in X,\ v \in D_x,\ op_r \in \{=, \neq\}$. The conditions of LC_0 (over X) are defined as follows:

- An atomic condition is a condition of LC_0.
- Let c_i and c_j be conditions of LC_0; then $c_i \wedge c_j$ is a condition of LC_0. □

Examples of some commonly used context variables are listed in what follows:

- OwnerConsent, domain={yes, no}; it represents data subject's consent.
- ParentalConsent, domain={yes, no}; it represents parent's consent for kids under 13.
- OwnerAge, domain={under13, teenage, adult}; it represents data subject's age scope.
- CurrentTime, domain={9AM-5PM, 5PM-11PM, 11PM-9AM}; it represents different temporal periods.

We illustrate through an example a privacy permission expressed with LC_0 conditions. Suppose that www.toys.org is a web site aiming at kids that deploys the following elements for expressing its privacy permissions[2]:

- Role = {DeliveryPartner, Marketing, BusinessPartner},
- Action = {Read},
- Data = {Orderinfo, PostalAddress, EmailAddress},
- Purpose = {Promotion, Billing, Shipping, Research},
- Context variables ={DataUser, OwnerConsent, ParentalConsent, OwnerAge}.

The following privacy policies

1. "Delivery partner can access customers' PostalAddress for shipping purpose."

[2]For simplicity, we only enumerate subsets of real elements for illustrative purposes, and some of them are used in other examples in the following sections

2. "Marketing employee can only access customers' Email address for promotion if they allow to do so."
3. "Our business partners may access customers' order information for research; however, customers will be informed by official Email."

are expressed as follows in Core P-RBAC:
PA_1: (DeliveryPartner, ((Read, PostalAddress), Shipping, N/A, $\emptyset$)))
PA_2: (MarketingEmployee, ((Read, EmailAddress), Promotion, OwnerConsent=Yes, $\emptyset$))
PA_3: (BusinessPartner, ((Read, OrderInfo), Research, N/A, Notify(ByOfficialEmail)))

2.1.2 Relationships between permission assignments

As new requirements for privacy protection arise, more and more new permission assignments are applied. For instance, in order to comply with COPPA, the privacy officer may define a new policy:

4. "Marketing employees can only access email addresses of customers having age under 13, for promotion purposes, if their parents allow to do so."

The corresponding permission assignment is expressed in Core P-RBAC as:
PA_4: (MarketingEmployee, ((Read, EmailAddress), Promotion, OwnerAge=Under13 $\wedge$ ParentalConsent=Yes, $\emptyset$).

Recalling that the permission assignment PA_2 is already defined, we now have two permission assignments related to the role MarketingEmployee. Such role accesses customers' Email address for promotion purposes. What is the effect of these two permission assignments, when considered together? Generally, one should apply PA_2 in order to access email addresses of customers older than 13 and, in sequence, apply PA_4 in order to access email addresses of customers younger than 13.

A tricky issue here is whether the system should verify OwnerConsent when kids' email addresses are accessed for promotion. Core P-RBAC achieves that by considering different permission assignments as linked by a conjunction. That is, if a user U of Role R want to read data D for purpose Pu, all permissions of R related to $((Read, D), Pu)$ must be checked. Only if there exists at least one permission, and U can meet all conditions in the permission(s), U can read the D. If a new permission assignment is related to the same role, same data, same action and same purpose of some old permission assignments, it is not used to relax the access condition but to make the access condition stricter. If privacy officers want to relax the condition, they can do so by revising the original permission assignment instead of writing a new one.

If there are two permission assignments $(R_1, ((A_1, D_1), Pu_1, C_1, \emptyset))$ and $(R_1, ((A_1, D_1), Pu_1, C_2, \emptyset))$ [3], could we simply replace them with a new permission assignment $(R_1, ((A_1, D_1), Pu_1, C_1 \wedge C_2, \emptyset))$? The answer is negative. Consider PA_2 and PA_4, after rewriting them, we obtain:
PA_5: (MarketingEmployee, ((Read, EmailAddress), Promotion, OwnerConsent = Yes $\wedge$ OwnerAge = Under13 $\wedge$ ParentalConsent = Yes, $\emptyset$)).

Translating PA_5 into plain English, we obtain "Marketing employee can only access customers' Email address for promotion if the customers are younger than 13 and agree to do so and their parents allow to do so." Apparently, we miss something because now MarketingEmployee cannot access teenagers' and adults' email addresses, for any purpose. The reason for this is the context variable $OwnerAge$ in PA_4. The importance of the variable OwnerAge is that its different values actually separate the values of EmailAddress into three disjoint sets: email addresses for customers having age under 13, email addresses for teenager customers, and email addresses for adult customers. PA_2 thus applies to all three kinds of customers, while PA_4 only applies to email addresses for customers younger than 13. Simply combining condition in PA_2 with condition in PA_4 actually removes permission assignments for teenagers' and adults' email addresses.

In order to handle this situation, we introduce the notion of splitting context variables. Such variables partition the data with which they are associated according to the values they assume.

Definition 2.3. A splitting context variable (SCV) is a context variable that satisfies the following conditions.

1. A SCV is related to data subject's information.
2. The values of an SCV partition data items into disjoint sets.
3. A SCV is not used to represent information about consent. □

According to this definition, OwnerAge and OwnerSalary are SCV, whereas OwnerConsent and CurrentTime are not. The notion of SCV is very important and is used in definitions in the subsequent sections. We are now able to give an answer to the aforementioned question: only if both C_1 and C_2 do not involve SCV, or the SCV that they involve have the same values, they could be safely rewritten into $C_1 \wedge C_2$. Consider the following two permission assignments:
PA_6: (MarketingEmployee, ((Read, EmailAddress), Promotion, N/A, $\emptyset$))
PA_7: (MarketingEmployee, ((Read, EmailAddress), Promotion, OwnerConsent=Yes, $\emptyset$)).

They can be rewritten as:
PA_8: (MarketingEmployee, ((Read, EmailAddress), Promotion, OwnerConsent=Yes, $\emptyset$)).

[3]There are no obligations in these examples, and we discuss the cases with obligations later on in this section and the following section

Similarly, the following two permission assignments:
PA_9: (MarketingEmployee, ((Read, EmailAddress), Promotion, OwnerAge = Under13 ∧ ParentalConsent=Yes, ∅))
PA_{10}: (MarketingEmployee, ((Read, EmailAddress), Promotion, OwnerAge = Under13 ∧ OwnerConsent=Yes, ∅))
can be rewritten as:
PA_{11}: (MarketingEmployee, ((Read, EmailAddress), Promotion, OwnerAge = Under13 ∧ OwnerConsent=Yes ∧ ParentalConsent=Yes, ∅)).

PA_6 and PA_7 are equivalent to PA_8, and PA_{11} is equivalent to PA_9 and PA_{10}.

As an additional example, we could rewrite PA_2 and PA_4 into the following permissions:
PA_{12}: (MarketingEmployee, ((Read, EmailAddress), Promotion, OwnerAge ≠ Under13 ∧ OwnerConsent=Yes, ∅))
PA_{13}: (MarketingEmployee, ((Read, EmailAddress), Promotion, OwnerAge = Under13 ∧ OwnerConsent=Yes ∧ ParentalConsent=Yes, ∅)).

Although we cannot reduce the total number of permission assignments[4], the effects of new permission assignments PA_{12} and PA_{13} are easier to understand than that of PA_4 and PA_2.

Notice that PA_2 and PA_4 do not have obligations, what if they do have? Consider the following examples:
PA_{14}: (MarketingEmployee, ((Read, EmailAddress), Promotion, OwnerConsent=Yes, {Log()}))
PA_{15}: (MarketingEmployee, ((Read, EmailAddress), Promotion, OwnerAge=Under13 ∧ ParentalConsent=Yes, {Notify()})).

Intuitively, PA_{14} is fine with customers' email address, regardless to their ages. This means that the `Log` activity should be invoked as well, after accessing Under13 customers' data. Therefore, their equivalent forms are:
PA_{16}: (MarketingEmployee, ((Read, EmailAddress), Promotion, OwnerAge ≠ Under13 ∧ OwnerConsent=Yes, {Log()}))
PA_{17}: (MarketingEmployee, ((Read, EmailAddress), Promotion, OwnerAge = Under13 ∧ OwnerConsent=Yes ∧ ParentalConsent=Yes, {Log(), Notify()})).

In summary, a data access request related to role R, data D, action A, purpose Pu is authorized only if all conditions in permission assignments related to (R, D, A, Pu) are satisfied. If so, obligations in all applicable permission assignments are invoked after the data access request.

2.1.3 Obligations

In existing privacy policies specification languages, obligations are assumed to behave rather predictably. For example, in EPAL, obligations are assumed not to interfere with each other. However, such assumption does not hold in practice. For instance, different notifications may apply to different communication channels and/or with different information, e.g. one notification only provides information, like disclosure to third parties, to a data subject whereas another notification further provides the "opt-out" option to the data subject. In such case, the differences between these notifications are represented by different parameters in obligations. A first, very rough characterization of *conflicting* obligations is obligations that have the same name and different parameters. It is important to note that the above characterization of conflicting obligations is very preliminary. Detection of conflicting obligations heavily depends on the implementation. A full-fledged treatment of such an issue would require the definition of a formal specification of obligations activities, among other things. We plan to develop a suitable language for such specification as future work [23].

2.1.4 Conflicting Permissions Assignment

Complex environments, such as large enterprises, usually have to comply with complex security and privacy policies. As such, it is possible that the more complex a security policy is, the larger is the probability that such policy contains inconsistent and conflicting parts. In particular, permission assignments could conflict because of new requirements, new regulations, or just human mistakes. In the section, we discuss what are the causes of conflicting permission assignments.

Consider the following permission assignments:
PA_{18}: (BusinessPartner, ((Read, OrderInfo), Research, OwnerAge=Teenager ∧ CurrentTime=5PM-11PM, ∅))
PA_{19}: (BusinessPartner, ((Read, OrderInfo), Research, OwnerAge=Adult ∧ CurrentTime=11PM-9AM, ∅)).

Notice that there is a SCV OwnerAge used in the conditions but with different values; therefore those permission assignments do not conflict with each other because PA_{18} and PA_{19} actually work on different customers' OrderInfo. We call them incomparable permission assignments because they have *incomparable conditions*, that is, a SCV exists which has two disjoint value sets in the two conditions.

Definition 2.4. Let c_i and c_j be two conditions in two permission assignments. We say that c_i and c_j are incomparable conditions if there exists a common SCV that has disjoint value sets in conditions c_i and c_i. Otherwise, we say that c_i and c_j are comparable conditions, written as $c_i \approx c_j$. □

Consider the following two permission assignments which include comparable conditions:
PA_{20}: (BusinessPartner, ((Read, OrderInfo), Research, CurrentTime ≠ 9AM-5PM, ∅))
PA_{21}: (BusinessPartner, ((Read, OrderInfo), Research, CurrentTime ≠ 5PM-11PM, ∅))

[4] PA_2 and PA_4 cannot be rewritten into one permission assignment.

Because PA_{20} allows data access during 5PM-11PM or 11PM-9AM and PA_{21} allows data access during 9AM-5PM or 11PM-9AM, a data request occurs during 11PM-9AM could be authorized. These two permission assignments are compatible because they have *compatible conditions*: the intersection of value sets of context variable CurrentTime in different permission assignments is not empty[5].

Besides compatible conditions, we may have conflicting conditions.

PA_{22}: (BusinessPartner, ((Read, OrderInfo), Research, CurrentTime = 5PM-11PM, $\emptyset$))

PA_{23}: (BusinessPartner, ((Read, OrderInfo), Research, CurrentTime = 11PM-9AM, $\emptyset$)).

PA_{22} specifies that business partners are authorized to access order information for research during 5PM-11PM, whereas PA_{23} allows partners' access only during 11PM-9AM. Without loss of generality, we assume 11PM is not a common time point between 5PM-11PM and 11PM-9AM. Hence, when data request is issued, CurrentTime could not be both 5PM-11PM and 11PM-9AM. Therefore, any data request will be denied according to these two permission assignments. These two permission assignments conflict with each other because they have *conflicting conditions*, that is, no value of the context variable CurrentTime could satisfy both conditions.

DEFINITION 2.5. Let c_i and c_j be two comparable conditions in two permission assignments. We say that c_i and c_j are conflicting conditions if there exists at least one common context variable in c_i and c_j that has disjoint value sets, written as $c_i \asymp c_j$. Otherwise, we say that c_i and c_j are compatible conditions. □

Consider the following permission assignments which include conflicting obligations:

PA_{24}: (BusinessPartner, ((Read, OrderInfo), Research, CurrentTime $\neq$ 9AM-5PM, {Notify()}))

PA_{25}: (BusinessPartner, ((Read, OrderInfo), Research, CurrentTime $\neq$ 5PM-11PM, {Notify(Opt-out) })).

Once a data request is authorized, the system does not know which obligation should be executed (either Notify or Notify with Opt-out); therefore PA_{24} conflicts with PA_{25}. We denote the fact that two obligations o_i and o_j conflict as $o_i \asymp o_j$.

Based on aforementioned definitions and examples, we give the definition of conflicting permission assignments.

DEFINITION 2.6. Let $PA_i = (r_i, (d_i, a_i), pu_i, c_i, o_i)$ and $PA_j = (r_j, (d_j, a_j), pu_j, c_j, o_j)$ be two privacy-sensitive data permission assignments. We say that PA_i and PA_j are conflicting if one of the following two conditions holds:

- $(r_i = r_j) \wedge (d_i = d_j) \wedge (a_i = a_j) \wedge (pu_i = pu_j) \wedge (c_i \asymp c_j)$.
- $(r_i = r_j) \wedge (d_i = d_j) \wedge (a_i = a_j) \wedge (pu_i = pu_j) \wedge (c_i \approx c_j) \wedge (o_i \asymp o_j)$. □

In Core P-RBAC, conflicting permission assignments should be detected and one of them should be removed to prevent ambiguities when enforcing permissions.

2.1.5 Conflict Detection

Conflicting permission assignment detection is important in order to guarantee the consistency of permissions assignments. In this section, we present an algorithm to detect conflicts between two permission assignments and an algorithm to check conflicts in all permission assignments. Because condition conflicts detection is the basic building block of our solution, the corresponding algorithm (see Algorithm 1) is given first. The key point of the algorithm is that we first sort context variables used in conditions according to their name, then make a disjoint test for the value sets for a variable in the various conditions.

Algorithm 1 Condition-Conflict-Test(cva_1, cva_2)

Require: cva_1 and cva_2 to be two conditions applied in two permission assignments

```
1: cvl1 ← Sort context variables used in cva1 according to their name
2: cvl2 ← Sort context variables used in cva2 according to their name
3: i = j = 1
4: result = false
5: while i ≤ |cvl1| do
6:   while j ≤ |cvl2| do
7:     if cvl1[i].name = cvl2[j].name then
8:       if disjointTest(cvl1[i].value, cvl2[j].value, op1, op2) then
9:         if cvl1[i].SCV = true then
10:          return false
11:        else
12:          result = true
13:        end if
14:      else
15:        i++; j++
16:      end if
17:    else if cvl1[i].name < cvl2[j].name then
18:      i++
19:    else
20:      j++
21:    end if
22:  end while
23: end while
24: return result
```

[5]Only data access occurring between 11PM and 9AM is possible to be allowed. It makes sense because we may only allow our business partner to access data during spare time.

Based on the Condition-Conflict-Test algorithm, the conflict detection algorithm (see Algorithm 2) is given; this algorithm receives as input two permission assignments and determines whether they conflict according to the definition of conflicting permission assignments[6].

Algorithm 2 PA-Conflict-Detection(pa_1, pa_2)

Require: pa_1 and pa_2 to be two permission assignments and $pa_1.r = pa_2.r$

1: **if** $pa_1.d \neq pa_2.d$ or $pa_1.a \neq pa_2.a$ or $pa_1.pu \neq pa_2.pu$ **then**
2: **return false**
3: **end if**
4: **if** Condition-Conflict-Test($pa_1.cva, pa_2.cva$) **then**
5: **return true**
6: **else if** SCV-DisjointTest($pa_1.cva, pa_2.cva$) **then**
7: **return** Obligation-Conflict-Test($pa_1.o, pa_2.o$)
8: **else**
9: **return false**
10: **end if**

Based on the conflict test on two permission assignments, we now introduce Algorithm 3 that detects conflicts on all permission assignments. We apply a hash function to group all similar permission assignments first, compare each two of permission assignments in same group, and record conflicting permission assignments in a list.

Algorithm 3 PAL-Conflict-Detection(pal)

Require: pal a list of permission assignments; Hash(r,d,a,pu) a hash function based on role, data, action, and purpose

Ensure: $pacl$ a list of pairs of conflicting permission assignments

1: **for all** pa such that $pa \in pacl$ **do**
2: Key=Hash($pa.r$,$pa.d$,$pa.a$,$pa.pu$)
3: Insert(paq,key,pa) {paq is an array of which the key_{th} element is a pair $(num, head)$. The $head$ points to a linked list of pa and the num represents the number of elements in the list. Insert adds pa to the key_{th} list and increase the num by one}
4: **end for**
5: **for all** $(num, head) \in paq$ such that $num \geq 2$ **do**
6: **for** i=1 to num **do**
7: **for** $j = i$ to num **do**
8: **if** PA-Conflict-Detection((List.nth($head$,i)).pa, (List.nth($head$,j)).pa) **then**
9: List.insert($pacl$, ((List.nth($head$,i)).pa, (List.nth($head$,j)).pa))
10: **end if**
11: **end for**
12: **end for**
13: **end for**

2.2 Hierarchical P-RBAC

The *Hierarchical* P-RBAC component introduces role hierarchies (RH), data hierarchies (DH) and purpose hierarchies (PH). Role hierarchies represent an important notion in RBAC and are very often present in extensions to RBAC [22]. They are also commonly implemented in systems adopting RBAC. Role hierarchies are a natural means for structuring roles to reflect an organization's lines of authority and responsibility. By convention more powerful (or senior) roles are shown at the top of these diagrams, whereas less powerful (or junior) roles are at the bottom. Mathematically, role hierarchies are partial orders. A partial order is a reflexive, transitive and anti-symmetric relation. Inheritance is reflexive because a role inherits its own permissions; transitivity is a natural requirement in this context, and anti-symmetry rules out roles that inherit from one another and would therefore be redundant.

The purposes mentioned in privacy policies are often high-level, such as, for example, "marketing". More specific privacy permissions may deal with more specific purposes that fall under the domain of a high-level purpose, such as direct marketing by a third party via e-mail. Hence, in common business environments, purposes naturally have some hierarchical relationships among them [7], i.e., generalization and specialization relationships. This suggests that purposes can be organized according to the hierarchical relationships to simplify their management. Mathematically, a purpose hierarchy is represented as a tree. Each purpose (except the root purpose) has exactly one parent purpose and there are no cycles. A parent node represents a more general purpose than those represented by its children nodes. Thus the hierarchy of purposes can be intended as a grouping of more particular purposes into more general ones. For instance, the parent purpose P_5 groups all its children purposes $\{P_2, P_3\}$. Access for some kind of parent purpose, e.g. P_5, is allowed only if access for all children purposes, e.g. P_2 and P_3, is allowed.

The same argument applies to data objects. Data hierarchies are a common approach to organize large data sets by exploiting relationships among the various data objects. On the other side, data objects referred to in privacy permissions are often high-level classifications of data, such as customer contact information, or medical records. These data objects are used to distinguish classifications of collected data that need to be treated differently from a privacy point of view. Organizing the high level data according to a hierarchy not only directly models common data organizations, but also improves the expressiveness of permissions. For example, a permission referring to a high-level data applies to

[6]All algorithms in the paper are tractable, and most of them even run in linear time. Please refer to [21] for more details.

all its parts, thus reducing the total number of permissions. As for purposes hierarchies, a data hierarchy is represented as a tree structure. The meaning associated with a data hierarchy is analogous to the one mentioned for purpose hierarchies: a parent node represents a more general kind of data than the data associated with its children nodes. An access to a parent data object is allowed only if access to all its children is allowed. For instance, the parent data D_5 groups all its children $\{D_2, D_3\}$. Hierarchical P-RBAC extends Core P-RBAC with the three hierarchies introduced by the definition.

DEFINITION 2.7. Let R be a set of roles, D a set of data, and Pu a set of purposes. The Role Hierarchy, Data Hierarchy and Purpose Hierarchy are defined as follows.

- The *role hierarchy* $RH \subseteq R \times R$ is a partial order on roles, denoted as $\succeq_R$.
- The *data hierarchy* $DH \subseteq D \times D$ is a tree, that is a partial order, denoted as $\succeq_D$, on data objects such that each data object has at most one immediate ancestor.
- The *purpose hierarchy* $PH \subseteq Pu \times Pu$ is a tree, that is a partial order, denoted as $\succeq_{Pu}$, on purposes such that each data only has at most one immediate ancestor. □

The introduction of hierarchies of purpose and data supports compact permission assignments. For instance, the following permission assignments
PA_{26}: $(R_5, ((D_2, a), P_2, \emptyset))$
PA_{27}: $(R_5, ((D_2, a), P_3, \emptyset))$
PA_{28}: $(R_5, ((D_3, a), P_2, \emptyset))$
PA_{29}: $(R_5, ((D_3, a), P_3, \emptyset))$
could be replaced by only one permission assignment:
PA_{30}: $(R_5, ((D_5, a), P_5, \emptyset))$

2.2.1 Conflict Detection in Permission Assignments

Before presenting the conflict detection algorithm for Hierarchical P-RBAC, we introduce an extended definition of conflicting privacy sensitive data permissions. Such definition takes into account hierarchies.

DEFINITION 2.8. *Let* $PA_i = (r_i, (d_i, a_i), pu_i, c_i, o_i)$ *and* $PA_j = (r_j, (d_j, a_j), pu_j, c_j, o_j)$ *be two privacy-sensitive data permission assignments in Hierarchical P-RBAC. We say that* PA_i *and* PA_j *are conflicting if one of the following two conditions holds:*

- $((r_i \succeq_R r_j) \vee (r_i \succeq_R r_j)) \wedge ((d_i \succeq_D d_j) \vee (d_j \succeq_D d_i)) \wedge (a_i = a_j) \wedge ((pu_i \succeq_P pu_j) \vee (pu_j \succeq_P pu_i)) \wedge (c_i \asymp c_j)$
- $((r_i \succeq_R r_j) \vee (r_i \succeq_R r_j)) \wedge ((d_i \succeq_D d_j) \vee (d_j \succeq_D d_i)) \wedge (a_i = a_j) \wedge ((pu_i \succeq_P pu_j) \vee (pu_j \succeq_P pu_i)) \wedge (c_i \approx c_j) \wedge (o_i \asymp o_j)$.

Based on the above definition, we introduce the HPA-Conflict-Detection algorithm that determines whether two permissions conflicts. `Role-Relationship-Test` is used to test whether two roles have ancestor-descendent relationship, `Data-Relationship-Test` is used to test whether two data have ancestor-descendent relationship, and `Purpose-Relationship-Test` is used to test whether two purposes have ancestor-descendent relationship. Other tests in HPA-Conflict-Detection are as same as those in PA-Conflict-Detection.

Algorithm 4 HPA-Conflict-Detection(pa_1, pa_2)

Require: pa_1 and pa_2 to be two permission assignments
1: **if** NOT Role-Relationship-Test($pa_1.r$, $pa_2.r$) **then**
2: **return false**
3: **end if**
4: **if** NOT Data-Relationship-Test($pa_1.d$, $pa_2.d$) **then**
5: **return false**
6: **end if**
7: **if** NOT Purpose-Relationship-Test($pa_1.pu$, $pa_2.pu$) **then**
8: **return false**
9: **end if**
10: **if** Condition-Conflict-Test($pa_1.cva, pa_2.cva$) **then**
11: **return true**
12: **else if** SCV-DisjointTest($pa_1.cva, pa_2.cva$) **then**
13: **return** Obligation-Conflict-Test($pa_1.o, pa_2.o$)
14: **else**
15: **return false**
16: **end if**

Based on HPA-Conflict-Detection, we present HPAL-Conflict-Detection algorithm to detect all possible conflicting permission assignments in all permission assignments. The algorithm simply compares each two of them and insert conflicting permission assignments in a list.

3. RELATED WORKS

In this section, we provide a brief comparison of P-RBAC against other proposals. We start by pointing out major differences with respect EPAL. We then compare P-RBAC with the RBAC profile and the XACML privacy profile. We conclude the discussion with a brief comparison with two research proposals. We refer the reader to the extended version of this paper [21] for a more extensive discussion. Compared to EPAL, P-RBAC has the following major differences. First, one of the important design criteria of P-RBAC is to unify privacy policy enforcement and access control policy enforcement into one access control model. By contrast, EPAL is designed independently from any access control model. Second, the semantics of Roles in P-RBAC is different from that of User Categories in EPAL. User categories in EPAL represent categories of individuals that can access data and that are distinct from a privacy perspec-

tive; by contrast roles in RBAC represent job functionalities. Third, the first-applicable ruling adopted by EPAL during answering data access request is problematic[5], while P-RBAC supports conflict detection to guarantee that no conflicts arise in the set of PA, thus preventing the specification of unsafe privacy policies.

XACML [16] is a well known access control model based on XML. Its main goal is to provide an application independent policy language which enables the use of arbitrary attributes in different types of policies, including privacy policies. XACML aims at providing a flexible language for writing policies, but leaves the policy analysis task to policy analyzers. For example, XACML provides a very simple strategy to handle conflicts among rules. That is, when multiple rules in one policy yield different decisions for a same request, XACML will simply choose the decision from one rule according to the rule combining algorithm and ignore the effects of other rules. Although a hierarchical organization for resources and purposes is supported by XACML [17, 15, 18], there is no clear semantics for such hierarchies. Consequently, it is not clear how to solve conflicts introduced by hierarchies' usage.

In [13], the authors propose a Privacy-Aware Role-Based Access Control (PARBAC) model to enforce privacy policies within an organization which combines Chandramouli's DAFMAT framework [8] and privacy enforcement and management ideas from [14, 20]. However in PARBAC, there are many partial order relations, such as roles, tasks, purposes, and many-to-many relations, such as user-role, and subject-role. The semantics of these relations and the interactions between these relations is not formally defined, and therefore there are no consistency checking strategies. Moreover, there is no clear way to specify conditions in PARBAC.

Barth et al. [4] proposed a Linear Temporal Logic(LTL) based framework (CI) for expressing and reasoning about norms of transmission of personal information. However, CI cannot capture the notion of purposes directly. In order to capture this notion, CI has to decompose large agents (data provider, data user, and data receiver) into several smaller agents, one for each purpose, and purposes should then be expressed in CI by restricting the communication among the constituent agents. How to capture the notion of purpose concisely and the notion of purpose hierarchy seems to be a big barrier for CI.

4. CONCLUSIONS

In this paper, we have introduced a family of models for Privacy-aware Role Based Access Control and discussed the rationale behind these models. Formal definitions for Core P-RBAC and Hierarchical P-RBAC are given, conflicts between two P-RBAC permission assignments are discussed, and conflicts detection algorithms are presented.

However, there are a few interesting problems left open. First, we only discussed conflicts between two permission assignments; however, it is possible that there is no conflict between two permission assignments but there is a conflict between three or more permission assignments. Consider the following P-RBAC permissions:
PA_{31}: (BusinessPartner, ((Read, OrderInfo), Research, CurrentTime $\neq$ 9AM-5PM, $\emptyset$))
PA_{32}: (BusinessPartner, ((Read, OrderInfo), Research, CurrentTime $\neq$ 5PM-11PM, $\emptyset$))
PA_{33}: (BusinessPartner, ((Read, OrderInfo), Research, CurrentTime $\neq$ 11PM-9PM, $\emptyset$)).
Each two of them do not conflict with each other, but they together will not allow BusinessPartner to access order information. Second, there could exist redundant permission assignments in P-RBAC. For instance, PA_6 is redundant with respect to PA_7. Formal definitions of these two issues and their solution are our future work. We also plan to develop a prototype implementation for P-RBAC. Our long term goal is to provide a unified RBAC model that directly supports both security policies and privacy policies and consequently to investigate the interactions between security and privacy policies.

5. ACKNOWLEDGEMENT

The work reported in this paper has been partially supported by IBM under the OCR project "Privacy and Security Policy Management". Participants to this project are: Carnegie Mellon University, IBM T.J. Watson Research Center, Purdue University.

6. REFERENCES

[1] The enterprise privacy authorization language(epal 1.1). IBM Zurich Research Laboratory,Switzerland. Available at http://www.zurich.ibm.com/security /enterprise-privacy/epal/.

[2] Amazon.com. Amazon privacy notice. Available at http://www.amazon.com/exec/obidos/tg/browse/-/468496/102-8997954-0573735.

[3] A. H. Anderson. A comparison of two privacy policy languages: Epal and xacml. In *SWS '06: Proceedings of the 3rd ACM workshop on Secure web services*, pages 53–60, New York, NY, USA, 2006. ACM Press.

[4] A. Barth, A. Datta, J. C. Mitchell, and H. Nissenbaum. Privacy and contextual integrity: Framework and applications. In *SP '06: Proceedings of the 2006 IEEE Symposium on Security and Privacy (S&P'06)*, pages 184–198, Washington, DC, USA, 2006. IEEE Computer Society.

[5] A. Barth, J. C. Mitchell, and J. Rosenstein. Conflict and combination in privacy policy languages. In *WPES '04: Proceedings of the 2004 ACM workshop*

on Privacy in the electronic society, pages 45–46, New York, NY, USA, 2004. ACM Press.

[6] Blizzard.com. Blizzard entertainment online privacy policy. Available at http://www.blizzard.com/privacy.shtml.

[7] J.-W. Byun and N. Li. Purpose based access control for privacy protection in relational database systems. *The VLDB Journal The International Journal on Very Large Data Bases*, Sep 2006.

[8] R. Chandramouli. A framework for multiple authorization types in a healthcare application system. In *ACSAC '01: Proceedings of the 17th Annual Computer Security Applications Conference*, page 137, Washington, DC, USA, 2001. IEEE Computer Society.

[9] eBay.com. ebay privacy policy. Available at http://pages.ebay.com/help/policies/privacy-policy.html.

[10] Federal Trade Commision. Children's online privacy protection act of 1998. Available at http://www.cdt.org/legislation/105th/privacy/coppa.html.

[11] D. F. Ferraiolo, R. Sandhu, S. Gavrila, D. R. Kuhn, and R. Chandramouli. Proposed nist standard for role-based access control. *ACM Trans. Inf. Syst. Secur.*, 4(3):224–274, 2001.

[12] S. Fischer-Hubner. *IT-security and privacy: design and use of privacy-enhancing security mechanisms*. Springer-Verlag New York, Inc., New York, NY, USA, 2001.

[13] Q. He. Privacy enforcement with an extended role-based access control model. NCSU Computer Science Technical Report TR-2003-09, February 28, 2003.

[14] G. Karjoth and M. Schunter. A privacy policy model for enterprises. In *CSFW*, pages 271–281, 2002.

[15] OASIS. Core and hierarchical role based access control (rbac) profile of xacml v2.0. Available at http://www.oasis-open.org/.

[16] OASIS. extensible access control markup language (xacml) 2.0. Available at http://www.oasis-open.org/.

[17] OASIS. Hierarchical resource profile of xacml v2.0. Available at http://www.oasis-open.org/.

[18] OASIS. Privacy policy profile of xacml v2.0. Available at http://www.oasis-open.org/.

[19] Organisation for Economic Co-operation and Development. Oecd guidelines on the protection of privacy and transborder flows of personal data of 1980. Available at http://www.oecd.org/.

[20] C. S. Powers. Privacy promises, access control, and privacy management. In *ISEC '02: Proceedings of the Third International Symposium on Electronic Commerce*, page 13, Washington, DC, USA, 2002. IEEE Computer Society.

[21] E. B. J. L. Qun Ni, Alberto Trombetta. Privacy aware role-based access control. CERIAS Technical Report.

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

[23] C. Shankar and R. Campbell. A policy-based management framework for pervasive systems using axiomatized rule-actions. In *NCA '05: Proceedings of the Fourth IEEE International Symposium on Network Computing and Applications*, pages 255–258, Washington, DC, USA, 2005. IEEE Computer Society.

[24] TRUSTe.org. An independent, nonprofit enabling trust based on privacy for personal information on the internet. Available at http://www.truste.org/.

[25] United State Department of Health. Health insurance portability and accountability act of 1996. Available at http://www.hhs.gov/ocr/hipaa/.

[26] U.S. Senate Committee on Banking, Housing, and Urban Affairs. Information regarding the gramm-leach-bliley act of 1999. Available at http://banking.senate.gov/conf/.