PuRBAC: Purpose-Aware Role-Based Access Control

Amirreza Masoumzadeh and James B.D. Joshi

School of Information Sciences, University of Pittsburgh
{amirreza,jjoshi}@sis.pitt.edu

Abstract. Several researches in recent years have pointed out that for the proper enforcement of privacy policies within enterprise data handling practices the privacy requirements should be captured in access control systems. In this paper, we extend the role-based access control (RBAC) model to capture privacy requirements of an organization. The proposed purpose-aware RBAC extension treats purpose as a central entity in RBAC. The model assigns permissions to roles based on purpose related to privacy policies. Furthermore, the use of purpose as a separate entity reduces the complexity of policy administration by avoiding complex rules and applying entity assignments, coherent with the idea followed by RBAC. Our model also supports conditions (constraints and obligations) with clear semantics for enforcement, and leverages hybrid hierarchies for roles and purposes for enforcing fine grained purpose and role based access control to ensure privacy protection.

1 Introduction

Privacy can be defined as the right of individuals and organizations to control the collection, storage, and dissemination of information about themselves. Nowadays companies and enterprises gather more and more data about their users in order to provide more competitive services. This is especially true about applications on the Web that monitor the behavior of their users, resulting in heightened concern about potential disclosure and misuse of private information. Fortunately, the trend is such that organizations and enterprises are also becoming more serious about respecting the privacy of their customers. They not only are required to comply with existing privacy regulations, but also can take advantage of their privacy practices as an important capital to increase (or at least retain) their market share.

As described in a recently proposed road-map for web privacy by Antón et al. [1], there still remain vital research problems to be addressed. One major challenge is actual enforcement of privacy policies once the data has been collected. A big step towards enforcing privacy policies in an organization is considering them when making decisions over access to private data in information systems. With that vision, Powers et al. suggest privacy policy rules [2], comprising of *data type*, *operation* on the data, *data user*, *purpose* of data access, *condition* that restricts the accesses, and *obligations* that need to be carried out by the organization after the user is allowed to access.

R. Meersman and Z. Tari (Eds.): OTM 2008, Part II, LNCS 5332, pp. 1104–1121, 2008.

The well-known role-based access control model (RBAC) is a typical choice for organizational access control [3]. Therefore, enabling privacy policy specification within this model can be quite useful. Recently, Ni et al. propose P-RBAC [4], a privacy-aware role-based access control model, which incorporates notions of privacy policies defined in [2] into RBAC model. P-RBAC encapsulate data, action, purpose, condition, and obligation as privacy data permission. Although quite powerful, we argue that P-RBAC is moving away from the spirit of RBAC, that is simplicity of policy administration. With the use of roles as the intermediary entities between users and permissions, RBAC shifts from simply-represented but hard-to-manage paradigm that only consists of authorization rules, to a more manageable user-role and permission-role assignment scheme. However, privacy data permissions in P-RBAC do not consider that characteristic, and in presence of data and purpose hierarchies, the policy administration is as complex as authorization rule approaches such as [5].

We propose a slightly different extension to RBAC, called purpose-aware role-based access control (PuRBAC) model. In this model, we consider *purpose* as an intermediary entity between role and permission entities. The proposed PuRBAC model also includes constraints and obligations defined as conditions on assignment of permissions to purposes. We summarize the reasons for considering purpose as a separate entity in our model as follows:

- The core part of privacy policies usually state purposes for and circumstances under which collected data would be used, and the extent of use of personal information may differ based on the purpose of use. For example, health record of a job applicant may be used for the purpose of approval of qualification for a special job requiring certain degree of health, without disclosing details. However, the details may be used for the purpose of treatment of that person as a patient.
- There is a close relation between the notion of purpose in privacy policies and the notion of role in RBAC. A role in RBAC can be defined as a set of actions and responsibilities associated with a particular activity [6]. Purpose in privacy policies is defined as a reason for data collection and use [7], and business purposes can be identified through task definition within an organization's IT systems and applications [2]. Here, a close relation can be observed between what responsibility a user has (can be modeled as role) and its associated tasks for fulfilling the responsibilities (can be modeled as purpose). On the other hand, fulfilling the tasks requires access to data, which is represented by permissions in RBAC. Thus, considering purpose as intermediate entity between role and permission entities is intuitive.
- The other rational is to follow RBAC trend of breaking policy definition into different entities and relations between them (e.g. assignment relation between role and permission in RBAC), making management of different part of the policy as independent as possible. Unlike P-RBAC [4], treating purpose as a separate entity between role and permission makes our model coherent with that approach.

There is also a difference between our model and traditional access control models in that a subject should specifically assert the purpose of accessing data in its request to access a piece of data. In terms of RBAC, the access request can be treated as as a tuple containing a session identifier, purpose of access, and permission requested. Authorizing such a request ensures that a private information is accessed only for the valid purposes according to the privacy policy, without any ambiguities (in compliance with the use limitation principle as mentioned in [8]). Note that in practice the purpose assertion may be provided without user intervention by the application.

The rest of the paper is organized as follows. In Section 2, we introduce PuRBAC model; a base model is first defined formally (Section 2.1), followed by its hierarchy-extended (Section 2.2) and hybrid hierarchy-extended (Section 2.3) versions. In Section 3, our condition model is described separately from our access control model to simplify presentation. In Section 4, we discuss the strength of our approach and justify the our modeling. We review the previous major research in Section 5, and conclude the paper with a discussion of future directions in Section 6.

2 Purpose-Aware RBAC Model (PuRBAC)

We define a hierarchy of Purpose-aware RBAC (PuRBAC) models to clearly distinguish different features, following classic scheme of RBAC models [3]. Figure 1 shows the PuRBAC family of models. Figure 1.a illustrates the relationship between different models, and Figure 1.b shows different components and relations in the models. PuRBAC_B is the base model that specifies minimum required characteristics PuRBAC. PuRBAC_H extends PuRBAC_B with the notion of hierarchies of roles, purposes, and permissions. Although standard hierarchies are

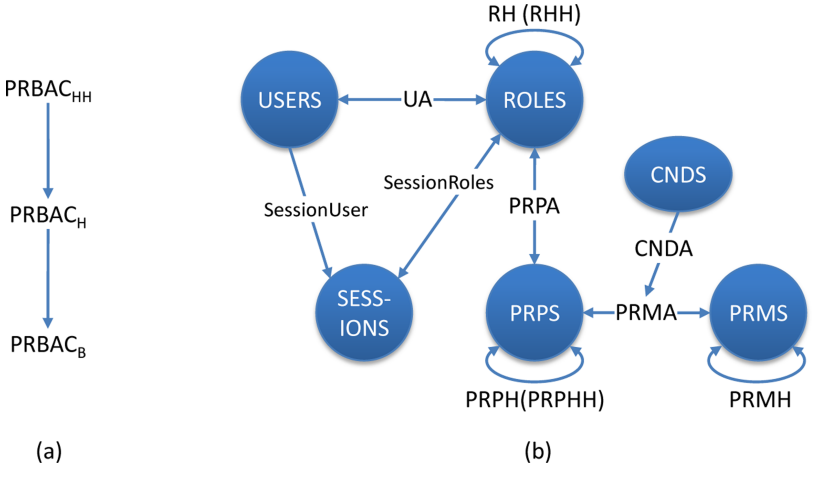

Fig. 1. Proposed Purpose-Aware Role-Based Access Control (PuRBAC) Model

considered a powerful property of RBAC models, they may introduce some risks because of conditional assignments that exist in the proposed PuRBAC model. To preclude such risks, $\text{PuRBAC}_{\text{HH}}$ extends PuRBAC_{H} with the notion of hybrid hierarchies [9] for roles and purposes.

In the following, we provide definition and semantics for each proposed model in the family.

2.1 Base Model: PuRBAC_{B}

Figure 1.b shows the overall structure of the proposed model. USERS, ROLES, and SESSIONS components and the relations among them are inherited from the standard RBAC [10]: individual users (USERS) are assigned to roles (ROLES), who can create sessions (SESSIONS) at runtime and activate assigned roles in the created sessions. The extension in PuRBAC is based on how permissions (PRMS) are exercised. We argue that permissions are exercised for a particular purpose; therefore, permissions are assigned to purposes (PRPS) for which they can be exercised, and then purposes are assigned to proper roles.

We adopt a similar approach to to the proposed NIST standard RBAC for permissions, in which permission represents a data type and corresponding action on that data. To enforce privacy policies in an enterprise data access system, such a model would be close to actual implementations. Whenever possible, we try to use the general notion of a permission.

As described before, privacy policies require flexible conditions for privilege management. To provide such flexibility, we enable conditions (CNDS) defined on permission assignments to purposes, that limit the assignment to particular cases or impose obligations. Here, we deliberately do not bring in an accurate definition for conditions to avoid complication in presentation of the model; though such a definition is independently provided in Section 3. For now, consider conditions as boolean predicates. An assignment is valid in a situation if and only if its condition is satisfied.

Formally speaking, the following sets and relations define the purpose-aware access control policy in PuRBAC_{B} model:

- $USERS$, $ROLES$, $PRPS$, and $DATA$; sets of users, roles, purposes, and data types, respectively.
- $PRMS$; set of pairs of the form $\langle d, a \rangle$ where $d \in DATA$ is a data type, and a is a valid action on that.
- $CNDS$; set of possible conditions.
- $UA \subseteq USERS \times ROLES$; user to role assignment relation.
- $PRPA \subseteq PRPS \times ROLES$; purpose to role assignment relation.
- $PRMA \subseteq PRMS \times PRPS$; permission to purpose assignment relation.
- $CNDA = CNDS \times PRMA$; condition to permission assignment binding relation.

Note that according to the definition of the $CNDA$ relation, there exists exactly one condition for each permission assignment. The execution semantics of

the model is as follows. As in standard RBAC, assigned roles to a user can be activated and deactivated by her in corresponding sessions; for a working user there is at least one session, which is unique to her. For exercising privacy-sensitive permissions, a user needs to provide a purpose; while only purposes assigned to the current active roles of a user can be asserted. The user is authorized to exercise permissions assigned to the provided purpose, given their conditions are satisfied. The following sets and functions capture the state of the system at runtime based on user interaction with the system:

- $SESSIONS$; set of sessions created by the users.
- $SessionUser : SESSIONS \rightarrow USERS$; mapping function from a session to its corresponding user.
- $SessionRoles : SESSIONS \rightarrow 2^{ROLES}$; mapping function from a session s to its active roles rs, where $rs \subseteq \{r | \langle SessionUser(s), r\rangle \in UA\}$.

Although the principal requesting access in an access scenario is the user, but in RBAC such requests are mediated through sessions. Therefore, we discuss authorizations for sessions in which authorizations are requested on behalf of a user. The following functions capture runtime authorization for role activation, purpose assertion, and conditional permission exercise in PuRBAC_B:

- $AuthRoles : SESSSION \rightarrow 2^{ROLES}$; mapping function from a session to the roles that can be activated in it. Formally: $AuthRoles(s : SESSIONS) = \{r \in ROLES | \langle SessionUser(s), r\rangle \in UA\}$.
- $AuthPurposes : SESSSION \rightarrow 2^{PRPS}$; mapping function from a session to the purposes that can be asserted for exercising permissions. Formally: $AuthPurposes(s : SESSIONS) = \{prp \in PRPS | \langle prp, r\rangle \in PRPA \ \wedge \ r \in SessionRoles(s)\}$.
- $CAuthPurposePermissions : PRPS \rightarrow 2^{CNDS \times PRMS}$; mapping function from a purpose to the conditional permissions that can be exercised through. Formally: $CAuthPurposePermissions(prp : PRPS) = \{\langle cnd, prm\rangle \in CNDS \times PRMS | \langle prm, prp\rangle \in PRMA \wedge \langle cnd, \langle prm, prp\rangle\rangle \in CNDA\}$.

The access control process of PuRBAC is different from classic access control models that only grant or deny the access request. At runtime, a user access request is submitted as a session, purpose, and requested permission. The access decision function (ADF) either determines a conditional authorization or responds with a denial decision (if no conditional authorization is resolved):

$$ADF(s : SESSIONS, prp : PRPS, prm : PRMS) =$$

$$\begin{cases} cnd & \text{if } \begin{array}{l} prp \in AuthPurposes(s) \\ \quad \wedge \exists \langle cnd, prm\rangle \in CAuthPurposePermissions(prp) \end{array} \\ \text{“deny”} & \text{otherwise} \end{cases}$$

If a conditional authorization is resolved by ADF, it will be passed to the access control enforcement function (AEF). The actual enforcement of condition by AEF is dependant on the condition model in use. However, generally it will check some constraints and enforce some obligations that eventually may result in granting or denying the access. We will provide enforcement details in Section 3.

2.2 Hierarchical Model: PuRBAC$_{\mathrm{H}}$

Hierarchies have been widely employed for propagation of authorization decision in access control models. Role hierarchy in standard RBAC allows senior roles to inherit permissions of junior roles. In PuRBAC$_{\mathrm{H}}$, senior roles inherit the purposes allowed for junior roles. Hierarchies are also useful for purpose based on generality/specificity concepts [11,12]. If a user can access an object for one purpose, she can also access that object for a more specific purpose. We also consider hierarchy for permissions, specifically based on data hierarchy (e.g. aggregation hierarchy) for the same actions; if a user is authorized for an action on one data type, she is also authorized for the same action on descendents of that data type.

PuRBAC$_{\mathrm{H}}$ considers hierarchies for roles, purposes, and permissions in the policy, defined as follows:

- $RH \subseteq ROLES \times ROLES$; a partial order relation on roles, denoted as $\geq_r$.
- $PRPH \subseteq PRPS \times PRPS$; a partial order relation on purposes, denoted as $\geq_{prp}$.
- $PRMH \subseteq PRMS \times PRMS$; a partial order relation on permissions, denoted as $\geq_{prm}$.

The existence of hierarchies impose some changes to the base model. In addition to directly assigned roles, a user can activate roles that are junior to the assigned roles. Because of role hierarchy the user can assert purposes assigned to not only her active roles, but also junior roles of her active roles. Moreover, because of purpose hierarchy the user can assert any more general purpose than she is entitled through role hierarchy.

Special care should also be taken when dealing with exercising permissions, which are conditionally assigned to purposes. Hierarchies in standard RBAC have only permissive behavior while in PuRBAC they can be both permissive and constraining. Since there are no condition on permission assignment in standard RBAC, inheritance of a previously owned permission by a role through hierarchy does not change the role permissions; therefore, the only effect of hierarchy is inheriting permissions assigned to junior roles by senior roles. In contrast, due to the existence of conditions on permission assignments in our model, if a hierarchical relation for a purpose leads to inheritance of a permission that was previously assigned to the purpose, there will be different conditional assignments for the same permission. In such a situation, there are two possible approaches for authorizing the permission for purpose: whether one of the conditions or all of them should be satisfied. We take the conservative approach and consider all the conditions applicable. Because the administrator can define more general privacy conditions at lower levels of the hierarchies, and be ensured that they are applicable to more fine-grained purposes and permissions. Hierarchical inheritance in this way constrains the permission by putting more conditions for the exercise. We show an example of such a scenario later in this section.

The following functions capture runtime authorization for role activation, purpose assertion, and conditionally assuming permissions in PuRBAC$_{\mathrm{H}}$:

- $AuthRoles : SESSSION \rightarrow 2^{ROLES}$; mapping function from a session to the roles that can be activated in it. Formally: $AuthRoles(s : SESSIONS) = \{r \in ROLES | \langle SessionUser(s), r' \rangle \in UA \ \wedge \ r \geq_r r'\}$.
- $AuthPurposes : SESSSION \rightarrow 2^{PRPS}$; mapping function from a session to the purposes that can be asserted for exercising permissions. Formally: $AuthPurposes(s : SESSIONS) = \{prp \in PRPS | \langle prp', r' \rangle \in PRPA \ \wedge \ prp \geq_{prp} prp' \ \wedge \ r \geq_r r' \ \wedge \ r \in SessionRoles(s)\}$.
- $CAuthPurposePermissions : PRPS \rightarrow 2^{CNDS \times PRMS}$; mapping function from a purpose to the conditional permissions which can be exercised through. Formally: $CAuthPurposePermissions(prp : PRPS) = \{\langle \prod cnd_i, prm \rangle \in CNDS \times PRMS | \langle prm', prp' \rangle \in PRMA \ \wedge \ prp \geq_{prp} prp' \ \wedge \ prm \geq_{prm} prm' \ \wedge \ \langle cnd_i, \langle prm', prp' \rangle \rangle \in CNDA\}$.

As described earlier in the case there are multiple applicable conditional assignments, all of them should be aggregated to be enforced; the term $\prod cnd_i$ refers to such an aggregation. The aggregation function itself is dependent to condition model in use. We provide the aggregation process of the conditions followed by our condition model in Section 3. Note that the access decision function ADF needs no change compared to the base model.

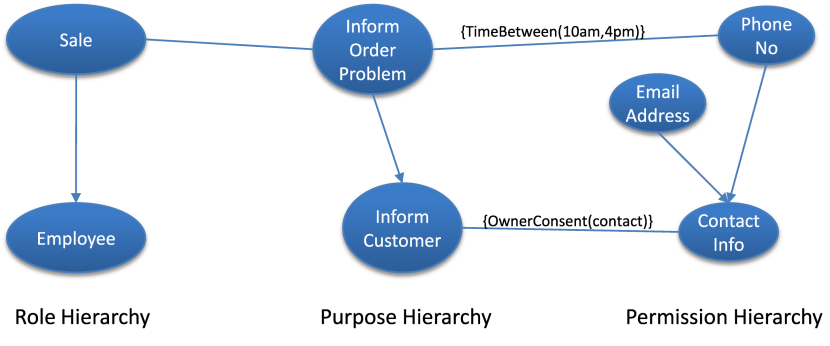

Fig. 2. Example Hierarchies and Their Assignments

For an example of access control process in PuRBAC_{H}, consider Figure 2 as partial role, purpose, and permission hierarchies in an online store system, along with their assignments. For simplicity, only the data types for permissions have been specified, and the action is *read* for all. The hierarchical relations are depicted with a directed line between entities. The role and purpose hierarchies are similar, e.g., role *sale* is senior to role *employee*, and purpose *inform order problem* is senior to (more specific than) purpose *inform customer*. The permission hierarchy is an aggregation hierarchy where the most coarse grained data is depicted at the bottom, e.g., permission to read *email address* is part of (more specific than) *contact info*. Data hierarchies are usually visualized in the opposite direction, but our visualization is aligned with the assignment inheritance, e.g., assignment of *contact info* to a role is also (indirectly) applied to *email*

address. Considering the mentioned relations, suppose that there is a problem with a customer's order and a sale employee wants to contact the customer. The employee first activates her *sale* role, which is assigned to the purpose *inform order problem*. For contacting the customer using email, she would request access to the customer's *email address*, asserting purpose *inform order problem* to the system. According to the policy setting, *contact information* can be accessed for purpose *inform customer* on the condition that owner of contact information has consented for contacting him using it. Based on the purpose and permission hierarchical relations, this authorization will be also applicable to the requested permission/purpose. Alternatively, the employee may call the customer by asserting purpose *inform order problem* to access customer's *phone number*. In that case there are two assignments applicable: there is a direct assignment between the requested purpose/permission with the condition of being daytime, and the previously mentioned assignment is also inherited through hierarchies with the condition of owner having consent. Therefore, the combined condition should be evaluated to true in order that access be granted to the employee.

2.3 Hybrid Hierarchical Model: PuRBAC$_{\mathbf{HH}}$

Hybrid hierarchy have been originally defined in the context of Generalized Temporal RBAC (GTRBAC) [13]. It separates the notion of permission inheritance and activation inheritance in role hierarchy, taking into account three types of relations: inheritance (I), activation (A), and inheritance-activation (IA). If r_1 is I-senior to r_2, it inherits the permissions of r_2. If r_1 is A-senior to r_2, any user assigned to r_1 can activate r_2, but the role r_1 does not inherit permissions of r_2. Finally, if r_1 is IA-senior to r_2, it inherits the r_2's permissions and also r_2 can be activated by anyone who can activate r_1. PuRBAC$_{\mathrm{HH}}$ leverages hybrid hierarchy for roles and purposes to provide more flexibility and overcome a weakness of PuRBAC$_{\mathrm{H}}$. Note that the semantics of hybrid hierarchy for purposes slightly differs from their semantic for roles in the way that purposes can be asserted instead of being activated.

One of the strengths of role hierarchy in RBAC is support for the principle of least privilege: a user is able to activate a junior role, which holds less permissions compared to a senior role, in the case she does not need the added permissions of the senior role for her current use of the system. PuRBAC$_{\mathrm{H}}$ model enables activating more junior roles in role hierarchy, and similarly asserting more general purposes in purpose hierarchy. Although purpose hierarchies are very similar to role hierarchies, asserting a more general purposes does not necessarily mean acquiring less privilege as it may even result in more privilege; if a user asserts a more general purpose than what she really intended for, she may have less restrictions for accessing some privacy-sensitive data. For instance, consider the previous example depicted in Figure 2. As mentioned before, if a sale employee asserts purpose *inform order problem* to access a customer's *phone number*, she is only allowed if the customer has consented before and it is daytime. But leveraging the hierarchical relation *inform order problem* $\geq_{prp}$ *inform customer*,

if the employee asserts purpose *inform customer*, she will be no longer restricted by the time constraint for accessing the phone number.

As described in previous section, the reason behind the possibility of such incident is that purpose hierarchies in our model can be either permissive or constraining, while role hierarchies in RBAC are permissive in nature. We can leverage the notion of hybrid hierarchies to overcome this issue, by allowing inheritance-only relation whenever we want to restrict the user's purpose assertions. For instance in the example above, if the relation between *inform customer* and *inform order shipment* is chosen as I-relation, user assigned to purpose *inform order shipment* will not be able to assert *inform customer* anymore, while the constraint for its corresponding assignment is still applied.

$\text{PuRBAC}_{\text{HH}}$ redefines role and purpose hierarchies in RBAC_{H} with the notion of hybrid hierarchies (permission hierarchy remains unchanged) as follows:

- RHH; hybrid hierarchy over roles includes three relations defined in $ROLES$: inheritance denoted as $\geq_{rI}$, activation denoted as $\geq_{rA}$, and inheritance-activation denoted as $\geq_{rIA}$. Note that $r_1 \geq_{rIA} r_2 \Leftrightarrow r_1 \geq_{rI} r_2 \wedge r_1 \geq_{rA} r_2$.
- $PRPHH$; hybrid hierarchy over purposes includes three relations defined in $PRPS$: inheritance denoted as $\geq_{prpI}$, assertion denoted as $\geq_{prpA}$, and inheritance-assertion denoted as $\geq_{prpIA}$. Note that $prp_1 \geq_{prpIA} prp_2 \Leftrightarrow prp_1 \geq_{prpI} prp_2 \wedge prp_1 \geq_{prpA} prp_2$.

The definitions for functions capturing runtime authorizations in PuRBAC_{H} are applicable to $\text{PuRBAC}_{\text{HH}}$ considering a few changes:

- In definition of $AuthRoles$, $\geq_r$ should be substituted with $\geq_{rA}$.
- In definition of $AuthPurposes$, $\geq_r$ and $\geq_{prp}$ should be substituted with $\geq_{rI}$ and $\geq_{prpA}$, respectively.
- In definition of $CAuthPurposePermissions$, $\geq_{prp}$ should be substituted with $\geq_{prpI}$.

3 Condition Model

Conditions that bind to permission assignments in PuRBAC have an important role in configuring access control policy to truly reflect the required privacy requirement. In Section 2, we defined a condition as a boolean predicate where the truth value affects the validity of corresponding permission assignment. In this section, we provide a more detailed approach for conditions defining their components and semantics.

Conditions can impose constraints on the assignment validity by the means of checking some data related to the accessed permission or any other access contexts. For instance, the consent of a data owner may be required to grant an access to the data. Also some data accesses may require that certain actions be properly pursued by the system or user before the access can be granted, referred to as pre-obligations. For instance, the policy may require the system to re-authenticate the user before authorizing access to highly privacy sensitive data such as social security numbers. In such a situation, if the user fails

to re-authenticate properly the permission can not be granted. Some other actions may be required to be carried out based on a data access after an access is granted, referred to as post-obligations. For example a data retention policy would schedule deletion of a data item in one year after its creation in data storage. Therefore, we model three types of conditions: constraints, pre-obligations, and post-obligations.

In order to provide the expected expressiveness for conditions, we define conditional constraints, pre-obligations, and post-obligations as follows:

- $CNDS$; conditions include conditional constraints, pre-obligations, and post-obligations. Formally: $CNDS = \{\mathcal{P}(PRDS \times CONS) \cup \mathcal{P}(PRDS \times PREOBGS) \cup \mathcal{P}(PRDS \times POSTOBGS)\}$, where
  - $PRDS$ and $CONS$ are sets of boolean predicates based on data variables in the system, representing general conditions and constraints, respectively.
  - $PREOBGS$ is the set of all valid pre-obligations in the system. Pre-obligations may require input parameters.
  - $POSTOBGS$ is the set of all valid post-obligations in the system. Post-obligations may require input parameters.

Semantically, constraints only query for data and provide a boolean result. Pre-obligations require the system or user to exercise some actions that may influence the access, while returning a boolean value that determines their proper enforcement. If any pre-obligation is not enforced properly, the access should be denied (and probably already-enforced pre-obligations be rollbacked). Post-obligations are enforced after the access is exercised. A special conditional predicate is available for post-obligations: *AccessGranted*, that can be used to control the dependency of enforcement of post-obligations on the result of access authorization. Figure 3 depicts how access control enforcement function (AEF) does the condition enforcement. The steps showing determination of constraints and

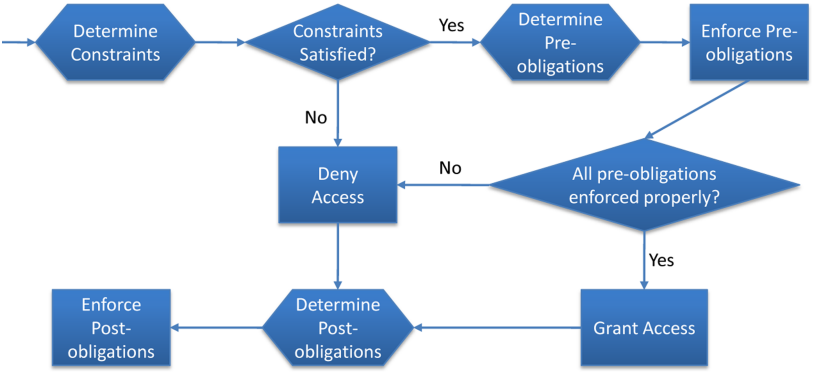

Fig. 3. Flowchart of Condition Enforcement by AEF

obligations check corresponding conditional predicates to decide which of them are applicable. Note that some post-obligations can be enforced even when the access is denied. Generally if a post-obligation does not check the conditional predicate *AccessGranted*, it will be enforced regardless of access decision. For instance, a post-obligation can log access attempt to an highly sensitive data item for operator review.

Different types of conditions can be defined and enforced based on our condition model. In the following examples d and a refer to the data instance being accessed and action on the data, respectively. Note that we use CN, PO, and OP for denoting conditional constraints, pre-obligations, and post-obligations, respectively. A category of important constraints in privacy policies is consent of data owner for collection or use of data. Such a constraint can simply be expressed as $CN(True, OwnerConsent(d, a) = True)$, where d and a refer to the data instance being accessed and type of access, respectively. However, in some situations a constraint is not applied for all instances of a data type. For example, Children Online Privacy Protection Act of 1998 (COPPA) [14] describes policies that are applicable to collection and processing of information about children under age 13. Here, there is a need to define conditional constraint whose condition is $OwnerAge(d) < 13$. In the case of COPPA, the consent of collection and use of data should be provided by a child's parent; therefore, the conditional constraint becomes $CN(OwnerAge(d) < 13, ParentalConsent(d, a) = True)$. Another category of conditions is data accuracy control which can be enforced through pre-obligations. For instance, suppose that a customer support wants to inform a buyer about the actual credit card that is billed, in response to an inquiry by the buyer. For such a purpose she needs only to verify the last 4 digits of the customer's credit card numbers. Such an accuracy can be controlled using an always-true pre-obligation on the permission assignment of accessing the credit card data field: $PO(True, FilterStringData(Length(d) - 4, Length(d)))$. In another example, suppose a child under age 13 wants to register in a Web site. The creation of a record for such a child fails in the first run because of a constraint similar to one mentioned previously. In such a situation, a post-obligation may seek parental consent if it hasn't been sought before: $OP(\neg AccessGranted \wedge ParentConsent = NA, AcquireParentalConsent(d, a))$. Other typical post-obligations for privacy policies are logging access or notifying some party such as the data owner about the access occurrence. Such post-obligations may be chosen to be enforced regardless of the access decision. The mentioned conditions do not constitute a complete list for possible conditions, but show a good variety of conditions for privacy control addressable by our model.

Besides support for different conditions, our model also supports aggregation of conditions. As mentioned in the previous section, in the case of PuRBAC_{H} and $\text{PuRBAC}_{\text{HH}}$ models, that multiple assignments may apply to an access request, all the bound conditions should be aggregated and applied to the access. Since conditions are sets of conditional constraint or obligations, aggregation of multiple conditions is the union of those conditions. For instance, consider the following conditions:

$c_1 = \{CN(True, OwnerConsent(d, a)),$
$OP(AccessGranted, SendOwnerNotification())\}$
$c_2 = \{PO(True, GetUserAcknowledgement()),$
$OP(Owner(d) \in MonitoredOwners, LogAccess())\}$
$c_3 = \{CN(True, OwnerConsent(d, a)),$
$PO(True, GetUserAcknowledgement()),$
$OP(Owner(d) \in MonitoredOwners, LogAccess()),$
$OP(AccessGranted, SendOwnerNotification()\}$

Here, condition c_1 includes a constraint (checking owner's consent) and a post-obligation (sending a notification to the data owner if access is granted). Condition c_2 imposes a pre-obligation (getting acknowledgement from user for accessing data) and a post-obligation (logging the access if the data owner is being monitored). Condition c_3 is the union of conditions c_1 and c_2, which includes all the mentioned constraints and obligations.

As conditions include obligations in addition to constraints, the unified set of conditions can result in inconsistencies or conflicts between obligations for enforcement. Moreover, enforcement of individual obligations may have inconsistencies or conflicts with the ongoing obligations in the system. In the case of two inconsistent obligations, the inconsistency can be resolved by overriding the one that can subsume the other. For instance, if an access results to a data retention post-obligation of one month, and there already exists a data retention obligation for one year, the one-month retention is enforced, discarding the other. But in the case of conflict, where none of the obligations can subsume the other, the system needs a meta-policy according to which it can determine which obligation should override, or possibly the access is being denied if no resolution is possible.

4 Analysis and Discussion

In this section we discuss strengths of the proposed model in Section 2 and some related issues.

4.1 Privacy Policy Management

Many privacy policies, particularly privacy acts and regulations, provide mandatory requirements for organizations to comply with. Those policies usually do not capture the internal system entities. Instead, they are focused on purposes and conditions that private data may be used. Therefore, the assignment of permissions on privacy-sensitive data to purposes along with the constraints can satisfactory model those privacy requirements; while authorization of roles for proper purposes complements the privacy policy.

The separation mentioned enables some independence for administration of the two assignment relations. Permission assignment should strictly consider

privacy regulations, for which the corresponding administrator should be completely informed and have clear understanding. However the latter part, purpose to role assignment, is more of an organizational issue and depend on roles' responsibilities, which possibly needs more modification over time compared to the former part. Therefore, we decrease the possibility of inadvertent manipulation of assignments privacy permission in the process of management of authorized organizational responsibilities and activities. Also, more flexibility for distributed administration is provided.

For instance, for the purpose *shipping order* in a company multiple permissions may be assigned to such as *read customer's shipping address* information. Suppose that shipping is done by sales department for some time, and hence purpose *shipping order* is assigned to role *sale*. Later if due to changes in organizational structure, shipment task is moved to the new shipping department, we need to change only the assignment of the purpose *shipping order* to the role *sale* to the new role *shipping*; there is no need to redefine the privacy permissions required for the purpose of shipping orders. In approaches such as P-RBAC [4], such a change requires manipulation of every privacy permission that is related to the purpose of *shipping order*.

In addition to the separation fact, a numerical comparison of possible assignments between our model and P-RBAC [4] can show the complexity differences. Note that we usually use term *assignment* for relating a pair of entities, and *rule* for relating multiple entities together (tuples). Therefore, assignments can also be considered as a tuple with two entities. The more the number of the rules a model deal with, the more complex it is to administer. For this comparison we exclude conditions (constraints and obligations), since they are present in both models and have similar influence on complexity. Suppose there are n roles, p purposes, and m permissions defined in a system. There can be as many as $n \times m \times p$ different authorization rules for roles in P-RBAC; while in our model we can have at most $p \times n$ assignments in $PRPA$ (purpose to role assignments) and $n \times p$ assignments in $PRMA$ (permission to purpose assignments), which sum up to $(n + m) \times p$ assignments. Considering that in practice the number of permissions (m) is much higher than the number of roles (n) in a system, our model can have about a factor of n less possible assignments, and hence is much less complex to administer.

4.2 Expressiveness Power

As described in the previous subsection, PuRBAC decreases the policy complexity by avoiding rules involving multiple components, namely role, purpose, and permission. Surely, such complexity decrease comes at the expense of some loss of expressiveness power. An expressiveness challenge may arise in a scenario that different roles with the same purpose can have different accesses. Consider the following scenario in a healthcare institute. The role *senior researcher* can access *complete profiles* of specific patients with the purpose *research*, with previous consent though. However, the role *research assistant* with the same purpose has only access to *limited profiles*. We believe that if purposes are defined

fine-grained enough in the system, there would not be an expression problem most of the time. For the mentioned scenario, although both accesses by roles *senior researcher* and *research assistant* have the purpose *research* at high level, they can be categorized into the more fine-grained purposes *complete research* and *limited research*, respectively.

If fine-grained purposes are not easy to define in a scenario, PuRBAC can cope with the issue by allowing predicates based on role in the conditional constraints on permission to purpose assignment. Therefore, permissions can be dynamically assigned to purposes based on the user's active roles. However, the use of such a role-based constraints should be restricted to special situations to keep the complexity advantage of PuRBAC.

4.3 Control over Purpose

Access control models that support privacy policies usually require the user to indicate the purpose of accessing information as one of the access request parameters. The indicated purpose is then used to check for compliance with policies in the system. The drawback in existing models is that users can indicate any purpose for information access without any restriction. Although the indicated purpose is checked against the policy, but that freedom makes system very vulnerable to misuse of data for purposes not really related to a role. That can happen with the existence of a simple error in the policy rules, that is not unlikely in practice considering the presence of role and purpose hierarchies.

In our model, the user cannot use data for a purpose without first having been authorized for that purpose. Such authorization is possible only for those purposes assigned to the user's currently active roles; and those assignment come from the fact that any role has a restricted set of responsibilities and functionalities which will define purposes for privacy-sensitive information access.

4.4 Role vs. Purpose

As mentioned in Section 1, notions of purpose and role can be very similar. The closeness and similarities between them tend to make it difficult to make distinction in some situations. For example, order processing in a store can be modeled as a specific role compared to widely scoped roles (as described in [15]), as well as a purpose for accessing customer record (as described in [11]). We take advantage of this tight relation to justify their direct assignment in our model, and argue against considering them both as (different type of) roles. Roles are usually derived based on organizational positions and responsibilities. But purposes have no relation to organizational structure, but to functions.

Someone may argue that the role entity itself can support both notions of role and purpose in our model, by having structural and functional roles, respectively. In such an approach, the model needs to deal differently with those two role types by allowing (*i*) assignment of permissions only to functional roles, (*ii*) inherence of functional roles by structural roles, and (*iii*) assignment of users only to structural roles. Moreover, a major difference of authorization model of

PuRBAC, compared to RBAC, is assertion of purpose as a part of access request. That feature is not supported in standard RBAC, as access check is only based on session and requested permission [10]. Therefore, the access checking function in the standard needs to be changed to be aware of the access purpose (functional role). Considering the mentioned differences and also the administration independence for role and purpose hierarchies in our model, we believe that there is enough motivation to consider purpose as a separate entity from role.

4.5 Sticky Policies

A very flexible approach for privacy policies would be the sticky policy paradigm [16]. In that approach, policies are defined and maintained for each data instance. Therefore, privacy policies can be different for different instances of the same data type. Although quite promising, we argue that it is less probable to be followed by organizations. The main drawback is that the organization would loose its centralized control over access control policies once the policy is stuck to the data. That is not preferred since the access control policy may require changes due to revision of high-level policies, or frequent improvement of the access control policy itself. Moreover, the storage and processing of access control policies will be very expensive in the case of using sticky policy approach.

5 Related Work

Enforcement of privacy policies have been studied by several researchers. Karjoth et al. propose Enterprise Privacy Architecture (EPA) as a methodology for enterprises to provide privacy-enabled service to their customers [17]. In [2], Powers et al. investigate privacy concerns from organizations' point of view and existing technologies, and describe an approach for enterprise-wide privacy management. They suggest expressing privacy policy in terms of privacy rules comprising of data type, operation on it, data user, purpose of data access, condition that restricts that access, and obligatory actions taken by the user when she is authorized. The Platform for Enterprise Privacy Practices (E-P3P) policy language, proposed in [5], contains authorization rules. In addition to mentioned components of the privacy policy, each authorization rule in E-P3P contains a parameter that indicates if the rule is either positive or negative authorization, and a precedence value. Tree hierarchies for data categories, users, and purposes are also considered. Another similar work has been done by Karjoth et al. that extends Jajodia's Authorization Specification Language (ASL) [18], to include obligations and user consent [11]. They also discuss a solution to automatically translate inner-enterprise privacy policy stated using E-P3P to publishable P3P policies for customers [19]. The language has been formalized and refined to form IBM Enterprise Privacy Authorization Language (EPAL) [20].

Ni et al. propose P-RBAC [4], a privacy-aware role-based access control model, which incorporates privacy policies into RBAC [3]. They encapsulate data, action, purpose, condition, and obligation as privacy data permission. A permission assignment in P-RBAC is an assignment of a privacy data permission

to a role (equivalent to data user in [2]). Also, more complex conditions have been considered in a conditional version of P-RBAC [21]. In previous sections, especially Section 4, we described and analyzed the advantages of our approach to modeling purposes compared to P-RBAC. From condition model perspective, compared to P-RBAC the support of conditional constraint and obligations in our model enables more concise privacy policy definition. P-RBAC requires specifying explicitly all the conditions when a privacy-sensitive data can be accessed (the notion of condition in P-RBAC is close to constraint in our model). However, conditional constraints in our model allows enforcement of constraints on permissions only when some conditional predicates are met. P-RBAC also lacks clear semantics for enforcement of obligations; our model provides clear semantics for the enforcement process flow of constraints, pre-obligation, and post-obligations. Ni et al. have proposed an obligation model extension for P-RBAC very recently, which includes the notion of conditional obligation (but not conditional constraints), and deals with temporal obligations and obligation dominance [22].

Byun et al. address the notion of purpose-based access control [12], seeking compliance between intended (allowed or prohibited) purposes defined for data and access purposes requested by users at runtime. The strength of their approach is dealing with purpose in complex hierarchical data management systems. However, the approach seems too complicated to be used as a general purpose access control model.

6 Conclusions

We proposed purpose-aware role-based access control (PuRBAC) model as a natural RBAC extension to include privacy policies. In PuRBAC, purpose is defined as an intermediary entity between role and permission. Users can only exercise permissions assigned to an asserted purpose, which itself should be authorized through assignment to the user's active roles. Also, assignments of permissions to purposes are bound with conditions that constrain the assignment validity or impose obligations. We also defined a general model of conditions, providing the enforcement semantics for conditional constraints, pre-obligations, and post-obligations.

The introduction of purpose as a separate high-level entity in RBAC requires further analysis of its impact on other aspects of RBAC paradigm such as separation of duty constraints, policy administration model, etc. Moreover, the assertion of purpose in the access control process needs to be studied in more detail from usability and accountability perspectives. The ultimate goal might be to enable the system intelligently identify the purpose of data access according to the user's tasks, while ensuring the user accountability.

Acknowledgements. This research has been supported by the US National Science Foundation award IIS-0545912. We would like to thank the anonymous reviewers for their helpful comments.

References

1. Antón, A.I., Bertino, E., Li, N., Yu, T.: A roadmap for comprehensive online privacy policy management. Communications of the ACM 50(7), 109–116 (2007)
2. Powers, C., Ashley, P., Schunter, M.: Privacy promises, access control, and privacy management: Enforcing privacy throughout an enterprise by extending access control. In: Proc. 3rd International Symposium on Electronic Commerce, October 18-19, 2002, pp. 13–21 (2002)
3. Sandhu, R.S., Coyne, E.J., Feinstein, H.L., Youman, C.E.: Role-based access control models. IEEE Computer 29(2), 38–47 (1996)
4. Ni, Q., Trombetta, A., Bertino, E., Lobo, J.: Privacy-aware role based access control. In: Proc. 12th ACM symposium on Access control models and technologies, pp. 41–50. ACM Press, New York (2007)
5. Ashley, P., Hada, S., Karjoth, G., Schunter, M.: E-P3P privacy policies and privacy authorization. In: Proc. ACM workshop on Privacy in the Electronic Society, pp. 103–109. ACM, New York (2002)
6. Sandhu, R.S., Samarati, P.: Access control: Principles and practice. IEEE Communications Magazine 32(9), 40–48 (1994)
7. Cranor, L., Langheinrich, M., Marchiori, M., Presler-Marshall, M., Reagle, J.: The platform for privacy preferences 1.0 specification. Technical report, W3C (2002)
8. OECD: Oecd guidelines on the protection of privacy and transborder flows of personal data (1980), `http://www.oecd.org/document/18/0,3343,en_2649_34255_1815186_1_1_1_1,00.html`
9. Joshi, J.B.D., Bertino, E., Ghafoor, A., Zhang, Y.: Formal foundations for hybrid hierarchies in gtrbac. ACM Transactions on Information and System Security 10(4), 1–39 (2008)
10. Ferraiolo, D., Kuhn, D.R., Chandramouli, R.: Role-based access control. Artech House computer security series. Artech House, Boston (2003)
11. Karjoth, G., Schunter, M.: A privacy policy model for enterprises. In: Proc. 15th IEEE Computer Security Foundations Workshop, June 24-26, 2002, pp. 271–281 (2002)
12. Byun, J.W., Bertino, E., Li, N.: Purpose based access control of complex data for privacy protection. In: Proc. 10th ACM symposium on Access control models and technologies, pp. 102–110. ACM Press, New York (2005)
13. Joshi, J., Bertino, E., Ghafoor, A.: Hybrid role hierarchy for generalized temporal role based access control model. In: Proc. 26th Annual International Computer Software and Applications Conference (COMPSAC), August 26-29, 2002, pp. 951–956 (2002)
14. FTC: Children's online privacy protection act of 1998 (coppa) (1998), `http://www.ftc.gov/ogc/coppa1.htm`
15. Samarati, P., De Capitani di Vimercati, S.: Access control: Policies, models, and mechanisms. In: Focardi, R., Gorrieri, R. (eds.) FOSAD 2000. LNCS, vol. 2171, pp. 137–196. Springer, Heidelberg (2001)
16. Karjoth, G., Schunter, M., Waidner, M.: Platform for enterprise privacy practices: Privacy-enabled management of customer data. In: Dingledine, R., Syverson, P.F. (eds.) PET 2002. LNCS, vol. 2482, pp. 69–84. Springer, Heidelberg (2003)
17. Karjoth, G., Schunter, M., Waidner, M.: Privacy-enabled services for enterprises. In: Proc. 13th International Workshop on Database and Expert Systems Applications, September 2-6, 2002, pp. 483–487 (2002)

18. Jajodia, S., Samarati, P., Sapino, M.L., Subrahmanian, V.S.: Flexible support for multiple access control policies. ACM Transactions on Database Systems 26(2), 214–260 (2001)
19. Karjoth, G., Schunter, M., Van Herreweghen, E.: Translating privacy practices into privacy promises: how to promise what you can keep. In: Proc. 4th IEEE International Workshop on Policies for Distributed Systems and Networks, June 4-6, 2003, pp. 135–146 (2003)
20. IBM: The enterprise privacy authorization language, `http://www.zurich.ibm.com/security/enterprise-privacy/epal/`
21. Ni, Q., Lin, D., Bertino, E., Lobo, J.: Conditional privacy-aware role based access control. In: Biskup, J., López, J. (eds.) ESORICS 2007. LNCS, vol. 4734, pp. 72–89. Springer, Heidelberg (2007)
22. Ni, Q., Bertino, E., Lobo, J.: An obligation model bridging access control policies and privacy policies. In: Proc. 13th ACM symposium on Access control models and technologies, pp. 133–142. ACM, New York (2008)