An Obligation Model Bridging Access Control Policies and Privacy Policies

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

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

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

ABSTRACT

In this paper, we present a novel obligation model for the Core Privacy-aware Role Based Access Control (P-RBAC), and discuss some design issues in detail. Pre-obligations, post-obligations, conditional obligations, and repeating obligations are supported by the obligation model. Interaction between permissions and obligations is discussed, and efficient algorithms are provided to detect undesired effects. Core P-RBAC is extended to support both access control policies and privacy policies simultaneously. We believe that a full-fledged obligation solution based on RBAC may have a great potential because it could be easily deployed in systems already adopting RBAC and would thus allow one to seamlessly introduce policies with obligation requirements, either for access control purposes or for privacy purposes.

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

Obligation, Role Based Access Control, Privacy, Policy

1. INTRODUCTION

Access control policies are widely used for controlling access to sensitive information and valuable resources in various environments. Privacy policies are specifically designed to protect privacy when collecting, using, and disclosing personal identifiable information. Both access control policies and privacy policies mainly focus on the specification and management of access control requirements, that is, which subjects are allowed to access which objects and when, how, and why. In addition to requirements concerning access control, modern corporation regulations, operation rules, and privacy laws often impose obligation requirements specifying some other actions that *must* be performed *sometime* in order to allow a certain action to be executed *now*. For instance, COPPA [12] requires that "Before collecting, using or disclosing personal information from a child, an operator must obtain verifiable parental consent from the child's parent.".

Because traditional access control policy languages cannot express these obligation requirements, these requirements are often hard-coded in policy enforcement engines or even in applications. Obviously such approach is not the best for enforcing obligation requirements. First, obligations end up being expressed as code and therefore there is no high-level specification of the obligations actually enforced. Second, it lacks flexibility. For instance, if obligations have to be changed, perhaps because of errors, the policy enforcement engine or the application has to be modified. Since the main benefit of a policy-based access control management system is its flexibility in responding to changes without requiring modifications to systems or applications, why not supporting the obligation in policy languages? A policy-based approach to obligations could allow an organization to quickly change obligation requirements when flaws in existing policies are found, to promptly react to new circumstances, and to accurately enforce complex obligation requirements. In recent years, several policy languages have been proposed to support obligations [22, 16, 10, 30, 19, 21]. Approaches have also been developed for monitoring the fulfillment of obligations [6, 5, 17].

The introduction of obligation in policies is not however as straightforward as it looks like at first glance. First, unlike traditional policies, obligations usually have a time interval within which they have to be fulfilled. Some obligations must be fulfilled before a certain action, referred to as *pre-obligations*, and others must be fulfilled after the action, referred to as *post-obligations*. There are cases in which an obligation has to be repeated. For instance, the GLB act requires that "Customers must receive a notice every year for as long as the customer relationship lasts." Therefore, a flexible temporal constraint model is a necessary component. Second, some obligations are conditional. For instance, "as long as the customer relationship lasts" is a condition for a financial organization to send a notice to its customers. Third, obligations are actions. In order to fulfill an obliga-

SACMAT'08 June 11-13, Estes Park, CO 80517

tion, some privileges are required as well. If the privileges needed are not available, such obligation is not fulfillable. The problem can become more complex because policies related to these privileges may further require execution of other obligations. Therefore, techniques used for analyzing the interactions between obligations and policies should be devised in order to detect invalid policies due to unfulfillable obligations.

In this paper, we propose a novel obligation model that satisfies the aforementioned requirements. The model is part of the Privacy-aware Role Based Access Control (P-RBAC) models [21, 20], which extend the Role-based Access Control Models [26, 13] to support privacy policies. Core P-RBAC, the base model of the P-RBAC model family, is characterized by a good balance between expressivity and complexity, which in turn provides an ideal platform to support a new obligation model. Furthermore, we believe that a full-fledged obligation solution based on RBAC may have a great potential because it could be easily deployed in systems already adopting RBAC and would thus allow one to seamlessly introduce policies with obligation requirements either for access control purposes or for privacy purposes. Major contributions of this paper are as follows.

- A terse yet highly expressive obligation model that supports pre-obligations, post-obligations, conditional obligations, and repeating obligations.
- A natural extension to Core P-RBAC to support both access control policies and privacy policies simultaneously.
- The seamless integration of the new obligation model and RBAC permissions without destroying the spirit of RBAC where no direct user-permission relation exists.
- The investigation of the interaction between permissions and obligations, and the development of efficient algorithms for detecting undesired effects, like infinite *obligation cascading*, resulting from this interaction.
- The introduction of the concept of the coverage of obligations, and its application to reduce the number of obligations returned for the fulfillment on an access request.

The rest of this paper is organized as follows. Section 2 discusses related work. Section 3 briefly introduces P-RBAC, focusing on Core P-RBAC which is the base of the work presented in this paper. Section 4 presents the new obligation model and discusses in detail relevant requirements for an obligation model and design choices. Section 5 focuses on two interesting issues in policy analysis and presents efficient solutions to these issues. Section 6 concludes the paper and outlines future research directions.

2. RELATED WORK

Several approaches to model and analyze obligations have been recently proposed [7, 8, 6, 15, 17, 11].

Bettini et al. [7, 8] have formalized policies as Horn clauses with additional provision and obligation formulas and investigated the problem of choosing the best policy rules to minimize the provisions and obligations that a user has to fulfill in order to execute certain actions. Bettini et al. [6] have further extended their policy model to express the handling of obligation violations. However, predicates on action, provisions, and obligations [7, 8] are disjoint sets that break the natural connection among them. Moreover, their approach lacks a mechanism to specify fine-grained temporal constraints and can thus only model a limited set of obligations.

Hilty et al. [15] have proposed a formal model for data protection policies using Distributed Temporal Logics. Provisional formulas that contain no future time temporal operators represent provisions and obligational formulas that contain no past time temporal operators represent obligations. They have investigated the observability of obligations, that is, the existence of evidence that the reference monitor has been informed about the fulfillment of obligations, and discussed several possible ways of transforming non-observable obligations into observable obligations. Our work investigates a different problem, that is, the undesirable interactions between permissions and obligations. In some cases, obligations in a policy cannot be fulfilled given the current permissions, e.g., the subject has no permission to perform the obligation, the permission conditions contradict the obligation conditions, or cascading obligations occur. Moreover, we also carefully investigate the timeline restrictions concerning pre-obligations and post-obligations, while Hilty et al. address this topic by a too simplistic approach [15].

Irwin et al. [17] have modeled the notion of obligation as a tuple of subject, action, objects, and a time window. They have defined system states based on obligations and time ticks, and investigated the obligation accountability problem based on different assumptions. However, their approach is limited by their less expressive obligation model. It does not support pre-obligations, repeating obligations, and conditional obligations, which are required by privacy acts and policies by financial institutions.

Barth et al. [2] have formulated privacy policies as Linear Temporal Logic formulas and have addressed policy compliance and refinement by translating these notions into the logical concepts of satisfiability and entailment. The distinct feature of their work is the focus on the transmission of personal identifiable information (PII) instead of the access control for PII. Moreover they have investigated the problem from a very general view (a top down view) that inevitably makes their solutions very complex, that is, with a PSPACE complexity or even worse. Our approach instead focuses on the access control for PII and adopts a bottom up approach by identifying the most necessary and basic components to express privacy policies first and gradually expanding the model to better capture the semantics of privacy laws while keeping policy analysis tractable.

Dougherty et al. [11] have presented an abstract model treating obligations as constraints on the program execution path. An obligation is considered fullfilled if every run of the program satisfies the constraint, and the obligation is violated if no run of the program satisfies the constraint. The model further supports an independent state space for obligations that is separated from program states in order to handle issues resulting from the interaction between programs and obligations. Static analysis and obligation monitoring have been handled by standard algorithms using Büshi automata. In constrast, we propose a concrete obligation model based on RBAC, and we develop an analysis for obligations which is efficient under reasonable assumptions.

Compared to our work, the model by Dougherty et al. [11] cannot express pre-obligations nor conditional obligations, and the analysis of policies it can express can be very expensive due to its general assumptions. Further, enforcing policies, especially obligations, in modern operating systems and database systems may be challenging because it is not straightforward to translate constraints on execution paths into executable policies or vice versa.

Gama et al. [14] have proposed a prototype obligation monitoring framework which tracks the fulfillment of pending obligations. Sailer et al.[24] have discussed a method that allows a third party to monitor obligation compliance in web services context. Skene et al. [27] have proposed a model and an analysis technique for reasoning about the monitorability of systems of service level agreements (SLAs) that is closely related to the monitorability of obligations. Our paper, on the other hand, formally defines a more expressive obligation model based on P-RBAC, an extension to a widely accepted access control model, and investigates the interaction between actions and obligations. Therefore, our work is complementary to aforementioned approaches.

Bertino et al. [3] have proposed an access control model in which periodic temporal intervals are associated with authorizations. An authorization is automatically granted in the specified intervals and revoked when such intervals expire. Deductive temporal rules with periodicity and order constraints are provided to derive new authorizations based on the presence or absence of other authorizations in specific periods of time. Later, Bertino et al. [4] have extended RBAC to support periodic role enabling and disabling and temporal dependencies among actions, which is the base of a Generalized Temporal Role-Based Access Control Model [18] that further enables a wider range of temporal constraints, e.g. periodic as well as duration constraints on roles, user-role assignments, and role-permission assignments. Compared to those temporal constraint models, the temporal constraint model introduced in this paper has a different goal, that is, to investigate properties of time restrictions with respect to the fulfillment of obligations which requires to look at temporal constraints from a different angle.

Prakken et al. [23] have investigated the proper representation of contrary-to-duty structures in deontic logics, and situations in which there is a primary obligation and a secondary obligation, which comes into effect when the primary obligation is violated. Brown et al. [9] have investigated the semantic treatment of conditional obligations, permissions, and prohibitions based on models with agents and branched time. In general, these theoretical approaches investigate some interesting behaviors of obligations, permissions and their relations and improve deontic logics to better represent them and reason on them. The problems investigated by those approaches and ideas that partially solve these problems can be incorporated into future extensions to the approach reported in our paper.

Last but not least, obligations that have been introduced in some modern policy languages, such as XACML [22], EPAL [16], KAoS [30], Ponder [10] and Rei [19], are rather limited. XACML, EPAL, and KAoS only support system obligations because no other subject can be indicated in their obligation language. Although Ponder and Rei enable user obligations, all of aforementioned languages do not provide an explicit placeholder supporting the specification of temporal constraints, and they do not support pre-obligations, conditional obligations, repeating obligations either. Some policy algebras [1, 25] supporting obligations have been proposed for composing enterprise privacy policies like EPAL. Unfortunately, obligation models in such algebras suffer from the same drawbacks of EPAL.

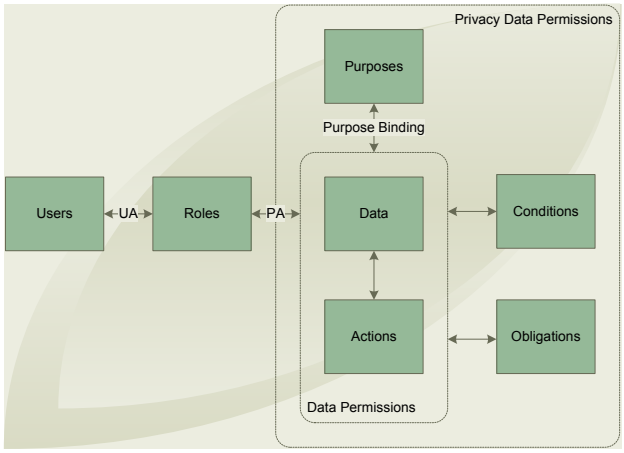

Figure 1: Core P-RBAC model

3. A BRIEF INTRODUCTION ABOUT P-RBAC

Because the obligation model presented in this paper is based on P-RBAC, in this section we briefly introduce relevant concepts about P-RBAC. P-RBAC is a family of Privacy-aware RBAC models that extend RBAC with support for privacy policies [21]. Core P-RBAC, the base model, is at bottom. There is a tradeoff between expressivity and complexity in the design of Core P-RBAC. On the one hand, Core P-RBAC has limited expressive power which is, however, sufficient for representing public privacy policies, privacy statements and privacy notices in Web sites, and policies based on privacy related acts, such as HIPPA [28], COPPA [12], and GLBA [29], in the US. On the other hand, conflicts detection in Core P-RBAC remains tractable. Advanced models in the family extend Core P-RBAC with additional modeling constructs.

Core P-RBAC is illustrated in Figure 1. The model includes several sets of entities: Users(U), Roles(R), Data(D), Actions(A), Purposes(Pu), Obligations(Ob), 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 related 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 are implemented.

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 pri-

vacy rules. The model captures 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 permission can be given, and the obligations that are to be finally performed if permission is granted.

Core P-RBAC conditions should not be confused with the constraints of the classic RBAC model. Constraints are a powerful mechanism for specifying higher-level organizational policies that cross over several roles, while conditions are a mechanism to precisely define a permission of a single role. 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.

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 1. [21] Let CV be a set of context variables; each variable $\mathsf{x} \in \mathsf{CV}$ has a finite domain of possible values, denoted as D_x; every domain is equipped with a pair of corresponding relational operators $=$ and $\neq$. $\perp$(false) and $\top$(true) are constant conditions. An atomic condition $\mathfrak{ac}$ defined over CV has the form $(\mathsf{x}\ \mathsf{op}_\mathsf{r}\ \mathsf{v})$ where $\mathsf{x} \in \mathsf{CV}$, $\mathsf{v} \in \mathsf{D}_\mathsf{x}$, $\mathsf{op}_\mathsf{r} \in \{=, \neq\}$. The conditions of LC_0 (over CV) are defined as follows:

- A constant condition is a condition of LC_0.
- An atomic condition is a condition of LC_0.
- Let c_i and c_j be conditions of LC_0. $\mathsf{c}_\mathsf{i} \wedge \mathsf{c}_\mathsf{j}$ is a condition of LC_0. □

Two typical context variables and their domains are as follows:

- oc: Owner Consent, domain={yes, no, na}; it represents data owner's consent for collecting, using, or disclosing his personal identifiable information.
- vpc: Verifiable Parental Consent, domain={yes, no, na}; it represents a parent's consent for collecting, using, or disclosing his child information.

Because the set of context variables is finite, each context variable has a finite domain, and only equality and inequality operators are supported, it is straightforward to see that the set of all possible conditions (module logical equivalence) that can be expressed in LC_0 is finite. We denote such set as C.

4. AN OBLIGATION MODEL

Typical obligations in privacy policies specify what actions a subject must perform at certain time in order to allow certain actions to be taken at present. Before presenting our obligation model, we first investigate the usage of obligations in privacy policies using a few case studies. We design our obligation model based on the analysis of these scenarios.

4.1 Desiderata

One of the regulations in COPPA requires that "Before collecting, using or disclosing personal information from a child, an operator must obtain verifiable parental consent from the child's parent. This means that an operator must make reasonable efforts (taking into consideration available technology) to ensure that before personal information is collected from a child, a parent of the child receives notice of the operator's information practices and consents to those practices." Thus, an obligation, "notifying a parent and obtaining verifiable parental consent", may have to be fulfilled before access to the children information. Another interesting point here is that we may need a conditional obligation. Once we have fulfilled the obligations and obtained some results, either consent or rejection, from a specific parent, we usually should not ask the parent the same question again. Therefore, before fulfilling the obligation, we may want to check whether we have already asked the question. Another example that requires a conditional obligation is: "An operator is required to send a new notice and request for consent to parents if there are material changes in the collection, use or disclosure practices to which the parent had previously agreed." Sending a new notice and requesting for consent can be considered as conditional obligations after collecting children information.

COPPA also says that "At any time, a parent may revoke his/her consent, refuse to allow an operator to further use or collect their child's personal information, and direct the operator to delete the information.". Our understanding of this statement is that once operators obtain parental consent and collect children information, they should immediately assign parents permissions such as "revoke consent", "refuse further use or collection", or "request deletion". One way to specify those permissions in formal policies is to consider them as obligations that should be fulfilled without delay after children information collection. Therefore, obligations may include actions like "grant permissions", "revoke permissions" and "delete data".

The GLB act says that "Consumers are entitled to receive a privacy notice from a financial institution only if the company shares the consumers' information with companies not affiliated with it, with some exceptions. Customers must receive a notice every year for as long as the customer relationship lasts.". An obligation, "sending consumers a privacy notice", should be fulfilled after the first time consumers' information is disclosed. Another obligation, "sending customers privacy notices", must be fulfilled periodically. Moreover, the latter, "sending customers privacy notices", seems not to be related to any action, e.g. collecting, using, and disclosing, on customers' information. However, such an obligation is related to an attribute of information providers. The attribute specifies that these providers are not only consumers, but also customers[1]. Therefore, the latter obligation is actually related to an action that changes a consumer to a customer.

Based on aforementioned cases of obligations, we summarize the following features that we consider relevant for obligations.

[1]In the GLB act, a consumer is an individual who obtains or has obtained a financial product or service from a financial institution for personal, family or household reasons. A customer is a consumer with a continuing relationship with a financial institution.

- Generally, obligations are associated with some action request[2], i.e., a subject promises to fulfill some obligations sometime in order to perform a specific action on some objects now. There are cases in which specific obligations are only associated with some special objects in the policies without reference to an action. However, a corresponding action can still be identified in practice because usually the action making these objects special is the action causing these obligations.

- Obligations have usually some specific temporal constraints. Some obligations should be fulfilled before an access is allowed and the result from the obligation fulfillment may affect the decision about an action request. We call this kind of obligations *pre-obligations*. Other obligations should be fulfilled after the action in the action request is performed. We call this kind of obligations *post-obligations*. Intuitively, there should be some time window allocated for each obligation. Otherwise, a policy enforcement engine does not know when it can start evaluating policy conditions, and subjects in a post-obligation can legally avoid obligations by simply saying "I will do it in the future". In some cases, temporal constraints require obligations to be fulfilled repeatedly until some condition becomes true.

- A subject's obligation may result from another subject's action, i.e., the subject of an obligation may be different from the subject who caused the obligation. For instance, when an operator discloses some children information to third parties, third parties may be required to fulfill similar obligations the operator has to fulfill. In some situations, the subject of an obligation may be the system itself, e.g., logging access history.

- Some obligations may be conditional, that is, conditional obligations are only required to be fulfilled if some related condition becomes true. For instance, COPPA says that "An operator is required to send a new notice and request for consent to parents if there are material changes in the collection, use or disclosure practices to which the parent had previously agreed.". Here, the material changes are the conditions that trigger the execution of the obligations "send a new notice" and "request for consent".

4.2 The Model

In this section, we present a formal obligation model for privacy policies that encompasses the features we discussed in the previous section. The model serves as the theoretical foundation for our later discussion on the analysis of obligations. Since obligations are actions that some subjects have to fulfill during some time interval, the obligation model introduces a temporal constraint component that clearly specifies such a time interval. As mentioned in the previous section, we are striving for a simple yet flexible mechanism to specify these temporal constraints. Two important requirements for such mechanism are that it should be able to support the specification of the most common temporal constraints and an efficient analysis of these constraints.

As we mentioned in the previous section, the initiator of an obligation may differ from the user who causes the obligation; therefore a component used to indicate the initiator of an obligation is also added to the permission. Such a component makes it possible to identify a subject for an obligation that could be different from the subject in the P-RBAC permission to which the obligation applies. This should not be interpreted as giving a blank permission to the subject of the obligation to execute the action imposed by the obligation. Independently of the obligation, the obligation subject should have a permission to execute the action in the time interval which the obligation must be fulfilled; otherwise the obligation will be violated.

It is quite common that when defining a permission assignment, the subject of the obligation is not fully identified. In some cases it is assumed that the user that submitted the action request is the subject of the obligation. There are other cases in which the subject is expected to be from a set of users assigned to a particular role. In those cases the P-RBAC permission explicitly identifies the subject of the obligation using a special set of context variables listed in Table 1. The set of those special context variables is a subset of set CV of context variables of LC_0. These variables are mainly used in conditions and as subjects of obligations. Their use avoids the introduction of new notation in the model to identify obligation subjects. Details of how they are used will be apparent after we formally introduce the model.

Our temporal constraint model is based on a simple notion of time domain, that is, the pair $(\mathbb{Z}; \leqslant)$. In our context, each element of $\mathbb{Z}$ is referred to as a *time instant* and $\leqslant$ is a total order on $\mathbb{Z}$. In what follows, given $\mathtt{t},\ \mathtt{t}' \in \mathbb{Z}$, $[\mathtt{t}, \mathtt{t}']$ denotes the time interval starting at time instant $\mathtt{t}$ and ending at time instant $\mathtt{t}'$. Next definition introduces a terse yet flexible definition for temporal constraints which is the key notion in our temporal constraint model.

Definition 2. A *temporal constraint* $\mathtt{tc}$ is a tuple $(\mathtt{t_s}, \mathtt{t_e}, \mathtt{cnt})$, where $\mathtt{t_s}, \mathtt{t_e} \in \mathbb{Z}$, and $\mathtt{cnt} \in \mathbb{N}^*$. $\mathtt{tc}$ denotes a sequence of time intervals defined as follows:

- $[\mathtt{t_s}, \mathtt{t_e}], [\mathtt{t_e}+1, 2\mathtt{t_e}-\mathtt{t_s}+1], ..., [\mathtt{t_s}+(\mathtt{cnt}-1)(\mathtt{t_e}-\mathtt{t_s}+1), \mathtt{t_e}+(\mathtt{cnt}-1)(\mathtt{t_e}-\mathtt{t_s}+1)]$ if $\mathtt{t_e} \geqslant \mathtt{t_s} \geqslant 0$;

- $[\mathtt{t_s}-(\mathtt{cnt}-1)(\mathtt{t_e}-\mathtt{t_s}+1), \mathtt{t_e}-(\mathtt{cnt}-1)(\mathtt{t_e}-\mathtt{t_s}+1)], ..., [2\mathtt{t_s}-\mathtt{t_e}+1, \mathtt{t_s}-1], [\mathtt{t_s}, \mathtt{t_e}]$ if $0 \geqslant \mathtt{t_e} \geqslant \mathtt{t_s}$. □

For instance, a temporal constraint (3,7,3) represents a sequence of time intervals: [3,7],[8,12],[13,17]. For a time interval $[\mathtt{t_s}, \mathtt{t_e}]$, the time instants of the time interval are $\mathtt{t_s}, \mathtt{t_s}+1, ..., \mathtt{t_e}-1, \mathtt{t_e}$. For instance, the time instants in [3,7] are 3, 4, 5, 6, 7. For simplicity, we further assume in this paper that time is relative, and we model it as an integer representing the number of basic time instants from a predetermined time instant, denoted by 0, which is related to an action request. We identify two special time instants that are related to the predetermined time instant 0: 1) a time instant, referred to as *decision time*, that is equal to the time instant at which the decision of granting the permission to execute the action is made; 2) a time instant, referred to as *completion time*, that is the time instant of the completion of the action execution. For a time instant $\mathtt{t}$, if $\mathtt{t} > 0$, then $\mathtt{t}$ represents $|\mathtt{t}|$ time instants after a completion time. If $\mathtt{t} < 0$,

[2]In the access control literature, the term "access request" is usually used instead of "action request". However, in privacy policy, actions like "collect" and "disclose" are not an "access", therefore, we use a more appropriate term "action request" to replace "access request" thereafter in this paper.

Table 1: The ICV Set

Name	Domain	Representing
self	the set of users	the user who submits the action request
auser	the set of users	a possible user
ra	the set of roles	the role of the user whom the variable auser represents
users	the powerset of the set of users	all possible users
rs	the set of roles	the role of users whom the variable users represents

then t represents |t| time instants before a decision time. For a temporal constraint (t_s, t_e, cnt) [3], if $t_e \leqslant 0$, the temporal constraint indicates that the executed obligation is a *pre-obligation* and should be fulfilled during the time interval between the request time and the decision time. If $t_s \geqslant 0$, the temporal constraint indicates that the associated obligation is an *post-obligation* that should be fulfilled sometime after completion time. As a special case, $t_e = t_s = 0$ indicates a post-obligation that should be fulfilled right after the action has been executed.

The distinction between the time before granting permission and the time after the execution of the action gives $t = 0$ a different meaning depending on whether t represents a starting time t_s or an ending time t_e. In classical access control policies without obligations, the time at which an action request occurs, the time at which to evaluate the condition of applicable permission assignments, the time at which o authorize the action, and the time when the action is performed, are considered to be the same. In practice however these times are different especially when obligations and temporal constraints are introduced. The following example, with reference to Figure 2, illustrates the difference.

1. An employee submits a request to collect a child information on 1/6/2008.

2. Because the verifiable parental consent w.r.t. the child is not available, an applicable permission assignment requires the employee to obtain a verifiable parental consent within 6 days before making a decision for the request. Since a time interval of 6 days is allocated to the employee to obtain the consent, the decision process about the action request is suspended until 1/12/2008.

3. The decision process is resumed and the conditions in the permission start being evaluated from 1/12/2008. A policy enforcement engine may need some additional time, say 3 days, before authoring the action request, for example to verify the validity of the parental consent received and to verify whether applicable post-obligations can be fulfilled in the near future.

4. The action request is authorized at 1/15/2008 assuming all applicable post-obligations are fulfillable and the condition in the permission is evaluated to be true.

5. Once the employee obtains a clearance to collect the information, he/she may need some additional time, say 2 days, to decide whether he really wants to perform the action or prepares for obligations that should be performed immediately after the action.

6. The collecting action is performed on 1/17/2008. The timer for post-obligations starts counting.

In a temporal constraint, when $t_e = 0$ and $t_s < 0$, the constraint requires that the evaluation of the pre-obligation be completed before the evaluation of the permission condition can start. Such requirements ensures that the execution of actions within the pre-obligation that affect the value of context variables in permission conditions is completed before the permission condition is evaluated. For instance, in the example policy "Before collecting, using or disclosing personal information from a child, an operator must obtain verifiable parental consent from the child's parent.", the obligation to "obtain verifiable parental consent" may affect a context variable νpc's value by assigning to it a value from the set { "yes", "no" , and "na"}. The νpc's value in turn affects the evaluation of the conditions in the policy that are used to decide whether children information can be collected, used, or disclosed. $t_s = 0$ represents a different situation. If an action request is granted, it is possible that the subject who submits the request decides not to execute the action because of the burden imposed by the post-obligations. Therefore, it is reasonable that only after the action has been performed the timer recording the time interval allocated to post-obligations starts counting.

The key aspect of pre-obligations is that their fulfillment affects decisions about access requests. In a temporal constraints (t_s, t_e, cnt), $t_e > 0$ means that the fulfillment of the obligation is not required to precede the decision. Thus, the fulfillment has no real effect on the decision. Therefore, it is not a "meaningful" pre-obligation and it is actually a post-obligation because of $t_e > 0$. For such an obligation, it is counter intuitive for a subject to fulfill the obligation before the decision is made even though the temporal constraint allows this situation, because the decision could be "deny". Even if the decision is "permit" the subject may refuse to carry on the action requested due to reasons like heavy obligations. Therefore, in this situation, $t_s < 0$ seems not to make sense. In what follows, we assume (t_s, t_e, cnt) to be $(0, t_e, cnt)$ if $t_s < 0$ and will not discuss this situation separately again.

The following examples show how some common temporal constraints are expressed according to above definition. In the following examples, we assume "day" as the basic unit of each time instant, i.e. a time instant 1 represents 1 day.

- $(-6, 0, 1)$: this temporal constraint requires that an obligation be fulfilled within 7 days after an action request and before the decision time, that is, before making a decision on the action request, a 7-day time period is allocated to fulfill the obligation.

- $(-6, 0, 2)$: this temporal constraint requires an obligation to be fulfilled two times with 14 days after an

[3]It is obvious that $t_s \leqslant t_e$ must hold.

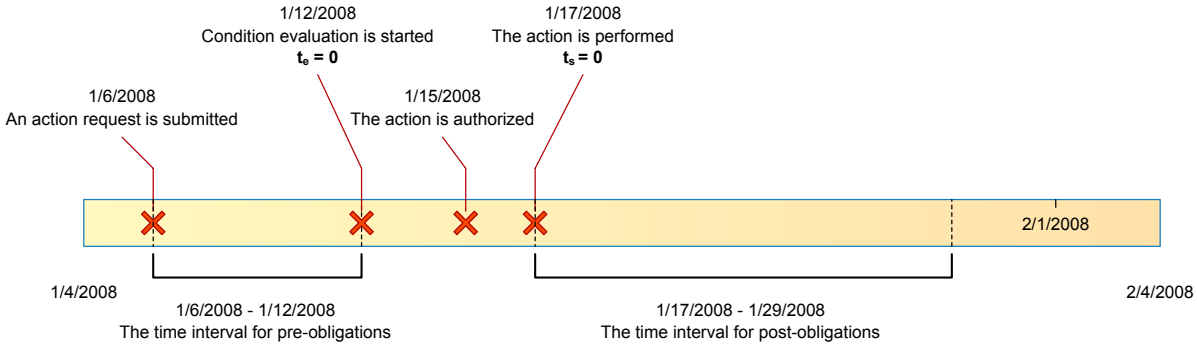

Figure 2: Important time instants on the timeline for the decision making on an action request

action request and before the decision time. One execution of the obligation happens at some time instant in a time interval [-13,-7], and another execution happens at some time instant in a time interval [-6,0].

- $(0, 181, 1)$: this temporal constraint requires that an obligation be fulfilled within half a year after an action is performed (the completion time).

- $(0, 181, \infty)$: this temporal constraint requires that an obligation be fulfilled every half a year after an action is performed, that is, after the action requested is performed, an infinite number of 182-day time periods are allocated to fulfill the obligation repeatedly, that is, one obligation in each time period.

It should be noted that the fact that a conditional obligation should be fulfilled within a time period does not mean that the obligation will be fulfilled within the time period. For instance, if the condition in a conditional obligation cannot be satisfied within its time period, the obligation will not be fulfilled. However, the condition may be evaluated again during the time period if the condition does not hold in previous attempts[4]. If $\mathfrak{cnt}$ is equal to a positive integer $\mathfrak{n}$, the corresponding obligation will be fulfilled at most $\mathfrak{n}$ times because the obligation condition may not always hold.

Readers may have noticed that in the last example it seems that the obligation would be repeated an infinite number of times. In order to prevent a possible infinite obligation fulfillments, we require that before starting a new obligation cycle the policy enforcement engine checks the condition of the obligation again. Only if the condition is still valid, the new obligation cycle can be started. By adopting this approach, policy authors can set other constraints for repeating obligations other than repeating numbers. For instance, by requiring the subject of an obligation with infinite cycles to be in a special role, the obligation is actually suspended after removing the subject from the role.

Definition 3. Let C be a set of all possible conditions expressible in LC_0, U be a set of users, ICV be a set of special context variables, A be a set of actions, O be a set of objects, and TC be a set of temporal constraints. An obligation is a tuple $(\mathsf{c}, \mathsf{s}, \mathfrak{a}, \bar{\mathfrak{o}}, \mathsf{tc})$, where $\mathsf{c} \in \mathsf{C}$, $\mathsf{s} \in \mathsf{U} \cup \mathsf{ICV}$, $\mathfrak{a} \in \mathsf{A}$, $\bar{\mathfrak{o}} \in \mathfrak{P}(\mathsf{O})$(the powerset of O), and $\mathsf{tc} \in \mathsf{TC}$. □

[4]The maximal number of tries depends on implementation.

U represents both human subject and any other subject who can initiate an action. To be clear, we call any object who can initiate an action a *user* in this paper. O contains any other component introduced in the definition, e.g. U, ICV, and TC. Moreover, the set of data D, that is introduced in original P-RBAC definition [21], is a subset of O as well. It should be noted that in theory TC could be an infinite set. Nonetheless, it is reasonable for us to assume that for each deployment of Core P-RBAC there is only a finite set of temporal constraints applicable. Based on this assumption, we further conclude that the number of possible obligations is finite as well. Some obligation examples using context variables are as follows:

- $(\mathsf{c}, \mathsf{self}, \mathfrak{a}, \{\mathfrak{o}\}, \mathsf{tc})$: if condition c is true, the subject who submits the action request activating the obligation, performs the action $\mathfrak{a}$ on object $\mathfrak{o}$ under the temporal constraint tc.

- $(\mathsf{r}\mathfrak{a} = \mathsf{r}, \mathfrak{auser}, \mathfrak{a}, \{\mathfrak{o}\}, \mathsf{tc})$: a user with role r is obligated to perform the action $\mathfrak{a}$ on object $\mathfrak{o}$ under the temporal constraint tc.

- $(\mathsf{rs} = \mathsf{r}, \mathfrak{users}, \mathfrak{a}, \{\mathfrak{o}\}, \mathsf{tc})$: all users of role r are obligated to perform the action $\mathfrak{a}$ on object $\mathfrak{o}$ under the temporal constraint tc.

4.3 Revised Core P-RBAC Model

In the original definition of Core P-RBAC, we focused on the privacy sensitive data only, which was reflected by the fact that we only defined a data set D as the only possible objects for actions in our models. Our long term goal is, however, to develop a unified RBAC model that directly supports both access control policies and privacy policies and consequently to investigate the interactions between access control policies and privacy policies. Since obligations are special actions that some subjects are obligated to execute in order to allow a subject to perform an action now, obligations in privacy policies require that access control policies and privacy policies be integrated. For instance, in order to perform an obligation, a policy enforcement engine may have to check whether there is an access control policy allowing the subject of the obligation to perform the action in the obligation. Therefore, a revised definition of P-RBAC is required to reflect such connection and directly model both privacy and access control policies.

Definition 4. The revised Core P-RBAC model is composed of the following components:

- A set D of data, a set U of users, a set R of roles, a set Pu of purposes, a set A of actions, a set C of all conditions expressible in LC_0, and a set O of objects such that $\mathsf{D} \cup \mathsf{U} \cup \mathsf{R} \cup \mathsf{Pu} \cup \mathsf{A} \cup \mathsf{CV} \subset \mathsf{O}$.
- A finite set TC of temporal constraints defined according to Definition 2, when $\mathsf{TC} \subset \mathsf{O}$.
- A set Ob of obligations on O defined according to Definition 3.
- A set $\mathsf{P} \subseteq \mathsf{C} \times \mathsf{A} \times \mathfrak{P}(\mathsf{O}) \times \mathfrak{P}(\mathsf{Pu}) \times \mathfrak{P}(\mathsf{Ob})$ of permissions.
- A set $\mathsf{PA} \subseteq \mathsf{R} \times \mathsf{P}$ of permission assignments: a many-to-many permission to role assignment relation.
- A set $\mathsf{UA} \subseteq \mathsf{U} \times \mathsf{R}$ of user assignments: a many-to-many user to role assignment relation. □

Sessions, a component of the RBAC standard [13], are omitted here for simplicity. The concept of sessions could be integrated in our model, but its details and the analysis of the interaction between sessions and obligations are out of the scope of this paper. As we can see, for any permission $\mathsf{p} = (\mathsf{c}, \mathsf{a}, \bar{\mathsf{o}}, \bar{\mathsf{p_u}}, \bar{\mathsf{o_b}})$, p is a regular access control permission if $\mathsf{c} = \top$, $\bar{\mathsf{p_u}} = \emptyset$, and $\bar{\mathsf{o_b}} = \emptyset$. Therefore, the set of P-RBAC permissions subsumes the set of RBAC permissions. A detailed description about the differences between the revised Core P-RBAC model and the previous model other than the new obligation model is discussed in Appendix A.

4.4 Some Examples

We now show how privacy policies can be expressed as Core P-RBAC permissions by using rules from COPPA and GLB acts. In order to write concise permissions we use the acronyms listed in Table 2.

Policy 1(COPPA): "Before collecting, using or disclosing personal information from a child, an operator must obtain verifiable parental consent from the child's parent."

We must define three permission assignments, each of which corresponds to one of three actions: collect, use, and disclose. Because they are similar, we only show the permission related to collect. The corresponding permission assignment is PA_1 reported in Figure 3. Because the original policy does not specify a purpose, the purpose component is an empty set. wd is a predefined constant number indicating the number of days to wait for the response. $(-\mathsf{wd}, 0, 2)$ indicates that obtain is a pre-obligation and if there is no reply in wd days an operator can ask again. The value $\mathtt{na}$ indicates that the corresponding parent has never been asked for consent before. If the value of vpc is not $\mathtt{na}$, that is, it is either $\mathtt{yes}$ or $\mathtt{no}$, it is reasonable not to ask the parent again.

Policy 2(COPPA): "An operator is required to send a new notice and request for consent to parents if there are material changes in the collection, use or disclosure practices to which the parent had previously agreed."

This policy is a good example to show the expressivity of our revised model. At first glance, it seems that we need an event (material changes) to trigger the execution of some actions by an operator. However, those actions are essentially obligations required after an operator has performed a modify action on some data. Therefore we do not need an additional event system here. Another interesting aspect is related to the "request for consent to parents". Since Policy 1 has already required "verifiable parental consent" for collecting, using or disclosing children information, we only need to reset the value of the context variable vpc to be $\mathtt{na}$ in the permission assignment related to Policy 2. After the resetting, the first action related to children information, e.g. collecting, using, or disclosing, will require an operator to obtain a new verifiable parental consent. The corresponding permission assignment is PA_2 in Figure 3.

$\top$ means that the condition is always true. The reset action resets the value of vpc to be $\mathtt{na}$. A temporal constraint $(0, 0, 1)$ requires the corresponding obligations to be fulfilled right after the action is performed.

It should be noticed that even though we explicitly place an obligation to re-obtain a parent consent here, the reset obligation is still required because the obligation "obtain verifiable consent" may take a few days to be fulfilled, and the system should make sure that no operator can collect, use, or disclose the children information during this period.

Policy 3(COPPA): "At any time, a parent may revoke his/her consent, refuse to allow an operator to further use or collect their child's personal information, and direct the operator to delete the information."

Basically, the policy authorizes some permissions to a parent who has allowed some operator to collect his/her children information. In other words, permissions assigned to a parent who has allowed operators to share his/her children information are different from those of another parent who has not. According to the spirit of RBAC, they are in different roles. In order to model this policy, we have to identify the most appropriate place to change the role of a parent. Obviously, the best place is right after the operator receive

Table 2: Acronyms Adopted in Examples

	Sort	Meaning
vpc	context variable	Verifiable Parental Consent
ci	data object	Children Information
pi	context variable	the Parent whose child Information is collected, used, or disclosed in permissions
cp	role	Children Parents
ap	role	Parents who share their children information now
bp	role	Parents who once shared their children information but do NOT share now
na	value	Not Available
wd	constant	the number of Waiting Days
am	data object	some Agreed Material of children information for collection, use, or disclosure between operators and parents
cui	data object	CUstomer Information
coi	data object	COnsumer Information
do	context variable	Data Owner whose information is disclosed
nac	role	Companies that are Not Affiliated with the company that defines policies

PA_1 :(operator, vpc = yes, collect, {ci}, ∅, {(vpc = na, self, obtain, {vpc, pi}, (−wd, 0, 2))})

PA_2 :(operator, ⊤, modify, {am}, ∅, {(⊤, self, reset, {vpc}, (0, 0, 1)), (⊤, self, notify, {cp, am, change}, (0, 0, 1))})

PA_3 :(ap, ⊤, revoke, {vpc}, ∅, {(ra = operator, auser, revoke, {self, ap}, (0, 0, 1)), (ra = operator, auser, grant, {self, bp}, (0, 0, 1))})

PA_4 :(ap, ⊤, request, {operator, deletion}, ∅, {(ra = operator, auser, delete, {ci, self}, (0, wd, 1))})

PA_5 :(bp, ⊤, request, {operator, deletion}, ∅, {(ra = operator, auser, delete, {ci, self}, (0, wd, 1))})

PA_6 :(company, ra = nac, disclose, {coi, auser}, ∅, {(⊤, self, send, {do, notice}, (0, wd, 1))})

PA_7 :(company, ra_1 = consumer, grant, {$auser_1$, customer}, ∅, {(ra_2 = company ∧ ra_1 = customer, $auser_2$, send, {$auser_1$, notice}, (0, wd, ∞))})

Figure 3: Examples

the parent's verifiable consent. In order to fully capture the semantics of this policy, several permission assignments are required, see PA_3, PA_4, and PA_5 in Figure 3.

The revoke action will set vpc to no. Once a parent performs the action allowed in the permission assignment, no operators can further collect, use, or disclose his/her children information according to Policy 1. Here we can see the ambiguities in a natural language policy. The sentence "refuse to allow an operator to further use or collect their child's personal information" does not seem to impose limitations on the disclose action. If a policy only denies the operator to further collect and use children information but do not forbid the operator to further disclose children information collected, the policy seems to be incomplete. Once a parent revokes his/her consent, his/her role will be changed from ap to bp. Since for parents in both ap and bp roles, their children information may have been collected, we assign them permissions to request some operator to delete their children information.

Policy 4(GLB act): "Consumers are entitled to receive a privacy notice from a financial institution only if the company shares the consumers' information with companies not affiliated with it, with some exceptions. Customers must receive a notice every year for as long as the customer relationship lasts."

Consumers and customers are obviously different roles. In order to grasp the essence of Policy 4, we need to specify a repeating obligation for customers. At a first glance, it is not obvious with which permission we can associate the obligation. However, since one of the key differences between customers and consumers is the repeating obligation, one natural place for the obligation to appear is the permission assigning a user to the role customer.

We do not use the role nac to replace auser in the objects of action disclose, because the role represents all companies in nac, and we believe that Policy 4 means if the company discloses a consumer information to any company in role nac it should notify the consumer. In the latter permission, we do not use self in the obligation because for a repeating obligation, its initiator usually is not the initiator of the action that causes the obligation. Even more interesting, different obligation cycles may have different initiators even though all of these initiators should belong to a same role: company[5]. In other words, the employee who deals with a customer at the beginning may not be the employee who will send privacy notices to the customer repeatedly. Because our obligation model requires that before starting a new obligation cycle, the policy enforcement engine should check the condition of the obligation again, the send obligation will be stopped once the user loses his/her customer role.

[5]In practice, we expect a more fine-grained role like customer service to replace company.

5. POLICY ANALYSIS

Large scale environments, such as enterprises, usually have to comply with complex access control policies and privacy policies. The more complex these policies are, the higher is the possibility that policies contain mistakes. Such situation can arise because of new requirements, new regulations, or just human mistakes. Therefore, there is a need for techniques to detect incorrect policies before they are deployed.

For convenience in defining concepts in this section, we refer to an action together with the objects to which the action applies as an *action pair*.

Definition 5. Let A be a set of actions, O be a set of objects, $(a, \bar{o}_1)$ and $(a, \bar{o}_2)$, where $a \in A$ and $\bar{o}_1, \bar{o}_2 \in \mathfrak{P}(O)$, be action pairs. $(a, \bar{o}_1)$ contains $(a, \bar{o}_2)$ if and only if $\bar{o}_2 \subseteq \bar{o}_1$, written as $(a, \bar{o}_2) \subseteq (a, \bar{o}_1)$. □

For instance, the action pair (correlate, $\{o_1, o_2, o_3\}$) contains the action pair (correlate, $\{o_1, o_2\}$). The containment relation between action pairs is important when we define the relation between permissions and obligations. For instance, if (correlate, $\{o_1, o_2, o_3\}$) is assigned to a role r, we can safely infer that a user with role r can perform (correlate, $\{o_1, o_2\}$) as well. By contrast, if a role r only has a permission to perform action pairs (correlate, $\{o_1, o_2\}$) and (correlate, $\{o_1, o_3\}$), r does not have sufficient privilege to perform (correlate, $\{o_1, o_2, o_3\}$).

5.1 Invalid Permissions

In the proposed obligation model, the execution of an obligation can trigger the execution of another obligation. We refer to such phenomenon as *obligation cascading*. A user that performs an obligation also needs a permission. The permission may require the execution of some other obligations. The new obligations, in turn, may require the execution of more obligations. We refer to the action pairs involved in the obligation cascading for a permission as the *cascading bag* [6] of the permission.

[6]The difference between a bag and a set is a bag can contain repeating elements.

Definition 6. Let C be a set of all conditions expressible in LC_0, A be a set of actions, O be a set of objects, Ob be a set of obligations, Pu be a set of purposes, $\mathsf{p} = (\mathsf{c}, \mathsf{a}, \bar{\mathsf{o}}, \bar{\mathsf{pu}}, \bar{\mathsf{ob}})$, where $\mathsf{c} \in \mathsf{C}$, $\mathsf{a} \in \mathsf{A}$, $\bar{\mathsf{o}} \in \mathfrak{P}(\mathsf{O})$, $\bar{\mathsf{pu}} \in \mathfrak{P}(\mathsf{Pu})$, $\bar{\mathsf{ob}} \in \mathfrak{P}(\mathsf{Ob})$, be a permission. p is an *invalid permission* if one of the following conditions holds:

1. c is not satisfiable.
2. $\mathsf{ob}' = \{\mathsf{c}', \mathsf{s}', \mathsf{a}', \bar{\mathsf{o}}', (\mathsf{t}'_\mathsf{s}, \mathsf{t}'_\mathsf{e}, \mathsf{cnt}')\} \in \bar{\mathsf{ob}}$ exists such that one of the following conditions holds:
   (a) c' is unsatisfiable.
   (b) $\mathsf{c} \wedge \mathsf{c}'$ is unsatisfiable.
   (c) s' has no permission to perform action a' on $\bar{\mathsf{o}}'$.
   (d) c' is equal to $\top$ and cnt' is ∞.
3. an action pair, which exists in the cascading bag of permission p, requires p. □

If a condition in a permission or an obligation is not satisfiable, the permission or the obligation is useless (conditions 1, 2(a), and 2(b)). If an obligation requires a non-existing permission, the obligation cannot be fulfilled (condition 2(c)). Conditions 2(d) and 3 are used to prevent infinite obligations caused by a permission. Conditions 1-2(d) are easy to check. We now present an algorithm to detect condition 3 (refer to Algorithm 1).

Lemma 1. Let Σ be a set of Core P-RBAC permissions. If no invalid permission exists in Σ, given any permission p in Σ, the cascading bag of p is finite. □

Proof. (Sketch) Since the set of actions is finite and the set of objects is finite, the set of all action pairs is finite. Because no invalid permission exists, each action pair can only appear once in the cascading bag. Therefore, the number of action pairs in the cascading bag of p at most equals the number of all action pairs. □

Algorithm 1 Condition 3 detection

```
1: function C6Detection(p, cb)                ▷ p: the permission to be checked; cb: the list of obligations, initial value are the obligations in p
2:     if cb == null then
3:         return false
4:     end if
5:     ob ← removehead(cb)
6:     if p.actionpair ⊇ ob.actionpair then
7:         return true
8:     end if
9:     pl ← permissions required for ob
10:    for all p′ in pl do
11:        concat cb with the obligation list in p′
12:    end for
13:    return C3Detection(p, cb)
14: end function
```

The key idea of the algorithm is to build a list to store the cascading obligations, and then recursively execute the following steps: 1) retrieve an obligation from the list (the obligation is removed from the list); 2) check the obligation; 3) retrieve the permissions required for the obligation; 4) add all obligations in these permissions to the list. Assuming the number of different action pairs to be n, the worst case complexity is in $\mathrm{O}(\mathsf{n})$. The reason for such complexity is that during the recursive calls, each action pair at most appears once. Lines 6 and 9 may require some additional time. However, given reasonable assumptions, such as the maximal number of objects in an action pair to be a constant, the maximal number of permissions of a role to be a constant, and the maximal number of users of a role to be constant, Lines 6 and 9 run in constant time. It should be noted the algorithm is just for illustrating the basic idea and is not optimized to detect condition 6 for all permissions. The complexity of a naive application of this algorithm to detect condition 3 for all permissions is $\mathrm{O}(\mathsf{mn})$ assuming the number of permissions to be m. However, it is easy to define an $\mathrm{O}(\mathsf{m}+\mathsf{n})$ algorithm able to detect all invalid permissions caused by condition 3. One possible approach with an $\mathrm{O}(\mathsf{m}+\mathsf{n})$ complexity is to apply dynamic programming by creating an array to record the intermediate results of action pairs having been checked. The details are omitted for space reason.

5.2 Coverage of Obligations

In Core P-RBAC, given an action request, pre-obligations in all permissions that contain the action pair in the request have to be fulfilled before evaluating the conditions, and post-obligations in all applicable permissions have to be fulfilled in order to perform the action. Therefore, we can expect that some action request could lead to a large number of obligations returned, especially from ill-written policies. Therefore reducing the number of obligations to be executed may have significant practical impact. Obviously, the remaining obligations should not decrease the duty required by the original policies. On the other hand, we can imagine that many of these obligations are similar to each other since they are obligations associated with similar permissions. If the similarity can lead to some obligation relation like set containment, we may safely remove some obligations.

In order to better understand the possibility before entering into details, we first discuss one example. Given two post-obligations, one requiring to send a privacy notice to both children and a parent within one week, and another requiring to send the same privacy notice to the parent within two weeks, if both of them are in the post-obligation set returned upon a user action request, it is reasonable that the user only needs to fulfill the former one because the duty represented in the latter one is "covered" by the former one. In this paper, we use the term *coverage* to represent this relation, that is, the former obligation covers the latter one. There are several factors affecting the coverage relation of obligations, and the first of them to be investigated is the temporal constraint.

Definition 7. Let $\mathsf{tc}_1 = (\mathsf{t}_{s1}, \mathsf{t}_{e1}, \mathsf{cnt}_1)$ and $\mathsf{tc}_2 = (\mathsf{t}_{s2}, \mathsf{t}_{e2}, \mathsf{cnt}_2)$, $\mathsf{t}_{s1}, \mathsf{t}_{e1}, \mathsf{t}_{s2}, \mathsf{t}_{e2} \in \mathbb{Z}$ and $\mathsf{cnt}_1, \mathsf{cnt}_2 \in \mathbb{N}^*$, be two temporal constraints. tc_2 is stricter than tc_1, written as $\mathsf{tc}_2 \rhd \mathsf{tc}_1$, if and only if one of the following conditions hold:

- $\mathsf{t}_{s1} \geqslant \mathsf{t}_{s2} \geqslant 0$ and $(\mathsf{t}_{e1} - \mathsf{t}_{s1}) \geqslant (\mathsf{t}_{e2} - \mathsf{t}_{s2})$ and $\mathsf{cnt}_2 \geqslant \mathsf{cnt}_1$.
- $\mathsf{t}_{e2} \leqslant \mathsf{t}_{e1} \leqslant 0$ and $(\mathsf{t}_{e1} - \mathsf{t}_{s1}) \geqslant (\mathsf{t}_{e2} - \mathsf{t}_{s2})$ and $\mathsf{cnt}_1 \geqslant \mathsf{cnt}_2$. □

For instance, given the post-obligation temporal constraints (0, 5, 2) and (0, 4, 3), it is obvious that (0, 4, 3) is stricter than (0, 5, 2) because (0, 4, 3) requires that the fulfillment of a corresponding obligation be started right after an action is performed, and completed in 5 days. Moreover this obligation fulfillment cycle should be repeated 3 times. On the other hand, (0, 5, 2) requires that the fulfillment of a corresponding obligation be completed in 6 days and only to be repeated 2 times. Post-obligations are duties; therefore the smaller the repeating number is, the less strict the temporal constraint is.

Given the pre-obligation temporal constraints (-5,0,2) and (-3,0,1), (-3,0,1) is stricter than (-5,0,2) because only one chance, one 4-day time period, is given to fulfill a pre-obligation by (-3,0,1) and two chances, two 6-day time periods, are given to fulfill a pre-obligation by (-5,0,2). In order to understand the strictness of a temporal constraint in a pre-obligation, we have to first realize the difference between pre-obligations and post-obligations. Unlike post-obligations, pre-obligations are actions that must be fulfilled before the requested action can be allowed and *the result of the fulfillment can determine whether the requested action is executable or not*[7]. For instance, the result of $(\mathtt{obtain}, \{\mathtt{pi}, \nu\mathtt{pc}\})$ affects the value of $\nu\mathtt{pc}$ that further determines whether collection of children information is allowed in COPPA policy 1. Therefore, the smaller the count number in a pre-obligation temporal constraint is, the stricter the temporal constraint is. In other words, the bigger the count number in a pre-obligation temporal constraint is, the more chances are that the condition of the permission be satisfied. The careful reader may argue that in practice a parent may give his consent in the first time and revoke his consent the second time. However, on the one hand, this situation is a result of human factors that are out of the scope of our model and therefore is not related to temporal constraints. On the other hand, this situation can be avoided somewhat by the condition of an obligation. For instance, the condition $\nu\mathtt{pc} = \mathtt{na}$ in the obligation actually prevents an operator from asking $\nu\mathtt{pc}$ again once the operator obtains a positive answer in his/her first attempt.

Definition 8. Let $\mathtt{ob}_1 = (\mathtt{c}_1, \mathtt{s}_1, \mathtt{a}_1, \bar{\mathtt{o}}_1, \mathtt{tc}_1)$ and $\mathtt{ob}_2 = (\mathtt{c}_2, \mathtt{s}_2, \mathtt{a}_2, \bar{\mathtt{o}}_2, \mathtt{tc}_2)$, where $\mathtt{c}_1, \mathtt{c}_2 \in \mathsf{C}$, $\mathtt{s}_1, \mathtt{s}_2 \in \mathsf{CV} \cup \mathsf{U}$, $\mathtt{a}_1, \mathtt{a}_2 \in \mathsf{A}$, $\bar{\mathtt{o}}_1, \bar{\mathtt{o}}_2 \in \mathfrak{P}(\mathsf{O})$, and $\mathtt{tc}_1, \mathtt{tc}_2 \in \mathsf{TC}$, be obligations.

- a post-obligation $\mathtt{ob}_2$ covers a post-obligation $\mathtt{ob}_1$ if $\mathtt{c}_1$ implies $\mathtt{c}_2$, $\mathtt{s}_1 = \mathtt{s}_2$, $(\mathtt{a}_1, \bar{\mathtt{o}}_1) \subseteq (\mathtt{a}_2, \bar{\mathtt{o}}_2)$, and $\mathtt{tc}_2 \rhd \mathtt{tc}_1$, written as $\mathtt{ob}_2 \rhd \mathtt{ob}_1$.
- a pre-obligation $\mathtt{ob}_2$ covers a pre-obligation $\mathtt{ob}_1$ if $\mathtt{c}_2$ implies $\mathtt{c}_1$, $\mathtt{s}_1 = \mathtt{s}_2$, $(\mathtt{a}_2, \bar{\mathtt{o}}_2) \subseteq (\mathtt{a}_1, \bar{\mathtt{o}}_1)$, and $\mathtt{tc}_2 \rhd \mathtt{tc}_1$, written as $\mathtt{ob}_2 \rhd \mathtt{ob}_1$. □

Likewise, to understand the difference between the coverage of pre-obligations and that of post-obligations, we first have to understand some tricky but important differences between pre-obligations and post-obligations. A pre-obligation is an action to be fulfilled in order to satisfy some condition before the current action is taken. In order to improve the success rate, sometime some alternatives are provided in addition to the mandatory objects in the action pair of the obligations. The more alternatives provided by the pre-obligation, the higher the possibilities for the condition to be satisfied. Taking this $(\mathtt{obtain}, \{\mathtt{pi}, \nu\mathtt{pc}\})$ obligation as an example, if this obligation is combined with some incentive, say a free children magazine subscription, rewarded to the user giving his consent, it is easier to succeed. Therefore, $(\mathtt{obtain}, \{\mathtt{pi}, \nu\mathtt{pc}\})$ as a pre-obligation covers $(\mathtt{obtain}, \{\mathtt{pi}, \nu\mathtt{pc}, \mathtt{stimulus}\})$, i.e. if $(\mathtt{obtain}, \{\mathtt{pi}, \nu\mathtt{pc}\})$ succeeds, $(\mathtt{obtain}, \{\mathtt{pi}, \nu\mathtt{pc}, \mathtt{stimulus}\})$ succeeds as well.

On the contrary, $(\mathtt{obtain}, \{\mathtt{pi}, \nu\mathtt{pc}, \mathtt{stimulus}\})$ as a post-obligation covers $(\mathtt{obtain}, \{\mathtt{pi}, \nu\mathtt{pc}\})$ because after the action has been performed, these post-obligations become duties. "$\mathtt{c}_1$ implies $\mathtt{c}_2$" means if $\mathtt{c}_1$ is true then $\mathtt{c}_2$ is true as well. If a post-obligation $\mathtt{pob}$ covers another post-obligation $\mathtt{pob}'$, $\mathtt{pob}$ has more chances to be invoked. If a pre-obligation $\mathtt{eob}$ covers another pre-obligation $\mathtt{eob}'$, $\mathtt{eob}$ has less chance to be invoked. A pre-obligation $\mathtt{eob}$ makes it less probable that a condition be satisfied because of less "alternatives", and a post-obligation $\mathtt{pob}$ guarantees the answer to an action request is in accordance to the stricter policies. Based the definition of the coverage of obligations, we define rules to safely reduce the number of obligations:

- If a pre-obligation $\mathtt{eob}$ covers another pre-obligation $\mathtt{eob}'$, $\mathtt{eob}$ is removed because $\mathtt{eob}'$ actually increases the probability that an action request is allowed without violating policies.
- If a post-obligation $\mathtt{pob}$ covers another post-obligation $\mathtt{pob}'$, $\mathtt{pob}'$ is removed because the duties of $\mathtt{pob}'$ are fully covered by heavier post-obligations.

Based on the aforementioned definitions, now we present Algorithm 2 to compare the coverage between two post-obligations. The algorithm directly follows the definition. It should be noted that to detect the implication relation between two conditions, we directly apply the scope function which was introduced by Ni et al.[21], for computing the set of context variable assignments to satisfy a condition. If we reasonably assume the maximal number of atomic conditions in each condition to be constant and the maximal number of elements in each domain to be constant, line 19 runs in constant time. If we reasonably assume the maximal number of objects in an action pair to be constant, lines 6 and 11 run in constant time as well. Since there is no line that can run worse than in constant time, the algorithm runs in constant time. Such complexity result is important because the need of reducing the number of obligations arises when answering action requests. The coverage comparison is the base of some other policy analysis, e.g., consistency analysis, which is beyond the scope of this paper.

6. CONCLUSIONS

In this paper, we present a novel obligation model for P-RBAC and elaborate its design choices. Two efficient algorithms, one for minimizing invalid permissions and another for comparing the coverage of two obligations, are proposed as well. Due to the complexity of obligations in privacy policies, the work reported in this paper is at its initial stage. Many interesting problems are still left open. For instance, the interactions between obligations and the execution sequence of obligations need further investigation; compensation and reward mechanisms regarding the status of the fulfillment of obligations may be necessary for some

[7]If a pre-obligation has no effect on the condition associated with a permission, the pre-obligation should be a post-obligation.

Algorithm 2 Coverage Detection

```
1: function COVERAGE(ob1, ob2)          ▷ ob1, ob2: two obligations
2:     rtn ← 0
3:     if ob1.s ≠ ob2.s then
4:         return 0                     ▷ 0 represents non-comparable
5:     end if
6:     if ob1.actionpair ⊆ ob2.actionpair then
7:         lob ← ob1
8:         hob ← ob2
9:         rtn ← 2                      ▷ 2 represents that ob2 covers ob1
10:    end if
11:    if ob1.actionpair ⊇ ob2.actionpair then
12:        lob ← ob2
13:        hob ← ob1
14:        rtn ← 1                      ▷ 1 represents that ob1 covers ob2
15:    end if
16:    if rtn = 0 or lob.tc ⋪ hob.tc then
17:        return 0
18:    end if
19:    if Scope(lob.c) ⊈ Scope(hob.c) then
20:        return 0
21:    end if
22:    return rtn
23: end function
```

cases; mechanisms for monitoring the fulfillment of pending obligations and optimizing the sequence of the fulfillment of pending obligations need to be devised; a solution to the accountability problem [17] under P-RBAC context is interesting as well.

7. ACKNOWLEDGEMENT

The work reported here has been supported by the IBM OCR project "Privacy and Security Policy Management" and the NSF grant 0712846 "IPS: Security Services for Healthcare Applications".

8. REFERENCES

[1] M. Backes, B. Pfitzmann, and M. Schunter. A toolkit for managing enterprise privacy policies. In *ESORICS*, pages 162–180, 2003.

[2] A. Barth, A. Datta, J. C. Mitchell, and H. Nissenbaum. Privacy and contextual integrity: Framework and applications. In *S&P*, pages 184–198. IEEE Computer Society, 2006.

[3] E. Bertino, C. Bettini, E. Ferrari, and P. Samarati. An access control model supporting periodicity constraints and temporal reasoning. *ACM Trans. Database Syst.*, 23(3):231–285, 1998.

[4] E. Bertino, P. A. Bonatti, and E. Ferrari. Trbac: A temporal role-based access control model. *ACM Trans. Inf. Syst. Secur.*, 4(3):191–233, 2001.

[5] C. Bettini, S. Jajodia, X. Wang, and D. Wijesekera. Obligation monitoring in policy management. In *POLICY '02: Proceedings of the 3rd International Workshop on Policies for Distributed Systems and Networks (POLICY'02)*, page 2, Washington, DC, USA, 2002. IEEE Computer Society.

[6] C. Bettini, S. Jajodia, X. S. Wang, and D. Wijesekera. Provisions and obligations in policy management and security applications. In *VLDB '02: Proceedings of the 28th international conference on Very Large Data Bases*, pages 502–513. VLDB Endowment, 2002.

[7] C. Bettini, S. Jajodia, X. S. Wang, and D. Wijesekera. Provisions and obligations in policy management and security applications. In *VLDB*, pages 502–513. Morgan Kaufmann, 2002.

[8] C. Bettini, S. Jajodia, X. S. Wang, and D. Wijesekera. Provisions and obligations in policy rule management. *J. Network Syst. Manage.*, 11(3), 2003.

[9] M. A. Brown. Conditional obligation and positive permission for agents in time. *Nordic Journal of Philosophical Logic*, 5(2):83–112, 2000.

[10] N. Damianou, N. Dulay, E. Lupu, and M. Sloman. The ponder policy specification language. In M. Sloman, J. Lobo, and E. Lupu, editors, *POLICY*, volume 1995 of *Lecture Notes in Computer Science*, pages 18–38. Springer, 2001.

[11] D. J. Dougherty, K. Fisler, and S. Krishnamurthi. Obligations and their interaction with programs. In J. Biskup and J. Lopez, editors, *ESORICS*, volume 4734 of *Lecture Notes in Computer Science*, pages 375–389. Springer, 2007.

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

[13] 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.

[14] P. Gama and P. Ferreira. Obligation policies: An enforcement platform. In *POLICY*, pages 203–212. IEEE Computer Society, 2005.

[15] M. Hilty, D. A. Basin, and A. Pretschner. On obligations. In S. D. C. di Vimercati, P. F. Syverson, and D. Gollmann, editors, *ESORICS*, volume 3679 of *Lecture Notes in Computer Science*, pages 98–117. Springer, 2005.

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

[17] K. Irwin, T. Yu, and W. H. Winsborough. On the modeling and analysis of obligations. In *CCS '06: Proceedings of the 13th ACM conference on Computer and communications security*, pages 134–143, New York, NY, USA, 2006. ACM.

[18] J. Joshi, E. Bertino, U. Latif, and A. Ghafoor. A generalized temporal role-based access control model. *IEEE Trans. Knowl. Data Eng.*, 17(1):4–23, 2005.

[19] L. Kagal, T. Finin, and A. Joshi. A policy language for a pervasive computing environment. In *POLICY '03: Proceedings of the 4th IEEE International Workshop on Policies for Distributed Systems and Networks*, page 63, Washington, DC, USA, 2003. IEEE Computer Society.

[20] Q. Ni, D. Lin, E. Bertino, and J. Lobo. Conditional privacy-aware role based access control. In *ESORICS '07: Proceedings of the 12th European Symposium On Research In Computer Security*, pages 72–89. Springer, 2007.

[21] Q. Ni, A. Trombetta, E. Bertino, and J. Lobo. Privcy aware role based access control. In *SACMAT '07: Proceedings of the 12th ACM symposium on Access control models and technologies*, New York, NY, USA, 2007. ACM Press.

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

[23] H. Prakken and M. J. Sergot. Contrary-to-duty obligations. *Studia Logica*, 57(1):91–115, 1996.

[24] M. Sailer and M. Morciniec. Monitoring and execution for contract compliance. HPL-2001-261R1, HP LAB, HP. Available at http://www.hpl.hp.com/techreports/2001/HPL-2001-261R1.html.

[25] P. Samarati, P. Y. A. Ryan, D. Gollmann, and R. Molva, editors. *Computer Security - ESORICS 2004, 9th European Symposium on Research Computer Security, Sophia Antipolis, France, September 13-15, 2004, Proceedings*, volume 3193 of *Lecture Notes in Computer Science*. Springer, 2004.

[26] 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.

[27] J. Skene, A. Skene, J. Crampton, and W. Emmerich. The monitorability of service-level agreements for application-service provision. In V. Cortellessa, S. Uchitel, and D. Yankelevich, editors, *WOSP*, pages 3–14. ACM, 2007.

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

[29] 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/.

[30] A. Uszok, J. Bradshaw, R. Jeffers, N. Suri, P. Hayes, M. Breedy, L. Bunch, M. Johnson, S. Kulkarni, and J. Lott. Kaos policy and domain services: Toward a description-logic approach to policy representation, deconfliction, and enforcement. *policy*, 00:93, 2003.

APPENDIX

A. OTHER DIFFERENCES

The differences between the revised Core P-RBAC model and the previous model other than the new obligation model are as follows:

- Actions are performed on objects in our revised model rather than a data item in previous model. The data set D is a subset of O. Moreover, an action could be performed on a set of objects. The new permission can not only support regular access control policies, but also establish a natural connection between actions in permissions and actions in obligations. This topic will be discussed in next section.
- Zero or multiple purposes can be indicated rather than only a single purpose as in the previous model. If there is no purpose in a permission, the permission is a regular one. If there are several purposes, say $\mathfrak{n}$, the permission is semantically equivalent to $\mathfrak{n}$ permissions each of which has one of these $\mathfrak{n}$ purposes.

We use an example to illustrate the differences between these two models and some interesting effects resulting from these differences. Consider the following policy data set: $\mathsf{C} = \{\mathsf{c}\}, \mathsf{A} = \{\mathfrak{a}\}, \mathsf{O} = \{\mathsf{o}_1, \mathsf{o}_2\}, \mathsf{Pu} = \{\mathsf{pu}_1, \mathsf{pu}_2\}, \mathsf{Ob} = \{\mathsf{ob}_1, \mathsf{ob}_2\}$ and permission: $\mathsf{p} = (\mathsf{c}, \mathfrak{a}, \{\mathsf{o}_1, \mathsf{o}_2\}, \{\mathsf{pu}_1, \mathsf{pu}_2\}, \{\mathsf{ob}_1, \mathsf{ob}_2\})$. Then p is semantically equivalent to the following two permissions: $\mathsf{p}_1 = (\mathsf{c}, \mathfrak{a}, \{\mathsf{o}_1, \mathsf{o}_2\}, \{\mathsf{pu}_1\}, \{\mathsf{ob}_1, \mathsf{ob}_2\})$ and $\mathsf{p}_2 = (\mathsf{c}, \mathfrak{a}, \{\mathsf{o}_1, \mathsf{o}_2\}, \{\mathsf{pu}_2\}, \{\mathsf{ob}_1, \mathsf{ob}_2\})$. In other words, the new purpose component can be considered as a syntactic sugar for an efficient representation of purposes in permissions. The rationale behind this choice is easily understood.

However, p is not semantically equivalent to the following two permissions: $\mathsf{p}_3 = (\mathsf{c}, \mathfrak{a}, \{\mathsf{o}_1\}, \{\mathsf{pu}_1, \mathsf{pu}_2\}, \{\mathsf{ob}_1, \mathsf{ob}_2\})$ and $\mathsf{p}_4 = (\mathsf{c}, \mathfrak{a}, \{\mathsf{o}_2\}, \{\mathsf{pu}_1, \mathsf{pu}_2\}, \{\mathsf{ob}_1, \mathsf{ob}_2\})$. Once a permission p is assigned to a role r, permission p_3 and p_4 can be treated as if they were assigned to the role r as well. If a user assigned to a role is allowed to perform an action on objects o_1 and o_2, it is natural that the user should be allowed to perform the action on an object o_1 or on an object o_2 separately. Nevertheless, if a role is assigned both permissions p_3 and p_4, users assigned to the role cannot perform an action $\mathfrak{a}$ on objects o_1 and o_2 simultaneously. For instance, a permission p assigned to a role r_1 is running a data mining application dm on tables customer and orders, a permission p_3 assigned to a role r_2 is running dm on customer, and a permission p_4 assigned to a role r_2 is running dm on order. A user u_1 assigned to r_1 usually can obtain more information that another user u2 assigned to r_2 because there is no way for u_2 to infer more information by connecting the internal mining results from these tables.