CERIAS Tech Report 2005-12

PURPOSE BASED ACCESS CONTROL OF COMPLEX DATA FOR PRIVACY PROTECTION

by Ji-Won Byun and Elisa Bertino and Ninghui Li

Center for Education and Research in
Information Assurance and Security,
Purdue University, West Lafayette, IN 47907-2086

Purpose Based Access Control of Complex Data for Privacy Protection

Ji-Won Byun
byunj@cs.purdue.edu

Elisa Bertino
bertino@cerias.purdue.edu

Ninghui Li
ninghui@cs.purdue.edu

Center for Education and Research in Information Assurance and Security
and Department of Computer Sciences
Purdue University
656 Oval Drive, West Lafayette, IN 47907

ABSTRACT

As privacy becomes a major concern for both consumers and enterprises, many research efforts have been devoted to the development of privacy protecting technology. We recently proposed a privacy preserving access control model for relational databases, where purpose information associated with a given data element specifies the intended use of the data element. In this paper, we extend our previous work to handle other advanced data management systems, such as the ones based on XML and the ones based on the object-relational data model. Another contribution of our paper is that we address the problem of how to determine the purpose for which certain data are accessed by a given user. Our proposed solution relies on the well-known RBAC model as well as the notion of conditional role which is based on the notions of role attribute and system attribute.

1. INTRODUCTION

The rapid advances in database systems and information technology have greatly increased the awareness of the need for privacy protection. The fact that personal information can be collected, stored and used creates fear of privacy violation for many online consumers. Also, potential lawsuits brought up by consumers and recently enacted privacy legislations [11, 5, 10] require organizations to pay closer attention to the management of private data.

As privacy becomes a major concern for both consumers and enterprises, many research efforts have been devoted to the development of privacy protecting technology. As an important step for helping users to gain control over the use of their personal information, the W3C has proposed the Platform for Privacy Preference (P3P) [17]. P3P allows websites to encode their privacy practice, such as what information is collected, who can access the data for what purposes, and how long the data will be stored by the sites, in a machine-readable format. Even though P3P provides a standard means for enterprises to make privacy promises to their users, P3P does not provide any mechanism to ensure that these promises are consistent with the internal data processing.

To the best of our knowledge, the first approach to incorporate privacy protection within relational database systems was proposed by Agrawal et al. [1]. Introducing the concept of Hippocratic databases, Agrawal et al. showed that much work is needed to be done in order to build database systems which accurately protect private information. Observing the lack of adequate privacy protecting systems, we also recently proposed a privacy preserving access control model for relational databases, based on the notion of purpose [3]. In our work, purpose information associated with a given data element specifies the intended use of the data element. A key characteristics of our model is that multiple purposes can be associated with each data element. We also exploit query modification techniques to support data access control based on the purpose information.

Our previous work has, however, some limitations. The first is that it has been developed based on a simple data model, that is, the relational data model. However, many advanced data management systems, such as the ones based on XML and the ones based on the object-relational data model, need to manage objects that are complex, have hierarchical structures and are characterized by several semantic relationships. We thus need a more sophisticated purpose management model; such model is the first contribution of this paper. The second limitation of our previous work is that it does not adequately address the problem of how to determine the purpose for which certain data are accessed by a given user. We believe that this issue may be satisfactorily addressed by relying on the well-known RBAC model [13, 15]. However, in order to support policies specifying for which purpose a certain data can be accessed by a given role, we need to expand conventional RBAC models with the notion of conditional role which is based on the notions of role attribute and system attribute. Such extended RBAC model is the second contribution of this paper.

The remainder of this paper is organized as follows. Section 2 reviews related work that motivates the discussion in the paper. Section 3 provides a brief overview of purpose based access control. Section 4 introduces the notion of conditional role and presents a method for determining access purposes. Section 5 describes our hierarchical data model and intended purpose labeling scheme, whereas Section 6 discusses query compliance based on intended purpose labels. Section 7 suggests future work and concludes our discussion.

2. RELATED WORK

Our work is related to many areas of privacy preserving access control, specifically privacy policy specification and private data management systems. We also exploit the tremendous work carried out for traditional access control which mainly focuses on secure management of data. In

this section we briefly survey related work that motivates our discussion.

The concept of Hippocratic databases, incorporating privacy protection within relational database systems, was introduced by Agrawal et al. [1]. The proposed architecture uses privacy metadata, which consist of privacy policies and privacy authorizations stored in two tables. A privacy policy defines for each attribute of a table the usage purpose(s), the external-recipients and retention period, while a privacy authorization defines which purposes each user is authorized to use.

Recently, Lefevre et al. [9] presented an approach to enforcing privacy policy in database environments. Their work focuses on ensuring limited data disclosure, based on the premise that data providers have control over who is allowed to see their personal data and for what purpose. In their work, they introduce two models of cell-level limited disclosure enforcement; table semantics and query semantics. They also suggest an implementation based on query modification techniques.

Previous work on multilevel secure relational databases [16, 14, 6] also provides many valuable insights for designing a fine-grained secure data model. In a multilevel relational database system, every piece of information is classified according to a security level, and every user is assigned a security clearance. Based on this access class, the system ensures that each user gains access to only the data for which she has proper clearance, according to the well known basic restrictions [2]. These constraints ensure that there is no information flow from a lower security level to a higher security level and that subjects with different clearances see different versions of multilevel relations.

Rabitti et al. [12] developed a comprehensive authorization model designed for next-generation database systems. The data models considered in their work support object-oriented concepts and incorporate some key semantic data modeling concepts such as composite objects and versions. Furthermore, they formalize and develop a clear semantics for various types of authorizations such as strong/weak and negative/positive authorizations. A number of key issues that arise in implementing such a model are also discussed in their work.

Role Based Access Control (RBAC) [13, 15], which has made a significant impact on many access control systems, greatly simplifies the specification and management of security policies within an enterprise. The basic concept of RBAC is as follows: permissions are assigned to functional roles within an enterprise and individual users are then authorized to the necessary permissions by being appropriately assigned to a role or a set of roles. Most RBAC models also include a role hierarchy, a partial order defining a relationship between roles, to facilitate the administration tasks.

The idea of incorporating attributes into RBAC models has been proposed to address some distinct problems in RBAC. Chen et al. [4] introduced the attributes associated with roles in order to enforce global constraints such as the principle of separation of duty. They discuss various attributes for roles, permissions, users and sessions and suggest a practical way to specify and enforce constraints based on these attributes. The notion of role attribute is also presented in [7, 8]. In their work, role attributes are used to provide more flexibility to the access control in RBAC. Using their approaches, access control policies sensitive to the system context can be easily specified and enforced.

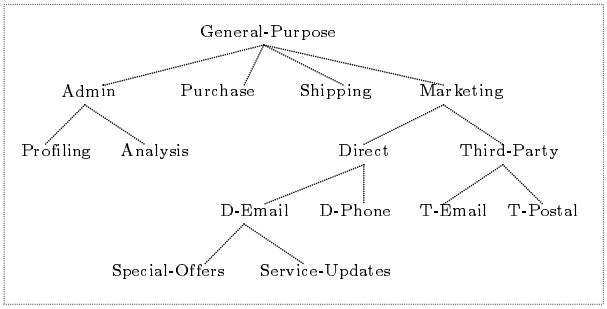


Figure 1: Purpose Tree

3. PRELIMINARIES

In this section, we present an overview of the notion of purpose, previously introduced in [3].

In privacy protecting access control models, the notion of purpose plays a central role as the purpose is the basic concept on which access decisions are made. In order to simplify the management, purposes are organized according to a hierarchical structure based on the principles of generalization and specialization, which is appropriate in common business environments.

Definition 1. (*Purpose and Purpose Hierarchy*) A purpose describes the reason(s) for data collection and data access. A purpose hierarchy is defined as a 2-tuple $\langle \mathcal{P}, \leq \rangle$, where $\mathcal{P}$ is a set of purposes and $\leq$ is a partial order defined over $\mathcal{P}$. Let p_i, $p_j \in \mathcal{P}$. We say that p_i is a specialization of p_j if $p_i \leq p_j$. □

As in [3], we assume that the partial order is a tree which we refer to as *Purpose Tree*. Figure 1 gives an example of purpose tree. The following notations will be used through out the paper.

Notation 1. (*Ancestors and Descendants*) Let $\mathcal{PT}$ be a purpose tree and $\mathcal{P}$ be the set of purposes in $\mathcal{PT}$. Let p_i be a purpose in $\mathcal{PT}$.

1. Ancestors(p_i) is the set of all nodes that are ancestors of p_i in $\mathcal{PT}$, including p_i itself.
2. Descendants(p_i) is the set of all nodes that are descendants of p_i in $\mathcal{PT}$, including p_i itself.

Intuitively, an access to a specific data item is allowed if the purposes allowed by privacy policies for the data include or imply the purpose for accessing the data. We refer to purposes associated with data and thus regulating data accesses as *Intended Purposes*, and to purposes for accessing data as *Access Purposes*. Intended purposes can be viewed as brief summaries of privacy policies for data, stating for which purposes data can be accessed. When an access to data is requested, the access purpose is checked against the intended purposes for the data.

In this model intended purposes support both positive and negative privacy policies. An intended purpose consists of two components: *Allowed Intended Purposes* and *Prohibited Intended Purposes*. This structure provides greater flexibility to the access control model. Moreover, by using prohibited intended purposes, one can guarantee that data

accesses for particular purposes are never allowed. Conflicts between the allowed intended purposes and the prohibited intended purposes for the same data item are resolved by applying the denial-takes-precedence policy where prohibited intended purposes override allowed intended purposes.

Definition 2. (*Intended Purpose*) Let $\mathcal{PT}$ be a purpose tree and $\mathcal{P}$ be the set of all purposes in $\mathcal{PT}$. An intended purpose, IP, is a tuple $\langle$AIP, PIP$\rangle$, where AIP $\subseteq \mathcal{P}$ is a set of allowed intended purposes and PIP $\subseteq \mathcal{P}$ is a set of prohibited intended purposes. Let AIP = $\{\text{aip}_1, \ldots, \text{aip}_n\}$ be a set of allowed intended purposes and PIP = $\{\text{pip}_1, \ldots, \text{pip}_m\}$ be a set of prohibited intended purposes. We define the set of intended purposes entailed by AIP and PIP as follows:

1. $\text{AIP}^{\downarrow} = \bigcup_{\text{aip}_j \in \text{AIP}} \text{Descendants}(\text{aip}_j)$
2. $\text{PIP}^{\updownarrow} = (\bigcup_{\text{pip}_k \in \text{PIP}} \text{Ancestors}(\text{pip}_k)) \cup (\bigcup_{\text{pip}_k \in \text{PIP}} \text{Descendants}(\text{pip}_k))$ □

Example 1. Suppose IP = $\langle$AIP={Admin, Direct}, PIP= {D-Email}$\rangle$ is defined over the purpose tree given in Figure 1.

1. $\text{AIP}^{\downarrow}$ = Descendants(Admin) ∪ Descendants(Direct) = {Admin, Profiling, Analysis} ∪ {Direct, D-Email, D-Phone, D-Postal, Special-Offers, Service-Updates}
2. $\text{PIP}^{\updownarrow}$ = Descendants(D-Email) ∪ Ancestors(D-Email) = {D-Email, Special-Offers, Service-Updates} ∪ {D-Email, Direct, Marketing, General-Purpose}

An access purpose is the purpose of a particular data access, which is determined or validated by the system when the data access is requested. We now present the formal definition of access purpose and discuss further the issue of how to determine access purposes in Section 4.

Definition 3. (*Access Purpose*) Let $\mathcal{PT}$ be a purpose tree. An access purpose, denoted by AP, is a purpose for accessing a data element, and it is a node in $\mathcal{PT}$. □

As already discussed, an access decision is made based on the relationship between the access purpose and the intended purposes of data. That is, an access is granted if the access purpose is entailed by the allowed intended purposes and not entailed by the prohibited intended purposes; in this case we say the access purpose is *compliant with* the intended purpose. The access is denied if any of these two conditions fails; we then say that the access purpose is *not compliant with* the intended purpose.

Definition 4. (*Access Purpose Compliance*) Let $\mathcal{PT}$ be a purpose tree. Let IP = $\langle$AIP, PIP$\rangle$ and AP be an intended purpose and an access purpose defined over $\mathcal{PT}$, respectively. AP is said to be compliant with IP according to $\mathcal{PT}$, denoted as AP $\Rightarrow_{\mathcal{PT}}$ IP, if and only if the following two conditions are satisfied:

1. AP $\notin \text{PIP}^{\updownarrow}$
2. AP $\in \text{AIP}^{\downarrow}$ □

Example 2. Let $\mathcal{PT}$ be the purpose tree in Figure 1, and let IP and AP be an intended purpose and an access purpose defined based on $\mathcal{PT}$, respectively.

1. Suppose IP = $\langle$\{General-Purpose\}, \{Third-Party\}$\rangle$. If AP = Marketing, then AP $\not\Rightarrow_{\mathcal{PT}}$ IP as Marketing $\in \text{PIP}^{\updownarrow}$. However, if AP = Admin, then AP $\Rightarrow_{\mathcal{PT}}$ IP as Marketing $\notin \text{PIP}^{\updownarrow}$ and Marketing $\in \text{AIP}^{\downarrow}$.
2. Suppose IP = $\langle$\{Admin, Purchase, Shipping\}, \{General-Purpose\}$\rangle$. Then no AP defined over PT is compliant with IP.
3. Suppose IP = $\langle$\{General-Purpose\}, $\emptyset\rangle$. Any AP defined over PT is compliant with IP.

4. ACCESS PURPOSE DETERMINATION

An access purpose is the reason for accessing a data item, and it must be determined by the system when a data access is requested. Evidently, how the system determines the purpose of an access request is crucial as the access decision is made directly based on the access purpose. In this section we present a possible method for determining access purposes. In our method, users are required to state their access purposes along with the data access requests, and the system validates the stated access purposes by ensuring that the users are indeed allowed to access data for the particular purposes. To facilitate the validation process, each user is granted authorizations for a set of access purposes, and an authorization for an access purpose permits users to access data with the particular purpose.

To ease the management of access purpose authorizations, users are granted authorizations through their roles; i.e., access purpose authorizations are granted to roles, not directly to individual users. This method has a great deployment advantage as many systems are already using RBAC mechanisms for the management of access permissions. This approach is also reasonable as access purposes can be granted to the tasks or functionalities over which roles are defined within an organization. However, using an RBAC mechanism for the management of both access permissions and access purposes may increase the complexity of the role engineering tasks. To address this problem, we introduce a simple extension to RBAC, which simplifies the role administration and also provides increased flexibility.

In this section, we first present our extended RBAC model and discuss the details of the access purpose authorization and verification based on this model. We do not discuss the general concepts of RBAC, assuming that readers are already familiar with them. For interested readers, we refer to [13, 15].

4.1 Role Attributes, System Attributes and Conditional Roles

As the role hierarchy is predefined for the access permission assignments, it is possible that the existing role definitions do not adequately specify the set of users to whom we wish to grant an access purpose. For instance, consider the purpose tree in Figure 1 and the role hierarchy in Figure 2. Suppose that we wish to allow some users in the E-Marketing team to access data for the purpose of "Service-Update". Under the "E-Marketing" role, there are two descendant roles: "E-Analysts" and "Writers". "E-Analysts" are the users who analyze the customer information and prepare the contents of emails, and "Writers" are the users who write and send out emails to customers. Note that these two roles are defined based on the access permission assignments in that the permissions to access the customer profiles are exclusively assigned to the "E-Analysts" role while the permissions to access the email addresses of the customers are exclusively assigned to the "Writers" role. However, as we want to assign the access purpose only to the users who are

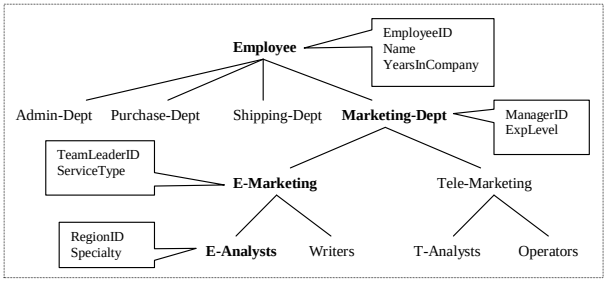


Figure 2: Role Hierarchy and Role Attributes

responsible for the specific task of sending out updated service information, neither the definition of "E-Analysts" or "Writers" matches our intention. Moreover, assigning the access purpose to the "E-Marketing" role is not desirable as it will allow all the users with the "E-Marketing" role to access data with the access purpose. An alternative solution is to split the "E-Marketing" role or the "E-Analysts" and the "Writers" roles into more specific roles. However, this method requires the reconstruction of both the user assignments and the permission assignments for the modified roles. Authorizing the access purpose on an individual basis is not an elegant solution either, as it does not utilize the existing role hierarchy and thus is not scalable. In order to address these issues, we introduce a new concept, *Conditional Roles*, which is based on the notion of *Role Attributes* and *System Attributes*.

Role attributes are the pre-assigned, specific descriptions associated with each role. The role attributes of each role are defined by system administrators at the time of role creation, and each role inherits the role attributes that are defined at the ancestor roles. When a user is assigned to a role, the values of the role attributes (both defined and inherited role attributes) are specified according to the relevant information of the particular user. Then when the user activates the role, the values of the role attributes are loaded and made available to the access control system until the user deactivates the role. The role attributes can be viewed as a cached user information that is relevant to the specific roles, and the role attribute values of a user should be updated if the user information changes.

Definition 5. (*Role Attributes*) Let $\mathcal{R}$ be the set of roles defined in the system. Every role $r \in \mathcal{R}$ is associated with a set of attributes, denoted by r.Attributes = {r.attr$_1$,..., r.attr$_n$}, that are defined for r or inherited from the ancestor roles of r. Each attribute r.attr$_i$ is associated with a set of values D_i, called the domain of r.attr$_i$. The values of the role attributes for a particular role r are specified within the domains for each user when a user is assigned to r. The specified role attribute values of a user u under r, denoted by $r(u)$.Attributes, are available to the access control system from the time the user activates r to the time the user deactivates r. □

Figure 2 shows an inverted tree role hierarchy and the role attributes for a hypothetical enterprise. The role in the root of the hierarchy, "Employee", represents the most general role (i.e., the junior-most role) and has three role attributes: *EmployeeID*, *Name*, and *YearsInCompany*. The roles below the root are constructed based on the principle of specialization, and they all inherit the role attributes of "Employee". For instance, the role attributes of the "Marketing-Dept" role include the role attributes of "Employee" as well as *ManagerID* and *YearsInDept*. Even though any role can be associated with a set of attributes, we only show the role attributes of a few selected roles in the figure for simplicity. We do not show the inherited role attributes for the same reason.

Using the role attributes we can assign an access purpose to a specific subset of users in the same role; that is, we can assign a particular access purpose to a role with a set of conditions that must be satisfied by the users of the role in order to access data with the access purpose. For instance, consider the previous scenario where we wish to assign the access purpose "Service-Update" to a particular group of users in the E-Marketing team. Provided that the attribute *ServiceType* is defined for the role "E-Marketing" as in Figure 2, we first set the value of this attribute to, say, "Update-Info" only for the users who are responsible for the task. Then by assigning the access purpose to the "E-Marketing" role with a condition saying that the user's value of the attribute *ServiceType* must be equal to "Update-Info", we can authorize the access purpose "Service-Update" only to the users whom we wish to grant the access purpose.

When authorizing access purposes, one may also need to consider the states of the system where the authorizations should become effective (or ineffective). For instance, we may wish to allow an access purpose to be used by a set of users only in a specific time interval (e.g., business hours) or only when the users are logged into specific machines (e.g., machine identification number). Thus, the system information that affects the access purpose authorizations must be clearly defined and available to the access control system. We refer to this system information as system attributes.

Definition 6. (*System Attributes*) Given a system $\mathcal{S}$, a set of attributes, denoted by $\mathcal{S}$.Attributes = {$\mathcal{S}$.attr$_1$,..., $\mathcal{S}$.attr$_n$}, is available to the access control system at all times. The system attributes are defined by system administrators for the application needs, and the values of the system attributes in a system state s, denoted by $\mathcal{S}(s)$.Attributes, specify the environment of the system in the state s. □

Now we introduce the notion of conditional role which utilizes both the role attributes and the system attributes to describe a specific set of users in a particular system environment.

Definition 7. (*Conditional Roles*) Let $\mathcal{R}$ be the set of roles defined in the system $\mathcal{S}$. A conditional role cr is defined as a 2-tuple ⟨r, C⟩, where $r \in \mathcal{R}$ and C is a finite propositional logic formula which may use the logical operators $\wedge$ and $\vee$. Each predicate in the formula is of the form x ϕ y, where x ∈ r.Attributes or x ∈ $\mathcal{S}$.Attributes, y is a constant, and $\phi \in \{<, \leq, >, \geq, =, \neq\}$. We say that a user u with the activated role r *belongs to* a conditional role $cr_i = \langle r_i, C_i \rangle$ in a system state s if and only if the following conditions are satisfied:

1. $r \in$ Descendants(r$_i$)
2. C$_i$(r, s), which substitutes the variables of C$_i$ with the values of $r(u)$.Attributes and $\mathcal{S}(s)$.Attributes, is evaluated to true. □

More specifically, if r = r$_i$ and the second condition is satisfied, we say that r *explicitly belongs to* cr$_i$. On the

other hand, if $r \in$ Descendants(r_i), but $r \neq r_i$, and the second condition is satisfied, then we say that r *implicitly belongs to* cr_i.

4.2 Access Purpose Authorization and Verification

As discussed in the previous section, access purposes are authorized to users through conditional roles. The use of conditional roles provides great flexibility in that the authorizations are sensitive to both the user profiles and the system environments. In this section, we formally define the access purpose authorization and its verification.

Definition 8. (*Access Purpose Authorization*) Let $\mathcal{PT}$ be a purpose tree, $\mathcal{P}$ be the set of purposes in $\mathcal{PT}$, and $\mathcal{R}$ be the set of roles defined in the system $\mathcal{S}$. An access purpose is authorized to a specific set of users by a 2-tuple $\langle ap, cr \rangle$, where $ap \in \mathcal{P}$ and cr is a conditional role defined over $\mathcal{R}$ and $\mathcal{S}$. □

Note that both the access purpose and the conditional roles are organized in hierarchies. Consequently, an access purpose authorization has its implications; i.e., authorizing an access purpose ap for a conditional role cr implies that the users belonging to cr either explicitly or implicitly are authorized to access data with ap as well as all the descendants of ap in the purpose tree. The definition of access purpose verification below captures these implications of access purpose authorizations.

Definition 9. (*Access Purpose Verification*) Let $\mathcal{PT}$ be a purpose tree and $\mathcal{P}$ be the set of purposes defined over $\mathcal{PT}$. $\mathcal{R}$ be the set of roles defined in the system $\mathcal{S}$. Given an access purpose ap and a role r activated by a user u, ap is valid for u under r if there exists an access purpose authorization $\langle \mathrm{ap}_i, \mathrm{cr}_j \rangle$, where $ap_i \in \mathcal{P}$ and $cr_j = \langle \mathrm{r}_j, \mathrm{C}_j \rangle$ is a conditional role defined over $\mathcal{R}$ and $\mathcal{S}$, satisfying the following two conditions:

1. $ap \in$ Descendants(ap_i)
2. The user u belongs to the conditional role cr_j either explicitly or implicitly. □

For example, the access purpose "Service-Update" in the previous scenario can be assigned to ⟨E-Marketing, (ServiceType = "Update-Info") ∧ (timeofday ≥ 9) ∧ (timeofday ≤ 17)⟩, assuming *timeofday* is defined as a system attribute. Then only the users who activate the role "E-Marketing" (or the two descendant roles) with their *ServiceType* attribute equal to "Update-Info" can access data with the purpose of "Service-Update" between 9 am and 5 pm.

5. HIERARCHICAL DATA MODEL AND INTENDED PURPOSE LABELING

As in [3], our access control model is based on purpose metadata associated with data objects. That is, every data object is tagged with intended purpose labels, and the data access is controlled according to these labels. In this section, we present a general model for hierarchical data and consider various key issues on labeling the hierarchical data objects with intended purposes in the model.

5.1 Hierarchical Data Model

In most practical systems, data elements are organized in hierarchical structures. In typical file systems the basic data objects (i.e., files) are organized and stored using type information (e.g., file extensions) and directory hierarchies, and in XML data elements are organized in XML documents which have tree-like data structures. A relational data model can be considered hierarchical to some extent as it stores data items in tables which are composed of multiple columns and rows. We now provide a general hierarchical data model which can relate to many variations of hierarchical data management systems.

Definition 10. (*Hierarchical Data Model*) A hierarchical data model $\mathcal{H}$ is represented as a 5-tuple $\langle \mathcal{T}_H, \mathcal{O}_H, \mathcal{IO}_H, \mathcal{SO}_H, \mathcal{RO}_H \rangle$, where

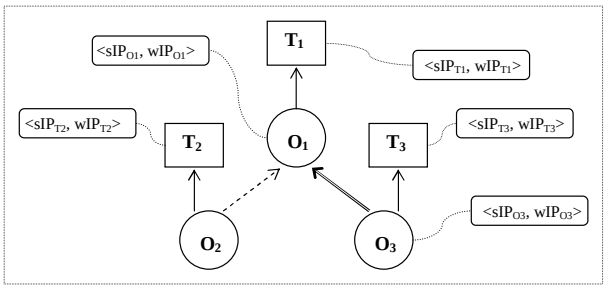
1. $\mathcal{T}_H$ is a set of types.
2. $\mathcal{O}_H$ is a set of objects.
3. $\mathcal{IO}_H : \mathcal{O}_H \longrightarrow \mathcal{T}_H$ is a function that assigns a type to each object. When $\mathcal{IO}_H(o) = t$, we say that the object o is an *instance-of* the type t and we write $o \rightarrow t$.
4. $\mathcal{SO}_H : \mathcal{O}_H \longrightarrow \mathcal{O}_H$ is a function that assigns a parent to each object. When $\mathcal{IO}_H(o_1) = o_2$, we say that o_2 is *the parent* of o_1 and o_1 is a *subelement-of* of o_2, and we write $o_1 \Rightarrow o_2$.
5. $\mathcal{RO}_H$ is a function, denoted as $\mathcal{O}_H \dashrightarrow \mathcal{O}_H$, mapping the *reference-of* relations between objects.

An auxiliary function, Nodes($\mathcal{H}$) returns a set containing all of the types and the objects in $\mathcal{H}$; i.e., Nodes($\mathcal{H}$) = $\mathcal{T}_H \cup \mathcal{O}_H$. □

For example, consider XML. An XML document O_i, which is an object, is created based on a DTD document (or an XML schema) T_j, which is a type; i.e., $\mathrm{O}_i \rightarrow \mathrm{T}_j$. Also, an XML document can contain an element E which consists of other subelements, say E_s and E_t, and in this case $\mathrm{E}_s \Rightarrow \mathrm{E}$ and $\mathrm{E}_t \Rightarrow \mathrm{E}$ and we say that E is the *parent* of E_s and E_t. An XML document O_i can also include a links that point to an external document O_j, and this relation is captured by the reference-of relations; i.e., $\mathrm{O}_j \dashrightarrow \mathrm{O}_i$.

5.2 Intended Purpose Labeling

One of the main issues on designing a data labeling scheme is at what level of granularity data should be associated with labels. In order to optimize the metadata storage space, labeling at a coarse granularity seems to be an appropriate choice. However, a coarse-grained labeling scheme means that data access will also be controlled at a coarse granularity. This can be highly undesirable in many private or secure data management systems. Especially in private data management systems, the granularity of access control models must be fine as access control decisions are determined based on privacy requirements and data owner's privacy preferences, both of which are typically complex and fine-grained. Explicitly assigning intended purpose labels to every type and every object in the system is not an ideal solution either, as it requires storing a great deal of redundant information and it increases the burden of the intended purpose label management. A solution that is both storage-efficient and fine-grained is to allow explicit intended purpose labeling at any granularity level and at the same time to make use of the notion of implicit intended purpose, which is analogous to the notion of *implicit authorization* [12]. For instance,



□: types, O: objects, →: instance-of relations,
⇒: subelement-of relations, --> : reference-of relations

Figure 3: Data Hierarchy and Intended Purpose Labels

labeling a type with an intended purpose implies that the intended purpose will apply to all objects that are instances of the type. Similarly, assigning an intended purpose to an object implies that the intended purpose will apply to all the subelements of the object. This concept of implicit intended purpose provide another advantage in that labeling a type (or an object) will ensure that all the instances of the type (or all the subelements of the object) will also be governed by the intended purpose. However, we note that the implication of intended purposes does not apply to the reference-of relations. The reason is that the referred objects are external to the referring objects; i.e., the referred objects are not part of the referring objects. Thus, they do not strictly share the same data hierarchy.

Bringing in the notion of implicit intended purpose raises a problem of possible conflicts. We remind readers that by our definitions in Section 3, an intended purpose contains an allowed intended purpose and a prohibited intended purpose, each of which is a set of purposes. As an intended purpose consists of both positive and negative semantics, it is possible that intended purposes have conflicts to each other. For instance, suppose an object O which is explicitly labeled with an intended purpose IP_O is an instance of a type T that is labeled with another intended purpose IP_T. Suppose that $\text{IP}_O = \langle$Admin, Marketing$\rangle$ and $\text{IP}_T = \langle$Marketing, Admin$\rangle$; i.e., IP_O states that O can be accessed for the "Admin" purpose but cannot be accessed for the "Marketing" purpose, while IP_T states that any instance of T, including O, can be accessed for the "Marketing" purpose but cannot be accessed for the "Admin" purpose. These two intended purposes clearly conflict; e.g., the access to O for the "Admin purpose" is explicitly allowed by IP_O but at the same time implicitly prohibited by IP_T. We address this issue by introducing two types of intended purposes in our model: *Strong Intended Purpose* and *Weak Intended Purpose*. A strong intended purpose of a type (or an object) cannot be overridden by intended purposes associated with the instances of the type (or subelements of the object) while a weak intended purpose can be overridden. We emphasize that our choice of the conflict resolution policy is very flexible and thus suitable for most private data management systems where both privacy requirements and user preferences should be expressed and enforced by the access control system. In next definition, we formally define the notion of intended purpose label.

Definition 11. (*Intended Purpose Label*) Let $\mathcal{PT}$ be a purpose tree and $\mathcal{P}$ be the set of all purposes in $\mathcal{PT}$. Let $\mathcal{IP}$ be the set of all possible intended purposes defined over $\mathcal{P}$; i.e., $\mathcal{IP} = \{\langle \text{AIP, PIP}\rangle | \text{AIP} \subseteq \mathcal{P}, \text{PIP} \subseteq \mathcal{P}\}$.

An intended purpose label is a 2-tuple $\langle$sIP, wIP$\rangle$, where sIP $\in \mathcal{IP}$ represents a strong intended purpose and wIP $\in \mathcal{IP}$ a weak intended purpose. An intended purpose label $\ell = \langle\langle$sAIP, sPIP$\rangle$, $\langle$wAIP, wPIP$\rangle\rangle$ is said to be *well-formed* if the following properties hold:

1. $(\text{sAIP}^{\downarrow} - \text{sPIP}^{\updownarrow}) \cap \text{wPIP}^{\updownarrow} = \emptyset$, that is, purposes allowed by the strong intended purpose cannot be prohibited by the weak intended purpose.
2. $\text{sPIP}^{\updownarrow} \cap (\text{wAIP}^{\downarrow} - \text{wPIP}^{\updownarrow}) = \emptyset$, that is, purposes prohibited by the strong intended purpose cannot be allowed by the weak intended purpose. □

These two properties ensure consistency(i.e., no conflicts) in an intended purpose label. In our data labeling model, any node(either a type and an object) in a data hierarchy can be associated with a well-formed intended purpose label. Note that the consistency properties are also enforced between the intended purpose labels of any two nodes in the same hierarchy. Now we formally define our data labeling model.

Definition 12. (*Intended Purpose Labeling*) Let $\mathcal{PT}$ be a purpose tree and $\mathcal{P}$ be the set of all purposes in $\mathcal{PT}$. Let $\mathcal{IP}$ be the set of all possible intended purposes defined over $\mathcal{P}$. Let $\text{WL}(\mathcal{PT})$ be the set of all well-formed labels in $\mathcal{PT}$. Let $\mathcal{H}$ be a data hierarchy, and $\text{Nodes}(\mathcal{H})$ be the set of all the nodes in $\mathcal{H}$. An intended purpose labeling function, $\mathcal{L}$: $\text{Nodes}(\mathcal{H}) \longrightarrow \text{WL}(\mathcal{PT})$, associates each node in $\mathcal{H}$ with a well-formed intended purpose label. We say that the intended purpose label of a node $D \in \mathcal{H}$, $\mathcal{L}(D) = \langle\langle \text{sAIP}_D, \text{sPIP}_D\rangle, \langle \text{wAIP}_D, \text{wPIP}_D\rangle\rangle$, is *consistent* with $\mathcal{L}$ if for every node $A \in \mathcal{H}$ that is an ancestor of D, the following properties hold for the intended purpose label of A, $\mathcal{L}(A) = \langle\langle \text{sAIP}_A, \text{sPIP}_A\rangle, \langle \text{wAIP}_A, \text{wPIP}_A\rangle\rangle$:

1. $(\text{sAIP}_A{}^{\downarrow} - \text{sPIP}_A{}^{\downarrow}) \cap \text{sPIP}_D{}^{\updownarrow} = \emptyset$, that is, purposes strongly allowed at a node cannot be strongly prohibited at a descendant node.
2. $\text{sPIP}_A{}^{\downarrow} \cap (\text{sAIP}_D{}^{\downarrow} - \text{sPIP}_D{}^{\updownarrow}) = \emptyset$, that is, purposes strongly prohibited at a node cannot be strongly allowed at a descendant node. □

Malformed or inconsistent intended purpose labels should be prevented by ensuring the properties whenever a new intended purpose label is introduced to the system or any existing intended purpose label is modified.

6. INTENDED PURPOSE INFERENCE AND QUERY COMPLIANCE

In this section we discuss two key issues in access control based on the intended purpose labels. First, we describe how the intended purpose of an object is inferred from the implicit and explicit intended purpose labels. Then we introduce the notion of query compliance, on which our access control model is based.

6.1 Intended Purpose Inference

As each object is associated with both the explicit intended purpose of the object and all the implicit intended purposes inherited to the object, the intended purpose that

Structure *IP*

1. sAIP: a set of strong allowed intended purposes
2. sPIP: a set of strong prohibited intended purposes
3. wAIP: a set of weak allowed intended purposes
4. wPIP: a set of weak prohibited intended purposes

End;

Function *Merge_IPs(IP ip_1, IP ip_2)*
Input: two intended purposes to be merged, ip_1 and ip_2
Output: the merged intended purpose, which ip_2 is merged over ip_1; i.e., ip_2 overrides ip_1 in case of any conflict.

1. if (ip_1 is empty) then
2. return ip_2;
3. else (if ip_2 is empty) then
4. return ip_1;
5. else
6. merged = create a new IP;
7. merged.sAIP = $(ip_1.sAIP)^{\downarrow} \cup (ip_2.sAIP)^{\downarrow}$;
8. merged.sPIP = $(ip_1.sPIP)^{\updownarrow} \cup (ip_2.sPIP)^{\updownarrow}$;
9. merged.wAIP = $(ip_1.wAIP)^{\downarrow} \cup (ip_2.wAIP)^{\downarrow}$;
10. merged.wPIP = $((ip_1.wPIP)^{\updownarrow} - (ip_2.wAIP)^{\downarrow}) \cup (ip_2.wPIP)^{\updownarrow}$;
11. return merged;
12. end if;

End;

Function *Get_Effective_IP* (Object o)
Input: an object o
Output: the effective intended purpose of o

1. if (o.parent is null) then
2. return Merge_IP(o.type.IP, o.IP);
3. else
4. IP temp = Get_Effective_IP(o.parent);
5. temp = Merge_IP(temp, o.type.IP);
6. return Merge_IP(temp, o.IP);
7. end if;

End;

Figure 4: Intended Purpose Inference(IPI) Algorithm

effectively governs the usage of the object, which we call the *Effective Intended Purpose*, should be inferred from both types of intended purposes. For example, consider the data hierarchy in Figure 3. The object O_1, explicitly labeled with $\langle sIP_{O1}, wIP_{O1}\rangle$, also inherits the intended purpose $\langle sIP_{T1}, wIP_{T1}\rangle$ from T_1. Similarly, the object O_3 inherits the intended purposes from T_3, O_1 and T_1. Note, however, that O_2, which is not labeled, inherits the intended purpose from T_2, but not from O_1 nor T_1, as the intended purposes are not inherited through the reference-of relation. We also note that in order to correctly override the weak intended purposes, the order of which effective intended purposes are inferred must be top-down. For instance, consider object O_1. The effective intended purpose of O_1 is inferred by merging the intended purpose of O_1 over the intended purpose of T_1; i.e., the weak intended purpose of T_1 gets overridden by the intended purpose of O_1. The effective intended purpose of O_3 is inferred in a very similar way. First, the effective intended purpose of O_1 is inferred, and then the intended purpose of T_3 is merged over the effective intended purpose of O_1. Let us call this intermediate result EIP′. Then the effective intended purpose of O_3 is inferred by merging the intended purpose of O_3 over EIP′. We provide the pseudocode Intended Purpose Inference(IPI) algorithm in Figure 4. The algorithm recursively applies a derivation process to determine the effective intended purpose for each data object.

Next theorem establishes the correctness of IPI algorithm.

Theorem 1. If all intended purpose labels in the system are well-formed and consistent, then the effective intended purposes generated by IPI Algorithm are also well-formed and consistent. □

PROOF. As the algorithm recursively merges intended purpose labels of types and objects from top to bottom, we only need to show that the theorem holds with respect to a process of merging two intended purpose labels. We prove only the well-formedness here, as the consistency can be proved analogously.

As the simplest case, if at least one of the labels is empty[1] (i.e., every set in the label is $\emptyset$), then the resulting intended purpose after the merge is well-formed.

Now suppose we have two non-empty labels. To facilitate our illustration, let O be an object, which is labeled with $\langle\langle sAIP_O, sPIP_O\rangle, \langle wAIP_O, wPIP_O\rangle\rangle$. Let A be the direct ancestor of O, which is labeled with $\langle\langle sAIP_A, sPIP_A\rangle, \langle wAIP_A, wPIP_A\rangle\rangle$. Then the effective intended purpose of O, $EIP_O = \langle\langle sAIP_E, sPIP_E\rangle, \langle wAIP_E, wPIP_E\rangle\rangle$ is inferred by IPI algorithm as $\langle\langle(sAIP_A \cup sAIP_O), (sPIP_A \cup sPIP_O)\rangle, \langle(wAIP_A \cup wAIP_O), [(wPIP_A - wAIP_O) \cup wPIP_O]\rangle\rangle$, and EIP_O satisfies the following properties.

1. $(sAIP_E{}^{\downarrow} - sPIP_E{}^{\updownarrow}) \cap wPIP_E{}^{\updownarrow} = [(sAIP_A{}^{\downarrow} \cup sAIP_O{}^{\downarrow}) - (sPIP_A{}^{\downarrow} \cup sPIP_O{}^{\updownarrow})] \cap [(wPIP_A{}^{\updownarrow} - wAIP_O{}^{\downarrow}) \cup wPIP_O{}^{\updownarrow}] = \emptyset$, as the following pairs are disjoint by Definition 11; $(sAIP_A{}^{\downarrow}, wPIP_A{}^{\updownarrow})$ and $(sAIP_O{}^{\downarrow}, wPIP_O{}^{\updownarrow})$.

2. $sPIP_E{}^{\updownarrow} \cap (wAIP_E{}^{\downarrow} - wPIP_E{}^{\updownarrow}) = (sPIP_A{}^{\updownarrow} \cup sPIP_O{}^{\updownarrow}) \cap [(wAIP_A{}^{\downarrow} \cup wAIP_O{}^{\downarrow}) - ((wPIP_A{}^{\updownarrow} - wAIP_O{}^{\downarrow}) \cup wPIP_O{}^{\updownarrow})] = \emptyset$, as the following pairs are disjoint by Definition 11; $(sPIP_A{}^{\updownarrow}, (wAIP_A{}^{\downarrow} - wPIP_A{}^{\updownarrow}))$ and $(sPIP_O{}^{\updownarrow}, (wAIP_O{}^{\downarrow} - wPIP_O{}^{\updownarrow}))$.

Therefore, EIP_O is well-formed by Definition 11. □

6.2 Data Query and Query Compliance

Users request an access to data by issuing a query. Query specifications (i.e., query languages) vary from a system to another, but how queries are specified is irrelevant to our discussion. We also do not consider how queries are processed in detail; e.g., how the data requested by a query is located and fetched. Thus, in this paper we consider only queries of the simplest form. Note that the queries of this form are very rudimentary, but they are sufficient in the scope of our discussion.

Definition 13. (*Query*) Let $\mathcal{PT}$ be a purpose tree, $\mathcal{P}$ be the set of purposes in $\mathcal{PT}$. Let o be an object in $\mathcal{O}$ and ap be an access purpose in $\mathcal{P}$. A query Q is a request for a particular data object o with a particular access purpose ap and denoted as a 2-tuple $\langle o, ap\rangle$. □

As discussed earlier, users are required to state their access purpose along with their data request, and the system validates the stated access purpose by ensuring that the users are indeed allowed for the access purpose. If the validation fails, the request is rejected without being further processed. If the validation succeeds, then the system

[1]This is case where a type or an object is not labeled with an intended purpose. Note that a "empty label" is both well-formed and consistent according to Definitions 11 and 12.

fetches the requested data object and checks whether or not the access purpose is compliant to the intended purpose of the data object. Note that intended purposes are inherited as implicit intended purposes through the instance-of relations or the subelement-of relations. Thus, in order to determine whether to grant a data access for a certain access purpose, the system must consider both the intended purpose explicitly associated with the data object and the implicit intended purposes inherited to the data object. That is, the system accepts a query if and only if the access purpose of the query is compliant to the effective intended purpose of the requested data object. We capture this concept with the notion of query compliance, which is an extension of the access purpose compliance in Section 3.

Definition 14. (*Query Compliance*) $\mathcal{PT}$ be a purpose tree, $\mathcal{P}$ be the set of purposes in $\mathcal{PT}$. Let $Q = \langle o, ap \rangle$ be a query accessing an object o in $\mathcal{O}$ with the access purpose ap in $\mathcal{P}$. Let $\text{EIP}_o = \langle \text{sEIP}_o, \text{wEIP}_o \rangle$ be the effective intended purpose of o. Q is said to be compliant with the intended purpose of o with respect to $\mathcal{PT}$, denoted as $Q \Rrightarrow_{PT} o$, if and only if $ap \Rightarrow_{PT} \text{sEIP}_o$ or $ap \Rightarrow_{PT} \text{wEIP}_o$. □

Only if the access purpose is successfully validated and the query is compliant with the effective intended purpose of the requested data object, the query is accepted and the result is returned to the user.

7. CONCLUSIONS AND FUTURE WORK

In this paper we presented a purpose-based access control suited for hierarchical data. We also proposed an efficient method for determining access purposes, which uses the notions of role attributes and conditional roles, another original contribution of the paper. Our future work includes implementing an access control system based on our work specifically for XML. On the implemented system, performance and storage efficiency will be measured and evaluated. We also plan on devising a high level language for purpose-oriented privacy policy which can be used to automatically manage the intended purposes of data. Compatibility issues with P3P will also be investigated. Another possible future work is to extend our model to deal with other elements of privacy such as obligations and complex conditions. Even though these elements are not trivial to handle, they are essential part of privacy protection.

8. REFERENCES

[1] R. Agrawal, J. Kiernan, R. Srikant, and Y. Xu. Hippocratic databases. In *The 28th International Conference on Very Large Databases (VLDB)*, 2002.

[2] D. E. Bell and L. J. LaPadula. Secure computer systems: mathematical foundations and model. Technical report, MITRE Corporation, 1974.

[3] J. Byun, E. Bertino, and N. Li. Purpose-based access control for privacy protection in relational database systems. Technical Report 2004-52, Purdue University, 2004.

[4] F. Chen and R. Sandhu. Constraints for role-based access control. In *the first ACM Workshop on Role-based access control*, 1996.

[5] F. T. Commision. Children's online privacy protection act of 1998. Available at www.cdt.org/legislation/105th/privacy/coppa.html.

[6] D. Denning, T. Lunt, R. Schell, W. Shockley, and M. Heckman. The seaview security model. In *The IEEE Symposium on Research in Security and Privacy*, 1998.

[7] C. Goh and A. Baldwin. Towards a more complete model of role. In *The 3rd ACM workshop on Role-based access control*, 1998.

[8] A. Kumar, N. Karnik, and G. Chafle. Context sensitivity in role-based access control. In *ACM SIGOPS Operating Systems Review*, July 2002.

[9] K. LeFevre, R. Agrawal, V. Ercegovac, R. Ramakrishnan, Y. Xu, and D. DeWitt. Disclosure in hippocratic databases. In *The 30th International Conference on Very Large Databases (VLDB)*, Aug. 2004.

[10] U. S. D. of Health. Health insurance portability and accountability act of 1996. Available at www.hep-c-alert.org/links/hippa.html.

[11] U. S. D. of Justice. The federal privacy act of 1974. Available at www.usdoj.gov/foia/privstat.htm.

[12] F. Rabitti, E. Bertino, W. Kim, and D. Woelk. A model of authorization for next-generation database systems. In *ACM Transactions on Database Systems (TODS)*, Mar. 1991.

[13] R. Sandhu. Role hierarchies and constraints for lattice-based access control. In *the European Symposium on Research in Computer Security*, 1996.

[14] R. Sandhu and F. Chen. The multilevel relational data model. In *ACM Transaction on Information and System Security*, 1998.

[15] R. Sandhu, D. Ferraiolo, and R. Kuhn. The nist model for role-based access control: Towards a unified standard. In *the fifth ACM workshop on Role-based access control*, 2000.

[16] R. Sandhu and S. Jajodia. Toward a multilevel secure relational data model. In *ACM International Conference on Management of Data (SIGMOD)*, 1991.

[17] World Wide Web Consortium (W3C). *Platform for Privacy Preferences (P3P)*. Available at www.w3.org/P3P.