2008 International Symposium on Parallel and Distributed Processing with Applications

Dynamic Purpose-based Access Control

Huanchun PENG, Jun GU and Xiaojun YE
Key Laboratory for Information System Security, Ministry of Education
School of Software, Tsinghua University, 100084 Beijing, China
{phc07,j-gu07}@mails.tsinghua.edu.cn,yexj@tsinghua.edu.cn

Abstract

This article presents a new approach for privacy preserving access control based on RBAC. The separation of authorization of access purpose from access decision improves the flexibility of private data control. A key feature of this approach is dynamic. The access purpose is determined in a dynamic manner, based on subject attributes, context attributes and authorization policies. Intended purposes are dynamically associated with the requested data object during the access decision. Finally, we give the algorithm to achieve the compliance computation between the access purpose and intended purposes.

1. Introduction

With the increasingly extensive application of information technologies in people's daily life, the protection of users' private data has become a major focus of research in the field of information security.

Privacy can be defined as the right of an individual to decide when, how, and to what extent he/she would like to share his/her information [2]. Conventional access models are not designed to enforce privacy policies and barely meet privacy protection requirements [5][12], particularly, purpose binding. That is because the traditional model of access control is based on controlling the specific operations of the user on relevant data [11].What counts to privacy policies, however, is the specific purpose for which the data has been collected and used [4][9].

The OECD Guidelines [6] are the most extensively adopted as well as the most influential principles of privacy protection guidelines, according to which, the support for purposes, conditions and obligations is the key to the transition from traditional access models to privacy preserving access control models. Purposes describe for what reasons data is collected or used. Obligations are actions to be performed after some operations on data objects, which are necessary in some cases. Conditions are prerequisites which should be met before or when any actions can be executed [7].

For the current extensive use of RBAC model [11] in database systems, we believe that it is highly possible to realize the access control model for privacy protection, which supports purposes, conditions and obligations, on the basis of RBAC model. In the first place, most database enterprisers have integrated RBAC model into their database management systems, and the privacy-aware models realized by extending RBAC model can be easily deployed into these systems [7]. Second, the mandatory implementation of privacy policies should be guaranteed, otherwise enterprisers may consciously or unconsciously violate their privacy policies in the actual practice. This paper presents a model of *Dynamic Purpose-based Access Control (DPBAC)*, which has the following features.

1. The separation of access purpose authorization from access decision. The data provider expresses his/her own privacy preferences through setting intended purposes, while the data owner is responsible for working out the policies for authorization of access purpose.

2. Rather than statically bound to data objects as labels, intended purposes are dynamically associated with the requested data objects during the access decision according to the well-designed hierarchy of private metadata.

3. DPBAC model determines the access purpose and purpose compliance in a dynamic manner based on subject attributes and context attributes of the system.

The rest of this paper is organized as follows. In Section 2, we briefly introduce some related work. In Section 3, we will give a formal description of DPBAC model. Section 4 presents the mechanism of realizing authorization of access purpose in DPBAC model. In Section 5, we will discuss how DPBAC model realizes the mechanism of access decision. Finally, Section 6 presents our future work.

978-0-7695-3471-8/08 $25.00 © 2008 IEEE
DOI 10.1109/ISPA.2008.80

2. Related Work

Currently Purpose Based Access Control (PBAC) [4], Fine Grained Access Control (FGAC) [1][10] and Extensible Access Control Markup Language (XACML) [8] are widely regarded as the most important privacy preserving access control methods.

The PBAC model builds the purpose hierarchy on both *intended purposes*, which specify the intended usage of data, and *access purposes*, which specify the purposes for which data elements are requested. When the user hands in a request, the access control engine would verify whether the access purpose complies with the intended purposes of user's requested data, and permit the access if it does or otherwise deny the request. The key feature of PBAC is that it supports explicit prohibitions and organizes purposes in a hierarchy structure. What's more, privacy preserving on different granularities can be achieved by associating intended purposes with an entire table, each column within a table, each tuple within a table, or each attribute within a tuple. However, the access purposes are declared explicitly by users themselves and thus lack flexibility, and the storage of privacy metadata is based on labeling schemes which leads to a large storage overhead.

In 2005, Agrawal et al. presented FGAC for achieving purpose-based privacy protection. The feature of this model is that external privacy policies which are described in P3P or EPAL can be interpreted to internal fine grained restrictions stored in the *privacy policies table* and *privacy authorizations table* by the policy transformer [2]. Fine grained restrictions can be applied to columns within a table, tuples within a table, or attributes within a tuple. The query engine would analyze the FGAC restrictions in the process of execution, and then change the inquired object (table) to a privacy authorization view to protect private personal information.

In XACML model [8], to achieve privacy protection, the requested resource is bound with an attribute called *urn:oasis:names:tc:xacml:2.0:resource:purpose*, which specifies the intended purpose of the resource, and the action element is bound with an attribute called *urn:oasis:names:tc:xacml:2.0:action:purpose*, which specifies the access purpose. These two attributes are matched up by adding the rule of *matching-purpose* to policies. However, this model does not take the hierarchy structure of purposes into consideration, and its implementation is rather sketchy.

3. DPBAC Model

RBAC model is a milestone in the field of access control models and become a NIST standard [11]. It is initially designed to satisfy the need of simplifying the authorization management and directly presenting access control policies. The key concept of RBAC model is *role*, which represents certain specific job functions in organizations. The permission of performing certain operations on certain data is assigned to roles.

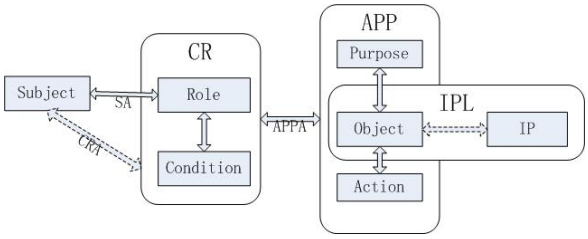

Figure 1. DPBAC Model

Currently at the research stage of privacy protection, the importance of *purpose*, *condition* and *obligation* has been admitted by most scholars. Based on RBAC model, *Dynamic Purpose-based Access Control(DPBAC)* model extends mainly in the following aspects.

1. The access permission is no longer a 2-tuple ⟨Object, Action⟩, but a 3-tuple ⟨Object, Action, Access Purpose⟩, which is called the access purpose permission.

2. The access purpose permission is not assigned to static roles but to *Conditional Roles (CR)*. Subjects dynamically activate conditional roles according to their context attributes during the access process.

Like P-RBAC model [7], DPBAC model also uses conditions and purposes to extend RBAC model. However, DPBAC model does not simply add these components to RBAC to constitute new privacy permissions. In DPBAC model, the privacy preserving access purpose is not explicitly associated with subjects, but given to conditional roles by access purpose permissions. The activation of conditional roles is dynamically implemented based on the attributes of subjects and the state of the system. The privacy preserving access control is not only achieved by authorization. Gaining the access purpose permission does not mean that you can directly perform operations on requested objects. The data object is dynamically bound with its intended purposes according to the corresponding privacy metadata. After the purpose compliance process, only the objects which are purpose-compliant can be returned to the subjects. Fig 1 shows the DPBAC model. The broken lines represent dynamic associations.

In DPBAC model, the entity *Subject* can be a user, an application program or a stored procedure. The entity *Role* represents the working function or working title assigned within the organization according to different authorities and obligations. The entity *Object* stands for the data which

the subject requests, and can be abstracted as data set. The entity *Action* signifies certain operation that the subject wants to perform on the object. The entity *Condition* is a predicate logic expression that role attributes and system attributes should satisfy. The entity *Purpose* presents all the possible access purposes in the system; while *IP* signifies the intended purposes bonded with each data object. The formalized definition of DPBAC is shown as follows.

Definition 1 (*DPBAC Model*).

- *Subject, Role, Action, Object, Purpose, Condition* represent the set of subjects, roles, actions, objects, purposes and conditions.
- $IP=\{\langle aip,pip\rangle | aip\subseteq P, pip\subseteq P\}$ is the set of object's intended purposes. *aip* signifies the object's permitted intended purposes; while *pip* represents the object's forbidden intended purposes [4].
- $CR=\{\langle r,c\rangle | r\in Role,\ c\in C\}$ is the set of roles with condition expression.
- $APP=\{\langle o,act,ap\rangle | o\in Object,\ act\in Action,\ ap\in Purpose\}$ is the set of access purpose permissions.
- $IPL=\{\langle o,ip\rangle | o\in Object,\ ip\in IP\}$ represents the set of data objects and their bound intended purposes.
- $RH\subseteq Role\times Role$ is a partial order on roles, called the inheritance relation, which is written as $\preceq$.
- $PT\subseteq Purpose\times Purpose$ is a partial order on purposes (generalization/specialization) shown in the purpose tree, which is written as $\preceq$.
- *Subject Assignment* $SA\subseteq Subject\times Role$ is a many-to-many mapping relation between subjects and their assigned roles.
- *Condition Role Assignment* $CRA\subseteq Subject\times CR$ is a many-to-many mapping relation between subjects and their conditional roles.
- *Access Purpose Permission Assignment* $APPA\subseteq CR\times APP$ is a many-to-many mapping relation between conditional roles and access purpose permissions. It signifies that the operation that certain role performs on certain object under certain conditions is based on certain access purpose.
- *Purpose Compliance* $PC\subseteq APP\bowtie IPL$ is a one-to-one relationship between each access purpose permission and data object as well as its bound intended purposes.

We give following function definitions to facilitate the discussion of DPBAC.

- *assigned_role: Subject*$\rightarrow 2^{Role}$, the mapping of a subject *s* onto a set of roles. Formally,

  assigned_role(s)$=\{r\in Role | \langle s,r\rangle \in SA\}$
- *active_condition_roles: Subject*$\rightarrow 2^{CR}$, the mapping of a subject *s* onto a set of condition roles.
- *access_purpose_authorization: CR*$\rightarrow$*APP*, the mapping of a condition role *cr* onto access purpose permissions. Formally,

  access_purpose_authorization(cr)$=\{app\in APP | \langle cr,app\rangle \in APPA\}$
- *purpose_binding: Object*$\rightarrow$*IP*, the mapping of a data object *o* onto intended purposes *ip* , which means finding the bound intended purposes of the object.
- *purpose_compliance*: $AP\times IP\rightarrow\{$True, False $\}$, is used to determine the compliance between the access purpose and the object's intended purposes [4]. Formally,

  purpose_compliance(ap,ip)=True iff ap$\in AIP^*\wedge$ap$\notin PIP^*$ among which

  $AIP^*=\bigcup_{aip_j\in AIP}$Decendants$(aip_j)$

  $PIP^*=(\bigcup_{pip_k\in PIP}$Ancestors$(pip_k))\cup(\bigcup_{pip_k\in PIP}$ Decendants$(pip_k))$[1]

The determination of the access purpose is based on enabling conditional roles dynamically and the association between access purpose permissions and conditional roles. The advantages are: 1) avoiding security fault caused by giving subjects access purposes directly; 2) simplifying the management of access purpose permissions through policies. During the access decision process, the system binds the requested data with its intended purposes according to privacy metadata, and sends the data whose intended purposes are compliant with the access purpose to the requester, which prevents privacy information from divulgence. The following two parts of the paper will present the *access purpose authorization* and *access decision for privacy preserving* respectively.

4. Access Purpose Authorization

There are three possible methods for access purpose determination [4]. First, subjects can be required to state their access purposes explicitly along with the requests. It is relatively easy to implement, but requires complete trust on the identity of subjects, which is not suitable for an open

[1] Ancestors(P_i) is a set composed of all the ancestors of P_i in the purpose tree including P_i itself. Descendants (P_i) is a set composed of all the descendants of P_i in the purpose tree including P_i itself.

environment. Most privacy preserving access control models are based on this method. Second, the system registers a special access purpose for each application or stored-procedure. This method, however, cannot be used for complex applications or stored-procedures for the reason that requesters may access data objects for multiple access purposes. Lastly, access purposes can be dynamically determined based on attributes of subjects and the context of the system, in addition to requested objects and operations. As privacy policies need to determine who can access what, under what conditions and for what purposes, it can be implemented in our model by assigning access purpose permissions, which represent what can be accessed for what purposes to condition roles, which represent roles under certain conditions.

The step of enabling condition roles plays an important part in the whole process of access purpose authorization, for the reason that access purpose permissions are directly assigned to condition roles. The difference between role activation in RBAC and condition role activation in our model is that enabling a condition role needs dynamic condition evaluation based on subject attributes and system context. A condition role is a 2-tuple $CR=\langle r, c\rangle$, where $r \in R$ represents the predefined static role, which is similar to the role in RBAC, and $c \in C$ represents the conditions that the values of role attributes and context attributes must meet in the session. We first give the formal definition of conditions, and then give the complete algorithm for condition roles activation.

Definition 2 (*Condition C*).

Let the sets *SystemAttribute*, *RoleAttribute* represent the sets of predefined system attributes and role attributes, and X=*SystemAttribute*∪*RoleAttribute*. Each variable $x \in$ X has a finite domain of possible values, denoted as *Domain(x)*. A predication *ac* defined over X has the form *(x* op_r *value)* where $x \in$X, *value*∈*Domain(x)* and $op_r \in \{=, \neq, <, >, \leq, \geq\}$. The condition *C* over *X* can be defined recursively as follows.

1. A predication *ac* can be a condition statement with the form *(x* op_r *value)*, which is called an atomic condition;
2. If ac_i and ac_j are conditions, then $ac_i \wedge ac_j$ is a condition.[2]

As mentioned above, access purpose permissions are assigned to condition roles. In DPBAC, the relations between subjects and static roles are predefined by security administrators. Therefore when a request arrives, enabling condition roles involves the following steps:

[2]It is unnecessary to define $ac_i \vee ac_j$, because we can consider *r* with condition $ac_i \vee ac_j$ as two condition roles. If a subject can activate $\langle r, ac_i \vee ac_j\rangle$, we will define two condition role: $\langle r, ac_i\rangle$ and $\langle r, ac_j\rangle$ for it

Algorithm for enabling condition roles

Input: *subject* is the one who requests an access
. *sys* is current system attributes
Output: condition role set *enabled_CR* activated by subject
1). let set *enabled_CR* be a empty set of condition roles
2). *subject.Attribute*← the attributes of *subject*
3). *Role*←*assigned_roles(subject)*
4). for each *role*∈*Role*
5). initial *role.Attribute* with *subject.Attribute*
6). for each *cr*∈*CR* when *cr*=⟨c,r⟩
7). if *role*=*r*
8). if *check_condition(c,role,sys)*
10). then *enable_CR*=*enable_CR*∪{*cr*}
11). return *enable_CR*

Figure 2. Algorithm to enable condition roles

1. Finding static roles for the subject;
2. Finding all the candidate condition roles for the subject. If the value of attribute *r* in the condition role $\langle c,r\rangle$ equals with static roles of the subject in the session, the $\langle c,r\rangle$ is a candidate condition role.
3. Returning condition roles that meet the condition evaluation during the session.

Thus, how to evaluate the conditions based on subject attributes and system attributes seems to be an important step in enabling condition roles. Here we define an auxiliary function *check_condition(c,role,sys)*, which returns *True* iff the condition logical expression is a tautology when each variable in condition express *c* is substituted with values of corresponding attributes. We say that a subject *s* with the static role *role* can activate a condition role cr_i in a system if and only if the following conditions are satisfied:

1. *role*=cr_i[*r*];
2. *check_condition(c,role,sys)=True*

The complete algorithm for enabling condition roles is shown in Fig 2.

After condition roles are enabled, access purpose authorization can be simply implemented by authorizing access purpose permissions to certain conditions roles based on the *Access Purpose Permission Assignment(APPA)*. *APPA* is predefined relations between *Condition Roles* and *Access Purpose Permissions* in the policies determined by database administrators. Therefore, after a subject enables a condition role *cr*, he gets the access purpose permissions which are assigned to *cr*.

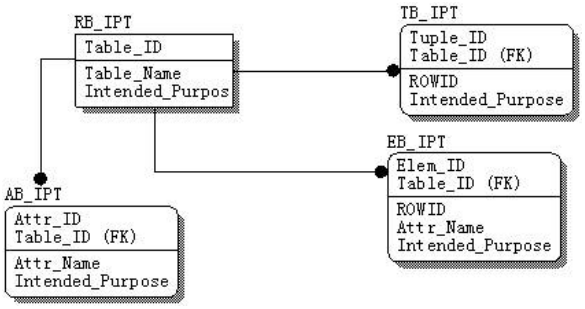

Figure 3. RB_IPT, AB_IPT, TB_IPT, EB_IPT

5. Access Decision Based on Purpose Compliance

5.1. Privacy metadata organization

Intended purposes can be associated with data objects in different granularities. For instance, in a relational model, the metadata can be on: relation level, tuple level, attribute level or element level. Through the establishment of the relationship between user data and privacy metadata, that is, data objects and intended purposes, the data providers are able to determine whether or not to release data in tables to requesters according to their own privacy requirements and preferences.

[5] proposes an approach that intended purposes for elements and tuples could be added into tables as privacy labels directly, while intended purposes for attributes and tables are stated by using schema tables. In this paper, we propose four uniform privacy metadata tables shown in Fig 3 to manage intended purposes on the four granularities.

Data providers are able to control the release of their own data by adding privacy policies into the metadata tables on the corresponding granularity, which will not affect user data in the database. The design provides the flexibility of the intended purpose binding, but will cause the conflict between the policies on different granularities. Therefore, several rules are established which are listed below:

Rule 1. *To solve the conflict of intended purposes defined in different metadata tables, we need to make out the pre-order of intended purposes in RB_IPT, AB_IPT,TB_IPT and EB_IPT. We define the relationship from the ordinary to the special of intended purposes is: RB_IPT$\preceq$TB_IPT$\preceq$AB_IPT$\preceq$EB_IPT. For instance, the bound data of tuple $\{obj_1,\langle AIP_1,PIP_1\rangle\}$ in RB_IPT and tuple $\{obj_2,\langle AIP_2,PIP_2\rangle\}$ in EB_IPT have an intersection, then the following rules should be fulfill:*

1. $AIP_2^* \subseteq AIP_1^*$
2. $PIP_1^* \subseteq PIP_2^*$

Rule 2. *To prevent repeated intended purpose definitions of the same data object, it is forbidden to have two or more tuples which are bound to the same data object in the same privacy metadata table.*

Rule 3. *According to the principle of fail-safe defaults, any data object which has no intended purposes defined in the metadata tables, the default intended purposes are $\langle$Ø, General Purpose$\rangle$.*

Based on the three rules, privacy protection access decision can be made by scanning the four tables and doing some basic data computation.

5.2. Privacy preserving access decision

According to the requirement of support for rich privacy related metadata [3], some additional information such as the retention of the data and legal actions should be contained in the privacy metadata table. We can extend the privacy metadata tables to implement it; however, the rules in section 5.1 must be changed.

When data providers submit data, intended purposes associated with data may be defined at different granularities. The access control engine needs a match process to finish the compliance computation between access purposes and intended purposes. If the requester's access purpose is compliant with the intended purposes of requested data, the engine will release data to the requester; otherwise, the returned data will be null.

According to Rule 1 of Section 5.1, if the access purpose can't be compliant with intended purposes in coarse-grained privacy metadata tables, it wouldn't be compliant with intended purposes in fine-grained privacy metadata tables. Therefore, the process of privacy preserving access decision could be started from searching privacy metadata in RB_IPT, refining the searching granularity step by step. At the same time, the engine replaces some data elements with null according to the privacy metadata. The algorithm is listed in Fig 4.

6. Conclusion

Purposes play an important role in the field of DBMS privacy preserving technique. In this article, we propose an access control model for privacy protection based on RBAC. In the model DPBAC, we discuss the separation of access purpose authorization from access purpose decision to strengthen privacy preserving. Besides, dynamic is the key characteristic of DPBAC. The notion of condition roles is used to determine the access purpose in a dynamic manner. Intended purposes are dynamically associated with the requested data objects. We also discuss the algorithm to achieve the compliance computation between the access

Privacy Preserving Access Decision Algorithm

1). ***Comp_Check (ap, ⟨AIP, PIP⟩)***
2). If ap$\in PIP^*$ then
3). Return False;
4). Else if ap$\notin AIP^*$ then
5). Return False;
6). End if

1). ***Access Decision (AP, Object R)***
2). For each tuple of RB_IPT where Table_ID=i (i=1 to n)
3). If R=$\prod_{Table\_Name}(\sigma_{Table\_ID=i}$(RB_IPT))
4). ip=$\prod_{Intended\_Purpose}(\sigma_{Table\_ID=i}$(RB_IPT))
5). If (*Comp_Check(AP, ip)*=False)
6). Return NULL;
7). For each tuple of AB_IPT where Attr_ID=j (j=1 to n)
8). attr=$\prod_{Attr\_Name}(\sigma_{Attr\_ID=j}$(AB_IPT))
9). If R=$\prod_{Table\_Name}(\sigma_{Attr\_ID=j}$(AB_IPT$\bowtie$RB_IPT)) and attr$\in${A|A is one of R's attributes.}
10). ip=$\prod_{Intended\_Purpose}(\sigma_{Attr\_ID=j}$(AB_IPT))
11). If (*Comp_Check(AP, ip)*=False)
12). R$\leftarrow\prod_{attr_1,...attr=null,...attr_n}$(R)
13). For each tuple of TB_IPT where Tuple_ID=k (k=1 to n)
14). rid=$\prod_{ROWID}(\sigma_{Tuple\_ID=k}$(TB_IPT))
15). If R=$\prod_{Table\_Name}(\sigma_{Tuple\_ID=k}$(TB_IPT$\bowtie$RB_IPT)) and rid$\in\prod_{R.ROWID}$(R)
16). ip=$\prod_{Intended\_Purpose}(\sigma_{Tuple\_ID=k}$(TB_IPT))
17). If (*Comp_Check(AP, ip)*=False)
18). R$\leftarrow\prod_{attr_1=null,...attr_i=null,...attr_n=null}(\sigma_{R.ROWID=rid}$(R))
19). For each tuple of EB_IPT where Elem_ID=l (l=1 to n)
20). rid=$\prod_{ROWID}(\sigma_{Elem\_ID=l}$(EB_IPT))
21). attr=$\prod_{Attr\_Name}(\sigma_{Elem\_ID=l}$(EB_IPT))
22). If R=$\prod_{Table\_Name}(\sigma_{Elem\_ID=l}$(EB_IPT$\bowtie$RB_IPT)) and attr$\in${A|A is one of R's attributes.} and rid$\in\prod_{R.ROWID}$(R)
23). ip=$\prod_{Intended\_Purpose}(\sigma_{Elem\_ID=l}$(EB_IPT))
24). If (*Comp_Check(AP, ip)*=False)
25). R$\leftarrow\prod_{attr_1,...attr=null,...attr_n}(\sigma_{R.ROWID=rid}$(R))
26). Return R

Figure 4. Access Decision

purpose and intended purposes. We are now verifying the model and doing some improvement on the algorithm. In the future work, the following problems are worth to research.

1. Try to describe requesters' obligations and integrate them into the implement of DPBAC.
2. Construct access policies indexes to improve the metadata algorithm.
3. Invest the implement of DPBAC for complex data models such as XML and object-oriented model.

7 ACKNOWLEDGMENT

This work is supported by NSFC 60673140 and national 863 program 2007AA01Z156.

References

[1] R. Agrawal, P. Bird, T. Grandison, J. Kiernan, S. Logan, and W. Rjaibi. Extending relational database systems to automatically enforce privacy policies. In *ICDE*, pages 1013–1022, 2005.
[2] R. Agrawal, J. Kiernan, R. Srikant, and Y. Xu. Hippocratic databases. In *VLDB*, pages 143–154, 2002.
[3] E. Bertino, J.-W. Byun, and N. Li. Privacy-preserving database systems. In *FOSAD*, pages 178–206, 2005.
[4] J.-W. Byun and N. Li. Purpose based access control for privacy protection in relational database systems. *VLDB J.*, 17(4):603–619, 2008.
[5] S. Fischer-Hubner. *IT-security and privacy: design and use of privacy-enhancing security mechanisms.* Springer-Verlag New York, Inc, New York, NY, USA, 2001.
[6] O. for Economic Co-operation and Development. *OECD guidelines on the protection of privacy and transborder flows of personal data of 1980.*
[7] Q. Ni, A. Trombetta, E. Bertino, and J. Lobo. Privacy-aware role based access control. In *SACMAT*, pages 41–50, 2007.
[8] OASIS. *Privacy policy profile of xacml v2.0.* Available at http://www.oasis-open.org/.
[9] M. Olivier. Database privacy: Balancing confidentiality, integrity and availability. *IGKDD Explorations Newsletter*, 4(2):20–27, 2002.
[10] S. Rizvi, A. O. Mendelzon, S. Sudarshan, and P. Roy. Extending query rewriting techniques for fine-grained access control. In *SIGMOD Conference*, pages 551–562, 2004.
[11] 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.
[12] N. Yang, H. Barringer, and N. Zhang. A purpose-based access control model. In *IAS*, pages 143–148, 2007.