Journal of Information Assurance and Security 1 (2008) 51-58

A Purpose-Based Access Control Model

Naikuo Yang Howard Barringer Ning Zhang
School of Computer Science
University of Manchester
Manchester, M13 9PL
{yangn, hbarringer, nzhang}@cs.man.ac.uk

Abstract:
Achieving privacy preservation in a data-sharing computing environment is becoming a challenging problem. Some organisations may have published privacy policies, which promise privacy protection practices on data collection, use and disclosure, but these practices may not be implemented. To maintain consistency between the privacy policy and the practices, privacy protection requirements in privacy policy should be formally specified. In specifying privacy policy, we use purpose as the basis of access control. In this paper, we extend our previous work to specify purpose management. Purpose can be divided into two categories: intended purpose and access purpose. Privacy policy is to ensure that data can only be used for its intended purpose, and the access purpose should be compliant with the data's intended purpose. We specify entities in the purpose-based access control model. Using the technique of VDM, we then specify the invariants corresponding to the privacy requirements in privacy policy, and then specify the operations in the model and investigate their proof obligations.

I. Introduction

With the rapid development of information technology, personal information can be collected, stored and used in various information systems, and therefore achieving privacy preservation in these data-sharing environments is becoming a major concern. Privacy is one of the major issues to be handled in many environments, such as the domain of health care [3, 5, 19], e-Learning [11, 16] and e-Commerce [1, 2, 17].
For protecting privacy, many organisations have published guidelines. The OECD guidelines [21] for data protection and FTC Fair Information Practice Principles (FIP) [9] provided general privacy requirements that organisations should comply with. In parallel to these definitions of general privacy principles, some organisations may have also published privacy policies, which promise privacy protection practices on data collection, use and disclosure, but these practices may not be implemented. The problem is amplified if personal data is used not only within the organisations that collected the data, but also by other external organisations, such as partner organisations or managing authorities with a legitimate need to access the data.
To maintain consistency between the privacy policy and the practices, privacy protection requirements in privacy policy should be formally specified.
The Platform for Privacy Preferences (P3P) [18] is a standardisation approach for formally specifying privacy policy of service provider. It depicts privacy policy into rules in a standard machine-readable format, and compares these rules with corresponding rules in consumer's user agent, which express privacy preferences of the consumer. But this approach is restrictively used for privacy protection on the internet, and it just gives the description of promises rather than technical measures for the enforcement of policies. Some other existing approaches for specifying privacy policies are also incomplete, either there is no entity of purpose, or there is no formal description of authorisation and validation mechanisms.
Since privacy policies are concerned with the purposes that data object is used for rather than the actions that users perform on the data object, traditional access control models cannot easily achieve privacy protection, and the notion of purpose should play a major role in access control model to protect privacy. In this paper, the concept of purpose is used as the basis of access control policy. Purpose can be divided into two categories: intended purpose and access purpose. Privacy policy is to ensure that data can only be used for its intended purpose, and the access purpose should be compliant with the data's intended purpose.
We proposed a privacy preserving access control model based on the notion of purpose [22]. We specify entities in the purpose-based access control model. This provides a framework for all required elements of a privacy policy. Using the technique of VDM [14, 6], we then specify the invariants corresponding to the privacy requirements in privacy policy, and then specify the operations in the model and investigate their proof obligations. This ensures that the privacy policy has a clear and unambiguous interpretation. Since many advanced data management systems, such as the ones based on the object data model, need to manage complex objects or objects with hierarchical structures, we need a more comprehensive purpose management model.

Received January 14, 2008

This paper is organised as follows. After related work is discussed in Section 2, we provide in Section 3 the entities of purpose-based access control model. Section 4 specifies privacy invariants corresponding to the privacy requirements in a privacy policy. Then in section 5 the operations in the model are specified and their proof obligations are also investigated. Finally, a summary of the paper is given in section 6.

II. Related Work

We will talk about previous work in the area of privacy preservation in two categories: extending the basic access control model for privacy preservation or specifying privacy requirements.

The Hierarchical Privacy-Sensitive Filtering (HPSF) [20] model was proposed by Oberholzer to protect personal privacy of a data owner in relational databases, through defining privacy-sensitive levels for data items. In the model, a data owner can specify privacy-sensitive levels (PSL) for his/her data items, and each user is also assigned with a user privacy-sensitive level (UPSL). The PSL value of a data item indicates how sensitive the data owner is about disclosure of the data item. The UPSL value for a specific user or role indicates the level of access that the user or role will be allowed with regard to the data item. Data access is only allowed if the PSL value of the data item is no higher than the UPSL value for user or role. The privacy-sensitive level in this approach is similar to the concept of security levels in the multi-level security (MLS) model [4] which mainly aims at preserving the confidentiality of the data. However, because the privacy-sensitive levels of data usage are different from the confidentiality levels, and they may vary from individual to individual, it is difficult to define persistent PSLs and UPSLs. Furthermore, the HPSF model has only been implemented and illustrated in a simple hospital scenario.

A task-based access control model was proposed by Fischer-Hübner et al [10]. The work extended task-based access control model with the notion of purpose and consent. Data can only be accessed in a controlled manner by executing a task. A task is specified by a set of operations which represent "necessary accesses" for performing that task. The data are related to consent of the data owner and certain purposes about the data usage when they are collected. A user can access the data if this access is necessary to perform its current task and the user is authorised to perform the task (requirement of necessity of data processing). The purpose of current task should be checked against the purpose for which the personal data were obtained or with the consent of the data owner (requirement of purpose binding). This approach specified privacy invariants, privacy constraints and information flow rules. The task is looked on as state transition, and those invariants, constraints, and information flow rules will be checked during the state transitions. The major contribution of this approach is that it has illustrated two important requirements - *necessity of data processing and purpose binding* - for privacy preservation and demonstrated how a privacy policy may be enforced. However, because it was based on the tasks that the user is performing, the access control lacks of scalability.

Privacy-Aware Role-Based Access Control (PARBAC) model [12] was proposed for enforcing privacy policies within an organisation. Privacy is considered together with security protection and combined with data management technologies. The model combines RBAC, Domain-Type Enforcement, and privacy protection, and it provides support to privacy enforcement by combining access control and privacy management. When a user requests to access certain data, not only the roles and permissions of the user are considered, the business purpose of the operation and the privacy policy that pertains to that user are also checked. PARBAC goes beyond traditional access control models in that it not only provides system security from an organisation's perspective, but also protects privacy from a data owner's standpoint. PARBAC enables organisations to act as a trusted custodian to protect customer data privacy, and using RBAC as the base of access control provides a favorable degree of scalability. However, this approach cannot guarantee privacy compliance because it is built upon putting the trust in organisations that collected data. Besides, this approach introduced the purpose related to data objects, but it did not give a systematic data object model together with purposes.

A privacy preserving access control approach [7, 8] was proposed by Byun et al based on the notion of purpose. Purposes in this approach are divided into two categories: intended purpose and access purpose. An intended purpose is related to a data object, and specifies the intended usage of the data object. An access purpose, on the other hand, is related to data accesses, and specifies the intentions for which a given data object are accessed. Each user is required to state his/her access purpose along with the data request. The system validates the stated access purpose to make sure that the user is indeed allowed for the access purpose. In addition, only when an access purpose is compliant to its intended purpose is the access allowed. This work focuses on the notion of purpose, which is a key element in privacy policy. It divides purposes into intended purpose and access purpose corresponding to the data object and the data access, which makes access control clearer. Based on this work, we will give formal description of the entities and the relationships among these entities, and then we will use it to specify the requirements for privacy preservation.

By extending Flexible Authorisation Framework (FAF)[13] with grantors and obligations, Karjoth et al proposed a formal model for authorisation management and access control in privacy protecting systems[15]. They created a privacy control language using the logical framework of the Authorisation Specification Language ASL with the extension of the notion of grantors and obligations. They then formalise the privacy policy with this language. The specified privacy

policy can serve as the basis for internal access control system. This work gives the privacy model entities as well as the specifications of privacy requirements in privacy policy, which provide a complete specification of privacy preserving access control system. However, this work does not give formal specification of entities.

III. Model Entities

Usually, three entities are used in a basic access control system: subjects, objects, and operations. For the system to be able to perform privacy-preserving access control, entities that can be used by data owners to state their privacy requirements and by the system to enforce these requirements have been included, e.g. the privacy sensitive levels of data objects, the consent of data owner, and the purposes of data usages and data accesses.

Since privacy policies are concerned with the purposes that data object is used for rather than the actions that users perform on the data object, traditional access control models cannot easily achieve privacy protection, and the notion of purpose should play a major role in access control model to protect privacy. Therefore, the concept of purpose is used as the basis of access control policy in our model.

In this section, the entities of our model and system state are defined. We will first define a suitable structure for representing data objects, then we will present the RBAC model with the extension of conditional role, we will then define the entity of purpose, and then we will define the entities for accessing data objects, finally we will specify the system state based on the definitions of entities.

A. An Object Data Model

In this section, an object data model, which gives a suitable structure for representing data objects, is defined.

In each organisation, there are a set of data objects, and typically they are organised using object type information. An object type corresponds to a set of data objects that satisfy some common properties. For example, as to the data collected from the patients in a medical care environment, some data belong to the type of registration data, some data belong to the type of admission data, and some data belong to the type of treatment history, etc. Data objects are classified into object types, because it is much easier to define and administer intended usages and necessary accesses for object types instead of defining them for each single data object.

Let *Object* denote the set of objects, and let *Type* denote the set of object types. Next, we define object type attributes and attribute values to specify the properties of object type.

[Object Type Attributes] Object type attributes are defined as a set of attributes associated with object type, and these attributes describe the properties about the collection of and access to this type of objects.

Let *TypeAttr* denote the set of type attributes. Figure **1** gives an example of some object type attributes in a data model.

[Attribute Values] Attribute values are defined as a set of possible values for object type attributes.

Let *AttrValue* denote the set of all possible attribute values. Figure **1** also gives an example of some possible attribute values associated with object type attributes.

Type Attributes
purpose
retention
service opt_in
service opt_out
...
Attribute Values
admin, diagnosing ...
1 day, 1 week, ...
true, false
true, false
...

Figure 1 An Example of Object Type Attributes and Attribute Values

The object data model is concerned with how the data objects are organised and how they are associated with type attributes.

Next we define the object data model in our system.

[Object Data Model] $ObjectDataModel :: object : 2^{Object}$

$$type : 2^{Type}$$
$$typeAttr : 2^{TypeAttr}$$
$$attrValue : 2^{AttrValue}$$
$$TypeOf : Object \rightarrow Type$$
$$AttrOf : Type \rightarrow 2^{TypeAttr}$$
$$ValueOf : Object \times TypeAttr \rightarrow AttrValue$$

inv mk-$ObjectDataModel$(*o, t, ta, av, To, Ao, Vo*) $\triangleq$
(dom $To = o \wedge$ rng $To \subseteq t$) $\wedge$
(dom $Ao = t \wedge$ rng $Ao \subseteq 2^{ta}$) $\wedge$
(dom $Vo = o \times t \wedge$ rng $Vo \subseteq av$)
where

1. *object* is a set of objects
2. *type* is a set of object types
3. *typeAttr* is a set of type attributes
4. *attrValue* is a set of attributes values
5. *TypeOf*: *Object*→*Type* is a total function giving the type associated with each object
6. *AttrOf*: *Type*→ $2^{TypeAttr}$ is a total function giving the type attributes associated with each type
7. *ValueOf*: *Object*× *TypeAttr*→ *AttrValue* is a total function giving the value of the attributes associated with objects.

The object data model *OM* in our system, of type $ObjectDataModel$, can then be represented as a tuple $\langle$*object, type, typeAttr, attrValue, TypeOf, AttrOf, ValueOf*$\rangle$.
In this section, we defined the object data model for representing data objects in our system. Next we will define the structure for representing the subjects in our system.

B. Users and Roles

The purpose-based access control approach of [7, 8] extends the RBAC model with the concept of conditional role, which is based on role attributes and system attributes. In this section, the formal definitions of role attributes, system attributes, and conditional role are presented.
First, we specify the entities of user and role in the basic RBAC model. Users are the active entities in a system, e.g. the staff in a medical care scenario. Let *User* denote the set of users.
The roles in a system reflect the responsibilities of positions or job descriptions in the context of an organisation, e.g. therapist, registration staff, or billing staff in a medical care scenario. Let *Role* denote the set of roles. A user may be assigned several roles and a role may be assigned to several users.
UserRole: $User \leftrightarrow Role$ is the relation between users and roles.
A user may be assigned many roles, but the user may not exercise all his roles at the same time. The roles that a user currently exercising are "active" roles.
Active Roles AR: *User*$\rightarrow 2^{Role}$ is a function that returns the roles for which a user is active.
Because the existing role definitions are predefined for the access permission assignments, they may not adequately specify the set of users to whom we wish to grant an access purpose. The concept of *conditional role* was then introduced. It is based on the notion of *Role Attributes* and *System Attributes*. Next we specify them accordingly.
[Role Attributes] Role attributes are defined as the set of role properties related to the grant of access purpose.
Let $RoleAttr$ denote the set of role attributes. Every role $r \in Role$ is associated with a set of role attributes, e.g. the specialty of therapists in a medical care scenario.
RoleAttrOf: $Role \rightarrow 2^{RoleAttr}$ is a function that returns the set of role attributes of a role.
Role attributes of a role r are denoted as $RoleAttrOf(r)$ = $\{r.attr_1, \ldots, r.attr_i, \ldots, r.attr_n\}$. Each attribute $r.attr_i$ is associated with some values. Let *RoleAttrValue* denote the set of all possible role attribute values.
RoleAttrValueOf: $Role \times RoleAttr \rightarrow RoleAttrValue$ is a function giving the value of the role attributes associated with a role.
For the access control system, we define system attributes to describe the properties of system context.
[System Attributes] System attributes are defined as the set of properties related to the context of the access control system.
For the access control system, it has a set of system attributes, e.g. the staff working hours within a hospital. The set of system attributes is denoted as $SysAttr$ = $\{sysattr_1, \ldots, sysattr_i, \ldots, sysattr_n\}$.
The values of system attributes specify the conditions of the access control system. Let $SysAttrValue$ denote the set of all possible system attribute values.
SysAttrValueOf: $SysAttr \rightarrow SysAttrValue$ is a function giving the value of the system attributes in a system.
With role attributes and system attributes, we can then define conditional role.
[Conditional Role] Conditional role is defined as a role with conditions specifying properties of it.

$$CondRole :: r : Role$$
$$cond : RoleAttrValue \times SysAttrValue \rightarrow B$$

where B is the boolean set, and $cond$: $RoleAttrValue \times SysAttrValue \rightarrow B$ is a truth-valued function.
We use $CR : 2^{CondRole}$ to hold the set of conditional roles in a system.
Current Conditional Role CCR: *User* $\rightarrow$ *CR* is a function that returns the conditional role that the user currently exercises.
Then, we can define the role model in our system.
[Role Model]

$$\begin{aligned} RoleModel :: role &: 2^{Role} \\ user &: 2^{User} \\ UserRole &: User \leftrightarrow Role \\ AR &: User \rightarrow 2^{Role} \\ roleAttr &: 2^{RoleAttr} \\ roleAttrValue &: 2^{RoleAttrValue} \\ RoleAttrValueOf&: Role \times RoleAttr \rightarrow \\ &\quad RoleAttrValue \\ sysAttr &: 2^{SysAttr} \\ sysAttrValue &: 2^{SysAttrValue} \\ SysAttrValueOf &: sysAttr \rightarrow sysAttrValue \\ CR &: 2^{CondRole} \\ CCR &: user \rightarrow CR \end{aligned}$$

The role model *RM* can be represented as a tuple $\langle$*role, user, UserRole, AR, roleAttr, roleAttrValue, RoleAttrValueOf, sysAttr, sysAttrValue, SysAttrValueOf, CR, CCR*$\rangle$.

C. Purpose and Purpose Tree

So far, the object data model for data objects and the role model for subjects in our system have been introduced. In this section, the entity of purpose is formally defined.
[Purpose] Purpose is defined as the intention of data collection or data access.
Data is collected for certain purposes, e.g. in a medical care scenario, the data is collected for registration, diagnosing, or billing. Moreover, each data access serves a certain purpose. It is necessary to determine purposes for which data is collected and purposes of data accesses. Let $Purpose$ denote the set of purposes in a system.
[Purpose Tree] Purposes are organised in a tree structure, which is called purpose tree.
Let P_T denote the purpose tree.

$$P_T = Tree \cup \{nil\}$$

$Tree :: pl : P_T$
$p : Purpose$
$pr : P_T$

$Node : P_T \rightarrow Purpose$ is a function returns the root node of a given purpose tree.

$Node(p) = p$
$Node(\langle pl, p, pr\rangle) = p$

$Edges : P_T \rightarrow 2^{Purpose \times Purpose}$ is a function returns the edges of a given purpose tree.

$Edges(p) = \{\}$
$Edges(\langle pl, p, pr\rangle) =$
$\{\langle p, Node(pl)\rangle, \langle p, Node(pr)\rangle\} \cup Edges(pl) \cup Edges(pr)$

In respect of purposes, some are general, and some are special. There are some relationships among them. The purposes are organised into purpose tree according to these relationships. Next we define relationships among purposes.

[Specialisation (Generalisation)] If p_1, p_2 are two nodes in a purpose tree, and $\langle p_1, p_2\rangle$ is an edge, then we say p_2 is a specialisation of p_1 (or p_1 is a generalisation of p_2).

$Specialisation$: $Purpose \times Purpose \rightarrow B$ is a truth-valued function that characterises the specialisation relation. $Generalisation$ relation can be defined accordingly. Both $Specialisation$ and $Generalisation$ are the transitive closure of the relation $Edges(P_T)$.

We have specified the purposes and the relationships among purposes. Next, we specify the purposes according to data collections and data accesses.

[Intended Purpose] Intended Purpose is defined as the specified usages for which the data objects are collected.

Intended purpose specifies the property of data objects.

IP: *object(OM)* $\cup$ *type*$(OM) \rightarrow 2^{Purpose}$ is a function that returns intended purposes of data object or object type. Here, *object* and *type* are defined in object data model *OM*, *Purpose* is the set of purposes.

[Access Purpose] Access Purpose is defined as the intentions for accessing data objects.

Access purpose specifies the property of data accesses.

Authorised Access Purpose AAP: $CR(RM) \rightarrow 2^{Purpose}$ is a function that returns authorised access purposes of a specific conditional role.

Next, we can define the purpose model in our system.

[Purpose Model]

$PurposeModel :: purpose : 2^{Purpose}$
$Specialisation : Purpose \times Purpose \rightarrow B$
$Generalisation : Purpose \times Purpose \rightarrow B$
$IP : object(OM) \cup type(OM) \rightarrow 2^{Purpose}$
$AAP : CR(RM) \rightarrow 2^{purpose}$

The purpose model *PM* in our system, of type $PurposeModel$, can then be represented as a tuple $\langle$*purpose, Specialisation, Generalisation, IP, AAP*$\rangle$.

D. Requests, Transactions, and Accesses

In this section, we will specify the entities for accessing data objects. We will specify requests, transactions and accesses.

[Request] $Request :: obj : object(OM)$
$ap : purpose(PM)$

When a conditional role *cr* wants to access an object obj, it makes a request for the data object with a particular access purpose *ap*. The request is denoted as a 2-tuple $\langle obj, ap\rangle$. For example, the request from a GP to treatment history for the purpose of diagnosing has the form of $\langle treatment\ history, diagnosing\rangle$. We use $Req : 2^{Request}$ to hold the set of requests in a system.

Current Request CReq: $CR(RM) \rightarrow Req$ is a function that returns current request which is the request currently presenting by a conditional role.

[Transactions] Transactions are defined as the procedures to be executed to achieve a request. In order to achieve an access purpose, it is not allowed to access the object arbitrarily. For a request, it may execute certain transactions that access objects in controlled manners. For example, as to the request of diagnosing, it consists of reading the treatment history, analysing the medical test results, and appending new diagnosis to the treatment history. Let *Transaction* denote the set of transactions in a system.

Current Transaction CT: $CR(RM) \rightarrow Transaction$ is a function that returns current transaction which is the transaction currently being performed by a conditional role.

Authorised Transactions AT: *Req* $\rightarrow 2^{Transaction}$ is a function returns the authorised transactions for performing a request.

Next we define entities about accesses in our system.

[Access Modes] Access modes define the modes of accesses performing on the data objects. Let *Mode* denote the set of access modes. $Mode = \{create, read, write, append, delete\}$

[Necessary Accesses] Necessary accesses are defined as the accesses that are needed to achieve an access purpose, performing transactions on object types in certain access modes.

For any access purpose, it has to be defined in advance what accesses are needed to achieve that particular access purpose. So we define necessary accesses for the access purpose.

$NecAcc :: ap : Purpose$
$tp : type(OM)$
$trans : Transaction$
$x : Mode$

We use NA: 2^{NecAcc} to hold the set of necessary accesses, which consists of tuples of the form $\langle ap_i, tp_j, trans_k, x\rangle$, where $ap_i \in Purpose$, $tp_j \in Type$, $trans_k \in Transction$, and $x \in Mode$. $\langle ap_i, tp_j, trans_k, x\rangle \in NA$ means that for a conditional role to achieve an access purpose ap_i, it is necessary to access data objects of the type tp_j in mode x through the transaction $trans_k$.

[Current Accesses] Current accesses are the accesses a conditional role is performing.

$CurAcc :: cr : CR(RM)$
$obj : object(OM)$
$x : Mode$

We use CA: 2^{CurAcc} to hold the set of current accesses,

which consists of tuples of the form $\langle cr, obj, x\rangle$, where $cr \in CR(RM)$, $obj \in object(OM)$, and $x \in Mode$.
Then, we can define the access model in our system.
[Access Model]

$$\begin{aligned} AccessModel :: \; & Req : 2^{Request} \\ & CReq : CR(RM) \to Req \\ & Trans : 2^{Transation} \\ & \quad CT : CR(RM) \to Trans \\ & \quad AT : Req \to 2^{Trans} \\ & Mode \\ & \quad NA : 2^{NecAcc} \\ & \quad CA : 2^{CurAcc} \end{aligned}$$

The access model *AM* in our system, of type $AccessModel$, can then be represented as a tuple $\langle$*Req, CReq, Trans, CT, AT, Mode, NA, CA*$\rangle$.
So far, the entities in purpose-based access model have been introduced. Then we can define the system state. The specification technique of VDM (the Vienna Development Method) [6, 14] is used for the specification of system state.

E. The State of System

The formalisation of the model consists of the specification of system state. System state consists of the state variables corresponding to the components defined in previous sections: OM, RM, PM, AM.
The state space, without invariant and initialisation condition as yet, is written as follows:

state PPS of
$OM : ObjectDataModel$
$RM : RoleModel$
$PM : PurposeModel$
$AM : AccessModel$
inv ...
init ...
end

The initialisation condition on the state is defined as:
The initialisation condition on the state is defined as:
init $\sigma \triangleq \{\sigma.object(OM) = \{\} \wedge \sigma.type(OM) = \{\} \wedge$
$\sigma.purpose(PM) = \{\} \wedge \sigma.AAP(PM) = \{\mapsto\} \wedge$
$\sigma.Req(AM) = \{\} \wedge \sigma.Trans(AM) = \{\} \wedge$
$\sigma.CA(AM) = \{\} \wedge \sigma.NA(AM) = \{\}\}$
In this section, the entities in purpose-based access control model and the system state have been introduced. Then we are able to specify privacy requirements in privacy policy. We will specify the state invariants corresponding to the requirements in next section, and we will also specify the operations in the model.

IV. Privacy Invariants in Purpose-Based Access Control Model

In this section, a way to specify privacy requirements using the Purpose-Based Access Control Model is described. We specify a privacy policy as an example. We will express privacy requirements in the privacy policy.

The following privacy policy was stated in [10]:

> *A subject may only have access to personal data if this access is necessary to perform its current task, and only if the subject is authorised to perform this task. The subject may only access data in a controlled manner by performing a transformation procedure, for which the subject's current task is authorised. In addition, the purpose of its current task must correspond to the purposes for which the personal data was obtained or consent must be given by the data subjects.*

There are two important aspects of data access that should be protected by a privacy-preserving access control system according to this policy: necessity of data accesses and purpose binding of accesses to data.
Using the entities we defined before, and according to the process of data access, we express privacy requirements in the privacy policy stated above in following invariants (we place the symbol "$'$" behind a state variable to refer to the variable in the new system state):
We define invariants through the process of data access. First, we define the invariants for the creation of data objects.
(a) Data Collection Invariants
(a1) A data object can be created only if it is necessary for a conditional role's current request.
Given two successive system states v, v',
$v = (OM, RM, PM, AM)$,
$v' = (OM', RM', PM', AM')$,
(v, v') satisfies privacy invariant-(a1), iff
$\forall\, cr \in CR(RM),\ type_j \in type(OM), ap \in purpose(PM)$:
$obj \notin object(OM) \wedge \langle obj, ap\rangle = CReq(AM)(cr)$
$\wedge \langle ap, type_j, CT(AM)(cr), create\rangle \notin NA(AM)$
$\Longrightarrow obj \notin object(OM') \vee TypeOf(OM')(obj) \neq type_j$
This invariant specifies the necessity of data object creation.
(a2) A data object may be created if and only if the purpose of a conditional role's current request match the purpose of the object's type.
Given two successive system states v, v',
$v = (OM, RM, PM, AM)$,
$v' = (OM', RM', PM', AM')$,
(v, v') satisfies privacy invariant-(a2), iff
$obj \notin object(OM) \wedge \langle obj, ap\rangle = CReq(AM)(cr) \wedge$
$ap \notin IP(PM)(type_j) \Longrightarrow obj \notin object(OM') \vee$
$TypeOf(OM')(obj) \neq type_j$
This invariant specifies the purpose compliance of data object creation.
(b) Role Authorisation Invariants
These invariants specify the authorisation of conditional role, access purpose and transaction.
(b1) A user's current conditional role has to be authorised.
For a system state $v = (OM, RM, PM, AM)$,
v satisfies privacy invariant-(b1), iff
$\forall\, u \in user(RM),\ \langle r, cond\rangle \in CR(RM)$:

$\langle r, cond\rangle = CCR(RM)(u) \Longrightarrow r \in AR(RM)(u) \wedge$
$cond \Leftrightarrow$ true
(b2) A conditional role's access purpose in its current request has to be authorised for the conditional role.
For a system state $v = (OM, RM, PM, AM)$,
v satisfies privacy invariant-(b2), iff
$\langle obj, ap\rangle = CReq(AM)(cr) \Longrightarrow ap \in AAP(PM)(cr)$
(b3) A conditional role's current transaction has to be authorised for the conditional role's current request.
For a system state $v = (OM, RM, PM, AM)$,
v satisfies privacy invariant-(b3), iff
$trans = CT(AM)(cr) \Longrightarrow$
$trans \in AT(AM)(CReq(AM)(cr))$
These invariants specify the authorisation of conditional role, access purpose and transaction.

(c) Data Access Constraints

(c1) A conditional role may only have current access to a data object if the access of executing the transaction on the object type is the necessary access for the access purpose.
For a system state $v = (OM, RM, PM, AM)$,
v satisfies privacy invariant-(c1), iff
$\langle obj, ap\rangle = CReq(AM)(cr) \wedge \langle cr, obj, x\rangle \in CA(AM) \Rightarrow$
$\langle ap, Typeof(OM)(obj), CT(AM)(cr), x\rangle \in NA(AM)$
This invariant specifies the necessity of data access.
(c2) A conditional role may only have current access to a data object, if the purpose of its current request is compliant to the intended purposes of the type of the object.
For a system state $v = (OM, RM, PM, AM)$,
v satisfies privacy invariant-(c2), iff
$\langle obj, ap\rangle = CReq(AM)(cr) \wedge \langle cr, obj, x\rangle \in CA(AM) \Rightarrow$
$ap \in IP(PM)(Typeof(OM)(obj))$
This invariant specifies the purpose compliance of data access.
(c3) A conditional role may delete a data object, if and only if it is necessary for its current request.
Given two successive system states v, v',
$v = (OM, RM, PM, AM)$,
$v' = (OM', RM', PM', AM')$,
(v, v') satisfies privacy invariant-(c3), iff
$obj \in object(OM) \wedge \langle obj, ap\rangle = CReq(AM)(cr) \wedge$
$\langle ap, Typeof(OM)(obj), CT(AM)(cr), delete\rangle \notin$
$NA(AM) \Longrightarrow obj \in Object(OM')$
This specifies the necessity of data object deletion.
(c4) A conditional role may delete a data object, if and only if the purpose of its current request is compliant to the intended purpose of the type of the object.
Given two successive system states v, v',
$v = (OM, RM, PM, AM)$,
$v' = (OM', RM', PM', AM')$,
(v, v') satisfies privacy invariant-(c4), iff
$obj \in object(OM) \wedge \langle obj, ap\rangle = CReq(AM)(cr) \wedge ap \notin$
$IP(PM)(Typeof(OM)(obj)) \Longrightarrow obj \in object(OM')$
This specifies purpose compliance of object deletion.
The invariant of the system state of PPS is the conjunction of these expressions, denoted as $inv\text{-}PPS$.

The invariants have been specified in this section. Next we specify the model rules, and give the proof obligations of model rules.

V. Model Rules

In this section, formal specifications of model rules will be given. They specify operations by which the state variables can be changed. The precondition and the postcondition are used to specify the operations. In addition, the proof obligations of operations [6] show that the operations are satisfiable.

Rule: create-object

Conditional role cr requests to create an object obj with the type tp. This is specified as following:
$create\text{-}object(cr : CR(RM), obj, tp : type(OM))$
ext rd $RM : RoleModel$ rd $PM : PurposeModel$
rd $AM : AccessModel$ wr $OM : ObjectDataModel$
pre $obj \notin object(OM) \wedge \langle obj, ap\rangle = CReq(AM)(cr) \wedge$
$ap \in IP(PM)(tp) \wedge$
$\langle ap, tp, CT(AM)(cr), create\rangle \in NA(AM)$
post $OM' = \langle object(OM) \cup \{obj\}, type(OM),$
$typeAttribute(OM), attributeValue(OM),$
$TypeOf(OM) \cup \{obj \mapsto tp\}, AttributeOf(OM),$
$ValueOf(OM)\rangle$
The precondition says that obj is not already in the set of objects, and to create object obj in current request is necessary access. The postcondition says that obj in included in the new object data model.
Next, defined symbols representing the operation's precondition and postcondition are introduced.
$pre\text{-}create\text{-}object(cr, obj, tp, OM, RM, PM, AM) \stackrel{\text{def}}{=}$
$obj \notin object(OM) \wedge \langle obj, ap\rangle = CReq(AM)(cr) \wedge ap \in$
$IP(PM)(tp) \wedge \langle ap, tp, CT(AM)(cr), create\rangle \in NA(AM)$
$post\text{-}create\text{-}object(obj, tp, OM, OM') \stackrel{\text{def}}{=}$
$OM' = \langle object(OM) \cup \{obj\}, type(OM),$
$typeAttribute(OM), attributeValue(OM),$
$TypeOf(OM) \cup \{obj \mapsto tp\}, AttributeOf(OM),$
$ValueOf(OM)\rangle$
There is the satisfiability obligation associated with this operation, and this proof obligation states that there must always be at least one state configuration satisfying the operation's postcondition whenever the system is in some legal state and when the operation's parameters satisfy its precondition in that state.

Proof Obligation: Operation $create\text{-}object$ is satisfiable.
$create\text{-}object\text{-}sat$
$OM : ObjectDataModel;\ RM : RoleModel;$
$PM : PurposeModel;\ AM : AccessModel;\ inv\text{-}PPS;$
$pre\text{-}create\text{-}object(cr, obj, tp, OM, RM, PM, AM)$

$\exists\, obj, type : type(OM), OM : ObjectDataModel,$
$OM' : ObjectDataType \cdot$
$post\text{-}create\text{-}object(obj, tp, OM, OM') \wedge inv\text{-}PPS'$
Next we give proof for this satisfiability obligation.

from $OM : ObjectDataModel; RM : RoleModel;$
$PM : PurposeModel; AM : AccessModel; inv\text{-}PPS;$
$pre\text{-}create\text{-}object(cr, obj, tp, OM, RM, PM, AM)$
1 $\{obj\} : 2^{object(OM')}$
2 $object(OM) \cup \{obj\} : 2^{object(OM')}$
3 $\{obj \mapsto tp\}$: $object(OM^{'}) \rightarrow type(OM^{'})$
4 $TypeOf(OM) \cup \{obj \mapsto tp\}$: $object(OM^{'}) \rightarrow type(OM^{'})$
5 from $inv\text{-}PPS$
5.1 $ap \in IP(PM)(tp)$
5.2 $\langle ap, tp, CT(AM)(cr), create\rangle \in NA(AM)$
infer $object(OM^{'}) = object(OM) \cup \{obj\} \wedge$
$TypeOf(OM^{'}) = TypeOf(OM) \cup \{obj \mapsto tp\}$
6 $\exists\, obj, tp : type(OM), OM : ObjectDataModel,$
$OM^{'} : ObjectDataModel \cdot inv\text{-}PPS^{'}$
infer $\exists obj, type : type(OM),$
$OM : ObjectDataModel, OM^{'} : ObjectDataType \cdot$
$post\text{-}create\text{-}object(obj, tp, OM, OM^{'}) \wedge inv\text{-}PPS^{'}$

VI. Conclusions

Privacy preservation requirements should be formally specified and enforced in order to maintain consistency between the privacy preservation promises and the practices. In this paper, the entities of the purpose-based access control model were formally specified. It extend our previous work to specify purpose management. The invariants corresponding to the privacy requirements in privacy policy were also given, the operations in the model were presented, the specifications of operations and their proof obligations have also been investigated.

References

[1] M. Ackerman, L. F. Cranor, and J. Reagle. Privacy in E-Commerce: Examining user scenarios and privacy preferences. In *Proceedings of the ACM Conference on Electronic Commerce (EC-99)*, pages 1–8, N.Y., Nov. 3–5 1999. ACM Press.

[2] Acquisti. Privacy in electronic commerce and the economics of immediate gratification. In *CECOMM: ACM Conference on Electronic Commerce*, 2004.

[3] R. J. Anderson. Privacy technology lessons from healthcare. In *IEEE Symposium on Security and Privacy*, pages 78–79, 2000.

[4] D. E. Bell and L. LaPadula. Secure computer systems: Unified exposition and multics interpretation. *MITRE technical report, MITRE Corporation, Bedford Massachusetts*, 2997:ref A023 588, 1976.

[5] J. Bhattacharya, S. K. Gupta, and B. Agrawal. Protecting privacy of health information through privacy broker. In *HICSS*. IEEE Computer Society, 2006.

[6] I. C. Bicarregui, I. S. Fitzgerald, P. A. Lindsay, R. Moore, and B. Ritchie. *Proof in VDM: A Practitioner's Guide*. Springer, London, 1994.

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

[8] J.-W. Byun, E. Bertino, and N. Li. Purpose based access control of complex data for privacy protection. In *SACMAT '05: Proceedings of the tenth ACM symposium on Access control models and technologies*, pages 102–110, New York, NY, USA, 2005. ACM Press.

[9] Federal Trade Commission (FTC). Privacy online: A report to congress. Technical report.

[10] S. Fischer-Hübner and A. Ott. From a formal privacy model to its implementation. In *Proc. 21st NIST-NCSC National Information Systems Security Conference*, pages 512–524, 1998.

[11] E. Franz, H. Wahrig, A. Boettcher, and K. Borcea-Pfitzmann. Access control in a privacy-aware elearning environment. In *ARES*, pages 879–886. IEEE Computer Society, 2006.

[12] Q. He. Privacy enforcement with an extended role-based access control model. Technical Report TR-2003-09, Department of Computer Science, North Carolina State University, Feb. 28 2003.

[13] S. Jajodia, P. Samarati, M. L. Sapino, and V. S. Subrahmanian. Flexible support for multiple access control policies. *ACM Trans. Database Syst*, 26(2):214–260, 2001.

[14] C. B. Jones. *Systematic Software Development using VDM (second edition)*. Prentice Hall, 1990.

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

[16] T. Klobucar. Privacy and data protection in technology-enhanced professional learning. In *AICT/ICIW*, page 14. IEEE Computer Society, 2006.

[17] L. Korba. Privacy in distributed electronic commerce. In *HICSS*, page 306, 2002.

[18] M. Marchiori. The platform for privacy preferences 1.0 (P3P1.0) specification. W3C recommendation, W3C, Apr. 2002. http://www.w3.org/TR/2002/REC-P3P-20020416/.

[19] I. Mavridis, G. Pangalos, M. Khair, and L. Bozios. Defining access control mechanisms for privacy protection in distributed medical databases, Jan. 02 1999.

[20] H. Oberholzer. A privacy protection model to support personal privacy in relational databases. Technical report, Rand afrikanns university, May 2001.

[21] Organisation of Economic Co-operation and Development (OECD). OECD guidelines on the protection of privacy and transborder flows of personal data, Oct. 26 2004.

[22] N. Yang, H. Barringer, and N. Zhang. A purpose-based access control model. In *Third International Symposium on Information Assurance and Security, 2007*, Aug. 2007.