ATTRIBUTE MUTABILITY IN USAGE CONTROL

Jaehong Park, Xinwen Zhang, and Ravi Sandhu
George Mason University
{jpark2, xzhang6, sandhu}@gmu.edu

Abstract The notion of Usage Control (UCON) has been introduced recently to extend traditional access controls by including three decision factors called *authorizations*, *obligations*, and *conditions*. Usage control also recognize two important decision properties of *continuity* and *mutability*.

In access control literature, an authorization decision is commonly made by utilizing some form of subject and object attributes. Identities, security labels and roles are some examples of attributes. Traditionally these attributes are assigned to subjects and objects by a security officer and can be modified only by administrative actions. However, in modern information systems these attributes are often required to be changed as a side effect of subject's usage on object. This requirement of updates has been recognized and defined as mutability property in usage control. In this paper, we discuss issues of this attribute mutability and show how usage control can apply this mutability property in various traditional and modern access control policies.

1. Introduction

The notion of usage control has been introduced recently in our previous papers [Park and Sandhu, 2002, Sandhu and Park, 2003, Park and Sandhu, 2004] . A Usage control (UCON) model called $UCON_{ABC}$ has been defined to extend traditional access control so it can cover modern access control systems. Although the UCON study has been inspired largely from digital rights management (DRM) whose main interest lies in commercial segment, usage control is a general purpose, unified framework that covers traditional access control, trust management and digital rights management and goes beyond them in its scope.

Over the last thirty years, majority of access control literature has dealt with authorizations mainly by utilizing some forms of subject attributes and object attributes such as security labels or roles. Traditionally, these attributes assigned to subjects and objects are relatively static and can be modified only by administrative actions. Although this approach might be adequate for some

traditional access control policies, this is no longer appropriate for access controls in modern information systems. Often, today's information systems utilize more dynamic and complex decision policies that require certain modifications on subject and object attributes as side effects of usages on digital resources. This has been identified as attribute mutability in usage control.

In this paper we discuss attribute mutability issues in UCON's perspective. In section 2, we first describe the general idea of usage control and summarize UCON components and a family of models based on the components. Then in section 3 we explore attribute management and mutability issues and identify taxonomy of attribute mutability. In section 4 we identify two types of mutable attributes called temporary and persistent attributes. We discuss several variations that require attribute mutability property by using access control policy examples. In section 5 we further discuss related issues on attribute mutability and section 6 gives our conclusion.

2. Usage Control

Access control has been studied for more than 30 years now. Some well-known access control models are access matrix model, lattice-based access control model, and role-based access control model. These traditional access control models have difficulty in addressing the needs of modern information systems. One of the main reasons is that traditional access control models have focused on authorization only. Here, authorization evaluates access requests based on subject attributes, object attributes and requested rights. However, modern information systems often require more than authorizations. For example, one may have to fill out a certain form or click 'yes' button for license agreement for usage allowance. We call these required actions as *obligations*. Obligations are requirements that have to be fulfilled by obligation subjects for usage allowance. Moreover, some digital objects can be played only on a certain device or location. These environmental restrictions are called conditions. Conditions are environmental and system-wide requirements that have to be satisfied for access. Obligations and conditions are rarely discussed in traditional access control models. In today's highly dynamic, distributed environment, obligations and conditions are also crucial *decision factors* for richer and finer controls on usage of digital resources.

Also in traditional access control models, authorization decision is made before access is allowed and there is no further enforcement during the access. Hence there is no ongoing control concept considered. Another shortcoming is that consumable rights are not supported. In modern e-commerce system, it is common to use consumable attributes or rights such as credit balance or limited number of usages. More fundamentally, in traditional access control, rights are pre-defined and granted to subjects. This means that subjects hold granted

rights for indefinite time whether the subjects actually exercise the rights or not. This might be fine for some authorization-based controls. However, this is not acceptable in obligation-based or condition-based controls as well as in other dynamic authorization controls.

Although these shortcomings are not new and have been recognized in recent literature, these recent studies are problem-specific and only deal with certain aspects of the issues. Usage control is a general purpose, unified framework that resolves all these aspects in a systematic way. In our previous paper, we have identified a family of $UCON_{ABC}$ models for next generation access control by integrating obligations, conditions as well as authorizations, and by including continuity and mutability properties. Here, continuity property recognizes ongoing controls for relatively long-lived access or for immediate revocation and mutability deals with updates on related subject or object attributes as a consequence of access. We call $UCON_{ABC}$ as a core model since it captures only the essence of usage control while there are other important issues uncovered such as administrative issues and delegation issues.

2.1 A Family of Usage Control Models

$UCON_{ABC}$ models consist of eight components. As shown in Figure 1, they are Subjects (S), Objects (O), Rights (R), Subject Attributes (ATT(S)), Object Attributes (ATT(O)), and three decision factors called Authorizations ($\mathcal{A}$), oBligations ($\mathcal{B}$), and Conditions ($\mathcal{C}$). In access control, S, O, R are well known concepts. In UCON, subjects are regarded as representing users for simplicity. A subject holds rights on objects and is associated with subject attributes. Objects are target resources that subjects hold rights on, and associated with object attributes either by themselves or together with rights. These attributes are used for usage decision making.

Authorization ($\mathcal{A}$) is a functional predicate that evaluates usage requests based on subject attributes, object attributes and requested rights, and returns either yes or no. This is a typical view of traditional access control. For example, in Mandatory Access Control (MAC), a subject's clearance is regarded as a subject attribute and an object's classification as an object's attributes and authorization is made based on simple or star security properties by utilizing these attributes. In addition to authorization, there are two other decision factors called obligations ($\mathcal{B}$) and conditions ($\mathcal{C}$). Obligation is a functional predicate that verifies if required obligation actions have been fulfilled or not. Condition is a functional predicate that check environmental or system status. Figure 1 shows these components of $UCON_{ABC}$.

In addition to these three decision factors, ABC model also includes two crucial properties called continuity and mutability. With continuity property, decision can be made either before (pre) or during ($ongoing$) a usage. Muta-

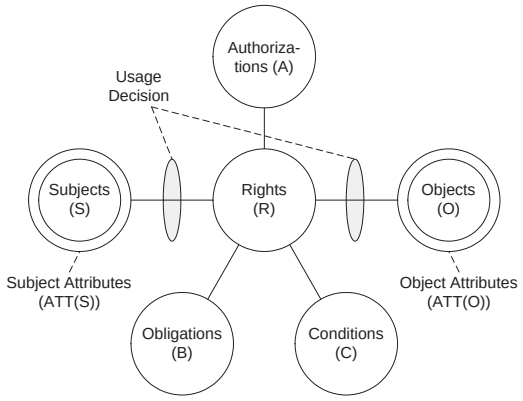

Figure 1. $UCON_{ABC}$ Model Components

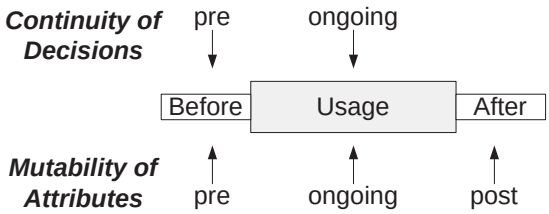

Figure 2. Continuity and Mutability Properties

bility means mutability of attributes. With mutability property, attributes can be either immutable or mutable. Immutable attributes can be modified by administrative actions while mutable attributes can be modified as side-effects of subject actions. In $UCON_{ABC}$, attribute updates can be made either before ($preUpdate$), during ($onUpdate$) or after ($postUpdate$) usages. These continuity and mutability properties are shown in Figure 2.

Based on these three decision factors and two properties, $UCON_{ABC}$ model contains various sub-models. The details of this family of models have been discussed in [Park and Sandhu, 2004]. Definition 1 shows the definition for authorization model with pre-decision making. Please note that ongoing decision model utilizes $stopped(s, o, r)$. In Definition 1, $allowed(s, o, r)$ means a subject s is allowed to exercise a right r on an object o. The $pre\mathcal{A}$ is a pre-authorization predicate that has to be satisfied for usage allowance. Here, $preUpdate$ and $postUpdate$ are optional procedures to perform update operations on $ATT(s)$ or $ATT(o)$. $onUpdate$ is not shown here because it can be

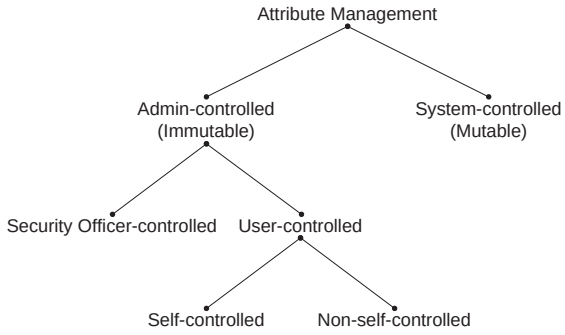

Figure 3. Attribute Management Taxonomy

utilized only with ongoing decisions.

Definition 1 The $UCON_{preA}$ model has the following components:

- $S, O, R, ATT(S), ATT(O)$ and $pre\mathcal{A}$;
- $allowed(s, o, r) \Rightarrow pre\mathcal{A}(ATT(s), ATT(o), r)$;
- $preUpdate(ATT(s)), preUpdate(ATT(o))$, $postUpdate(ATT(s)), postUpdate(ATT(o))$.

3. Attribute Management and Mutability

Usage control model includes several underlying presumptions. In usage control, usage decision is request-based. This means rights are not pre-assigned to subjects and usage decision is made at the time of usage request. Also, in usage control, authorization decision is made based on subject attributes and object attributes. Depending on access control policies, these attributes may have to be updated. Naturally management of these attributes is a key concern in usage control. Attribute management can be either *'admin-controlled'* or *'system-controlled'*. This section discusses these two categories. Figure 3 shows taxonomy of the attribute management.

3.1 Admin-controlled Attribute Management (Immutable)

Administrator-controlled attributes can be modified by administrative actions. These attributes are modified by administrator discretion but are "immutable" in that the system does not modify these automatically. Mutable attributes are modified by the system automatically. Here the administrator

can be either a *security officer* or a *user*. In general, administrative actions are made by security officers. Suppose a subject is assigned to a new security label or to a new membership group because of management decision. Here, updates on attributes are made by administrative actions. This is a typical approach in traditional access control policies such as MAC and RBAC. Static separation of duty and user-role assignment in RBAC belongs in this category. However, there are other cases where subject attributes are controlled by a user. This *user-controlled* attribute management can be further classified into *self-controlled* and *non-self-controlled*. An example of self-controlled attribute management is a role activation. In RBAC, a user can activate or deactivate his or her roles in a session. Here, the notion of a session corresponds to the notion of a subject. Controlling users' ability to update attributes (e.x., activated roles) is also considered as an administrative issue. In non-self controlled cases, a user other than the user of subjects or sessions controls attributes. For example, in online music store, parents of a child may preset the child's maximum purchase limits as 20 dollars a month. This is done by controlling attributes of the child. In UCON, all of these cases are considered as part of the administrative model and are not included in this paper.

3.2 System-controlled Attribute Management (Mutable)

Unlike admin-controlled, in system-controlled attribute management, updates are made as side effects of user's usage on objects. For instance, a subject's credit balance has to be decreased by the value of the usage on an object at the time of the usage. This is different from the update by an administrative action because the update in this case is done by the system while in admin-controlled management the update involves administrative decisions and actions. Because of this, we call a system-controlled attribute as a mutable attribute. Mutable attributes do not require any administrative actions for updates. Therefore attribute mutability is considered as part of core models. In both admin-controlled and system-controlled management, it is the security officer who manages the ability of user updates and system updates. In this paper our concern lies in the system-controlled mutability issue where updates are made as side effects of users' actions on objects since it is a main concern of core model for usage control.

4. Attribute Mutability in UCON

In $UCON_{ABC}$ model, attribute update can occur on both authorizations and obligations models. As shown in Definition 1, attribute updates are realized by adding update procedures within the model definition. In case of condition model, because it evaluates subject and object independent environmental or system-wide requirements, there is no update process required on attributes.

In usage control, attributes can be either mutable or immutable. In case of mutable models, there exist two kinds of attributes based on liveness of attributes. They are *temporary attributes* and *persistent attributes*. In this section we identify these two types of attributes. We further discuss several mutability variations of traditional and modern access control policies that require attribute mutability and show how these policy examples can be realized in $UCON_{ABC}$ models.

4.1 Mutable Attributes

Within mutable models, temporary attributes are alive only for a single usage while persistent attributes live longer for multiple usage decisions. Temporary attributes are created at the time a usage is started and deleted at the end of the usage. Suppose a system allows only 100 internet connection at the same time and allowed internet connections are terminated based on longest idle time to keep maximum 100 connections. The system has to keep the last active time of each connection and the number of current connections for continuous usage decision. This example is shown in Example 1. In this example, a subject's last active time ($lastActiveT$) exists only during the connection to be used for the decision of continuous connection. This is an example of temporary attributes. However, the system keeps the number of connections ($usageNum$) for a relatively long period for multiple usages decisions. This is considered as a persistent attribute. Persistent attribute is stateful and used to keep certain property of subjects or objects for multiple usages while temporary attribute is stateless and used for a single usage. Utilizing temporary attributes are largely determined as a design decision and can be eliminated in some cases. For example, Example 1 can be realized without using temporary attributes. Here, if we utilize an object attribute that consists of a set of subject ID and a last active time of each subject's usage on the object, we can eliminate subject's temporary attributes.

Example 1 Simultaneous connection, revocation using last activity time:
T is an ordered set of last activity times
UN is a set of concurrent usage numbers
N is a set of identification names
$id : S \rightarrow N$, $lastActiveT : S \rightarrow T$
$usageNum : O \rightarrow UN$, $activeId : O \rightarrow 2^N$
$ATT(s) : \{id, lastActiveT\}$, $ATT(o) : \{usageNum, activeId\}$

$allowed(s, o, r) \Rightarrow true$
$stopped(s, o, r) \Leftarrow (usageNum(o) > 100) \wedge (lastActiveT(s)$
$= \min\{t' | \exists s', t' = lastActiveT(s'), id(s') \in activeId(o)\})$

$preUpdate(usageNum(o)) : usageNum(o) = usageNum(o) + 1$
$onUpdate(lastActiveT(s))$, repeated updates on $lastActiveT(s)$
$onUpdate(activeId(o))$, repeated updates on $activeId(o)$
$postUpdate(usageNum(o)) : usageNum(o) = usageNum(o) - 1.$

Temporary attributes exist only in mutable models while persistent attributes exist in both mutable and immutable models. Within mutable models, temporary attributes can be utilized for two purposes. First, temporary attributes can be used for ongoing decision processes (ongoing-authorization and ongoing-obligation). A subject's last active time in Example 1 is an example of temporary attributes for ongoing authorization. Note that condition model does not utilize subject or object attributes for usage decision. Although condition model may use subject or object attributes this is only to select condition elements that have to be checked for a decision, not for the decision itself. Also note that temporary attribute doesn't have to be updated continuously. A temporary attribute can be set at the time a usage is started and can remain throughout the usage without any modification. Suppose in previous Example 1 if a decision is made based on longest connection time, the system may keep usage starting time throughout the usage so it can calculate usage time of each usage for ongoing authorization. In this case, usage starting time is a temporary attribute of a subject that doesn't require an ongoing update.

Second, temporary attributes can be used so the result of them or the result derived from them can be reflected into persistent attributes. This can be occurred in either pre-decision models or ongoing-decision models. Suppose a long-distance phone call system updates total usage of a month at the end of each phone call. The system may need to utilize start time to calculate the usage time of each phone call. Here, start time is considered as a temporary attribute and used to update total monthly usage. This is an example of pre-authorization that requires temporary attributes for persistent attribute updates. As shown in Example 2, if we use a prepaid phonecard to place a long distance call, the system will monitor if current usage time exceed allowed time throughout the call. This requires two temporary attributes. One is to store allowed time ($allowedT$) that has been calculated based on card balance and unit cost of the call and the other is a usage time ($usageT$) of current call that is continuously increased throughout the call. At the end of each phone call, the card balance ($cardBalance$) has to be decreased by the cost of current call. Here, the card balance is considered as a persistent attribute. This is an ongoing-authorization example that requires temporary attributes for persistent attribute updates. Note that the updates on persistent attributes to reflect the result of temporary attributes can be occurred either during (ongoing-update) or after (post-update) usages.

Example 2 Long-distance call using Pre-paid phonecard
N is a set of natural number, $value : O \rightarrow N$
$cardBal : S \rightarrow N$, $allowedT : S \rightarrow N$, $usageT : S \rightarrow N$
$ATT(s) : \{cardBal, allowedT, usageT\}$, $ATT(o) : \{value\}$

$allowed(s, o, connect) \Rightarrow cardBal(s) \geq value(o)$
$stopped(s, o, connect) \Leftarrow usageT(s) > allowedT(s)$
$preUpdate(allowedT(s)) : allowedT(s) = cardBal(s)/value(o)$
$onUpdate(usageT(s)) : usageT(s) + 1$
$postUpdate(cardBal(s)) : cardBal(s) - (usageT(s) * value(o))$

4.2 Mutability Variations

In usage control, attribute mutability can occur in various situations. We identify these variations based on the purpose of mutability usages in access control policies. Some of these variations are mainly from traditional access control policies while others are unique in usage control. This uniqueness is largely because of UCON's inclusion of obligations and continuity property as well as its attribute-based authorizations and request-based decision process.

In general, attribute mutability is utilized for history-based usage decision. This means that attributes have to be modified to reflect usage history of subject on objects for either current or future usage decision. Some cases are only for current usage decisions and others are only for future usage decisions while there are other variations that can be used for both current and future usage decisions.

In this section we identify five different variations based on the purpose of attribute mutability. They are mutability for *exclusive/inclusive attributes, consumable/creditable attributes, immediate revocation, obligation* and *dynamic confinement*. Each variation is discussed with examples in UCON's perspective. However, we do not aim to identify a complete list of mutability usages. Rather we recognize several variations for well-known policies and relatively new policies that require attribute mutability to show how attribute mutability can be utilized in different policies and how these policies can be viewed in usage control models.

Mutability for Exclusive/Inclusive Attributes. Mutable attributes can be modified by a system to enforce exclusive rights or inclusive rights. *Exclusive attributes* are used to resolve conflict of interests while *inclusive attributes* can be used to resolve consolidated interest. Suppose issuing a purchase order requires three steps of prepare, approval, and issue. Here a purchase order has to be approved by a user other than the preparer. This is an example of exclusive rights. On the other hand, if the purchase order has to be issued by the same user who has prepared the order this will be an example of inclusive

rights. Both require us to store subject's usage on the object for future usage decisions. Since both exclusive and inclusive attributes can be utilized in very similar way, we show additional examples for exclusive attributes only.

The notion of exclusive attributes or rights is a well-defined concept and shown in traditional access control policies. One of well-known access control policies for exclusive attributes is Dynamic Separation of Duty (DSoD). DSoD has been studied extensively in access control literature [Simon and Zurko, 1997, Gligor et al., 1998, Sandhu, 1988, Sandhu, 1990]. Simon and Zurko [Simon and Zurko, 1997] distinguish DSoD into four categories from simple to complex ones. Fundamentally all of these four categories require attribute mutability property to store subjects' activities on objects. Although they use the term 'history-based SoD' for the most complex category, we can view other simpler DSoD categories also as 'history-based' which require attribute updates for certain level of mutual exclusion. Sandhu [Sandhu, 1988] uses several examples to express transaction controls for DSoD by emphasizing history-based decisions.

The following example shows the object-based DSoD of Simon and Zurko. Here conflicting roles can be assigned to a user at the same time, but no user is allowed to access previously accessed objects. A subject is used to represent a user. In this example, a subject may have both a 'purchase clerk' role and 'account clerk' role at the same time. However, the subject is not allowed to issue a check that is prepared by himself. This example requires us to store the history of subjects' usage on objects for future usage decision to resolve conflict of interest.

Example 3 Object-based DSoD
ID is a set of identification number. T is a set of object type name.
$ROLE$ is a partially ordered set of role names.
$uid : S \rightarrow ID$, $sRole : S \rightarrow 2^{ROLE}$, $type : O \rightarrow T$
$prepareId : O \rightarrow ID$, $issueId : O \rightarrow ID$, $R : issue, prepare$
$ATT(s) = \{uid, sRole\}$, $ATT(o) = \{type, prepareId, issueId\}$

$allowed(s, o, prepare) \Rightarrow type(o) = `check', sRole(s) \geq `purchaseClerk'$
$preUpdate(prepareId(o)) : prepareId(o) = uid(s)$
$allowed(s, o, issue) \Rightarrow type(o) = `check', sRole(s) \geq `accountClerk',$
$uid(s) \neq prepareId(o)$
$preUpdate(issueId(o)) : issueId(o) = uid(s)$

Another traditional example is the Chinese Wall policy identified by Brewer and Nash [Brewer and Nash, 1989]. Chinese Wall policy aims to prevent information flows among companies in conflict of interest. Just like DSoD, Chinese Wall policy requires updates on attributes to store subject's usage history on

objects for usage decision process. This is shown in the following Example 4. No temporary attribute is used in example 3 and 4. Sandhu has shown how to express Chinese Wall policies in a lattice-based approach [Sandhu, 1992]. Both approaches can be easily realized in UCON models.

Example 4 Brewer and Nash's Chinese wall policy
$clName$ is a set of conflict of interest class names.
$coName$ is a set of company names.
$cl : O \rightarrow clName$, $co : O \rightarrow coName$
$accessedCl : S \rightarrow 2^{clName}$, $accessedCo : S \rightarrow 2^{coName}$
$ATT(S) = \{accessedCl, accessedCo\}$, $ATT(O) = \{cl, co\}$

$allowed(s, o, read) \Rightarrow (co(o) \in accessedCo(s)) \vee (cl(o) \notin accessedCl(s))$
$preUpdate(accessedCo(s)) : accessedCo(s) = accessedCo(s) \cup co(o)$
$preUpdate(accessedCl(s)) : accessedCl(s) = accessedCl(s) \cup cl(o)$
$allowed(s, o, write) \Rightarrow ((co(o) \in accessedCo(s)) \vee$
$(cl(o) \notin accessedCl(s)) \wedge (accessedCo(s) \cup co(o) = co(o))$
$preUpdate(accessedCo(s)) : accessedCo(s) = accessedCo(s) \cup co(o)$
$preUpdate(accessedCl(s)) : accessedCl(s) = accessedCl(s) \cup cl(o)$

Mutability for Consumable/Creditable Attributes. Mutability for consumable or creditable attributes means that the value of an attribute has to be either decreased (or consumed) or increased (or credited) as side effects of usages for either current or future usage decisions. We can distinguish these two as mutability for *consumable attributes* and mutability for *creditable attributes*.

In modern information systems, it is common to have a consumable rights or attributes for usage control. Digital rights management (DRM) with payment-based authorization is a typical example. With payment-based authorization, each usage allowance requires updates on credit balance. Commonly, a subject's credit balance has to be decreased by the amount of an object's value. Examples are a limited number of usages, a limited period of usage time, etc. In the initial policy of Apple's iTunes digital music service, a user is allowed only 10 times of CD burnings for an identical list of music files. In this case, we can consider the list of music files as an object and the object is associated with an attribute that includes available number of CD burning. In the following example, each burning reduces the number of available burnings by 1.

Example 5 Mutability for consumable attributes, limited CD burnings
N is a set of natural number, $available : O \rightarrow N$, $ATT(o) : \{available\}$
$allowed(s, o, burn) \Rightarrow available(o) \geq 1$
$preUpdate(available(o)) : available(o) = available(o) - 1$

We can think of an opposite case of mutability for creditable attributes where certain attributes are credited as a consequence of usages. This is categorized as mutability for creditable attribute. Modern information systems often require attribute mutability for creditable attributes. Suppose, in a hospital information system, a nurse has to have a minimum five times of operation observations to participate an operation. In the following example, this can be realized by updating a subject attribute called 'exp' that stores 'observation number'.

Example 6 Mutability for creditable attributes, Hospital information system
$ROLE$ is an unordered set of roles
$TYPE$ is a set of object types
N is a set of subject's total operation observation numbers
$exp : S \rightarrow N$, $sRole : S \rightarrow 2^{ROLE}$, $oType : O \rightarrow TYPE$
$ATT(s) : \{sRole, exp\}$, $ATT(o) : \{oType\}$

$allowed(s, o, observe) \Rightarrow$ '$nurse$' $\in sRole(s), oType(o) =$ '$operation$'
$preUpdate(exp(s)) : exp(s) = exp(s) + 1$
$allowed(s, o, participate) \Rightarrow$ '$nurse$' $\in sRole(s)$,
$oType(o) =$ '$operation$', $exp(s) \geq 5$

Mutability for Immediate Revocation. When used together with continuity property, attribute mutability can be utilized to support immediate revocation of usage. With continuity property, usage decision can be made continuously throughout usages. Hence, a system has to keep updating temporary or persistent attributes to resolve current status of usages for immediate revocation of current usage. Example 1 and 2 are examples that requires mutable attributes for immediate revocation.

Mutability for Obligation. Obligation is one of the decision factors in usage control. In obligation-based usage decision, an obligation subject has to fulfill required obligation actions for usage allowance. The result of obligation fulfillment has to be reflected in a form of attribute update and can be used for both current and future usage decisions. In the following example, a subject has to click a license agreement button for usage. The result of this obligation fulfillment is reflected in a subject attribute. This modified attribute is used for usage decision.

Example 7 License agreements for first time users only
$OBS = S, OBO = \{license\_agreement\}, OB = \{agree\}$
$registered : S \rightarrow \{yes, no\}$
$ATT(s) = \{registered\}$
$getPreOBL(s, o, r) =$

$$\begin{cases} (s, license\_agreement, agree), & \text{if } registered(s) = \text{'no'}; \\ \phi, & \text{if } registered(s) = \text{'yes'}. \end{cases}$$

$allowed(s, o, r) \Rightarrow preFulfilled(getPreOBL(s, o, r))$
$preUpdate(registered(s)) : registered(s) = \text{'}yes\text{'}$

Mutability for Dynamic Confinement. In usage control, attribute update can occur for dynamic confinements. In this case, attributes are updated for dynamic controls on usages. High-watermark property in mandatory access control is a traditional example. With high-watermark property, although a subject has top-secret clearance, the subject's clearance is assigned with 'unclassified' label at the beginning. However, as the subject accesses a secret object, his clearance is increased to secret label and he is no longer able to write on lower objects.

Example 8 MAC policies with high watermark property
L is a lattice of security labels with dominance relation $\geq$
$clearance : S \rightarrow L$, $maxClearance : S \rightarrow L$, $classification : O \rightarrow L$
$ATT(S) = \{clearance, maxClearance\}$, $ATT(O) = \{classification\}$

$allowed(s, o, read) \Rightarrow maxClearance(s) \geq classification(o)$
$preUpdate(clearance(s)) : clearance(s) =$
$LUB(clearance(s), classification(o))$

5. Discussion

Mutability variations that we have identified are not meant to be mutually exclusive. Rather real world examples are likely to include attribute mutability for multiple purposes. Suppose a subject has to click advertisement windows at least once in every 20 minutes for continuous Internet services. In this case, the system has to keep updating last click time of the subject throughout the connection. This example requires attribute mutability for both obligations and immediate revocations. Another example can be borrowed from [Sandhu, 1988] to show how one example can exhibit multiple mutability variations. Suppose a check issuing process includes three order-dependent steps of 'prepare check', 'approve check' and 'issue check'. If each step has to be performed by a different subject, this requires attribute mutability for exclusive attributes. Also if 3 approvals on a check are required to issue the check or if each step has to be performed in an ordered manner for the authorization of the following steps, these require attribute mutability for creditable attributes.

For access control policies or rules that require attribute mutability, updates can be made on either subject attributes or object attributes. For example, consider Example 3, object-based DSoD. Here, once a subject has prepared

a check, subject ID is stored as an object attribute called $prepareId$. Although this *object attribute update* might be more intuitive, same result can be achieved by implementing a *subject attribute update*. In this case, as shown in Example 9, if we create a subject attribute called $preparedObjId$ to store prepared object IDs, usage decision for issuing a check can be made by checking whether the currently requested object is found in the subject attribute $preparedObjId$. Although one way may be preferred to the other for different policy examples, this is likely to be a design decision rather than a concrete preference rules between subject attribute update and object attribute update.

Example 9 Object-based DSoD with subject attribute update
ID is a set of identification number. T is a set of object type name.
$ROLE$ is a partially ordered set of role names. $R = \{issue, prepare\}$
$oid : O \rightarrow ID$, $sRole : S \rightarrow 2^{ROLE}$, $type : O \rightarrow T$
$preparedObjId : S \rightarrow 2^{ID}$, $issuedObjId : S \rightarrow 2^{ID}$
$ATT(s) = \{sRole, preparedObjId, issuedObjId\}$, $ATT(o) = \{type, oid\}$

$allowed(s, o, prepare) \Rightarrow type(o) =$ '$check$', $sRole(s) \geq$ '$purchaseClerk$'
$preUpdate(preparedObjId(s)) : preparedObjId(s) =$
$preparedObjId(s) \cup oid(o)$
$allowed(s, o, issue) \Rightarrow type(o) =$ '$check$', $sRole(s) \geq$ '$accountClerk$',
$oid(o) \notin preparedObjId(s)$
$preUpdate(issuedObjId(s)) : issuedObjId(s) = issuedObjId(s) \cup oid(o)$

6. Conclusion

In this paper we have defined a taxonomy for attribute management to show how attributes can be controlled in usage control and how attribute mutability has to be viewed in the context of attribute management. We have further discussed mutable attributes and identified temporary and persistent attributes. We have also discussed several attribute mutability variations based on the purposes of mutability. Several examples has been discussed in UCON's point of view to show how mutable attributes are utilized in traditional and modern access control policies.

Attribute mutability is not new and has appeared in several traditional access control policies such as Dynamic Separation of Duty, Chinese Wall policy, or mandatory access control with high watermark property. Each of these policies has been studied extensively but separately. In usage control these policies are captured in a single framework together with other access control policies for modern information systems such as digital rights management or hospital information systems. When used together with continuity property and obligations, attribute mutability property can be utilized for various purposes as

shown in this paper. This paper is an initial step on this line of work and only covers mutable attribute issues. Further research on both mutable and immutable attributes is required. We believe the study has been done in this paper provides a foundation for future research on attribute management and usage control.

References

[Brewer and Nash, 1989] Brewer, D. and Nash, M. (1989). The Chinese Wall security policy. In *Proceedings of IEEE Symposium on Security and Privacy*, pages 215–228.

[Gligor et al., 1998] Gligor, V., Gavrila, S., and Ferraiolo, D. (1998). On the formal definition of separation-of-duty policies and their composition. In *Proceedings of IEEE Symposium on Security and Privacy*, pages 172 – 183.

[Park and Sandhu, 2002] Park, J. and Sandhu, R. (2002). Towards usage control models: beyond traditional access control. In *Proceedings of the seventh ACM symposium on Access control models and technologies*, pages 57–64. ACM Press.

[Park and Sandhu, 2004] Park, J. and Sandhu, R. (2004). The $UCON_{ABC}$ usage control model. *ACM Transactions on Information and Systems Security*, 7(1):128–174.

[Sandhu, 1988] Sandhu, R. (1988). Transaction control expressions for separation of duties. In *Proc. of the Fourth Computer Security Applications Conference*, pages 282–286.

[Sandhu, 1990] Sandhu, R. (1990). Separation of duties in computerized information systems. In *IFIP Workshop on Database Security*, pages 179–190.

[Sandhu, 1992] Sandhu, R. (1992). Lattice-based enforcement of chinese walls. *Computer and Security*, pages 753–763.

[Sandhu and Park, 2003] Sandhu, R. and Park, J. (2003). Usage control: A vision for next generation access control. In *Proceedings of The 2nd International Workshop on Mathematical Methods, Models and Architectures for Computer Networks Security*, pages 17–31.

[Simon and Zurko, 1997] Simon, R. T. and Zurko, M. E. (1997). Separation of duty in role-based environments. In *IEEE Computer Security Foundations Workshop*, pages 183–194.

Biography

Jaehong Park is a post-doc in Lab for Information Security Technology (LIST), George Mason University. His research interests include access control, digital rights management, and trusted computing.

Xinwen Zhang is a doctoral student in Lab for Information Security Technology (LIST), George Mason University. His research interests include access control models and technologies, and distributed systems security.

Ravi Sandhu is a Professor in Dept. of Information and Software Engineering, George Mason University, the director of Lab for Information Security Technology (LIST), and the Chief Scientist of NSD Security. His research areas include access control models, database security, network security, and distributed system security.