Usage Control: A Vision for Next Generation Access Control

Ravi Sandhu[1] and Jaehong Park[2]

[1] NSD Security and George Mason University
sandhu@gmu.edu
http://www.list.gmu.edu
[2] ISE Department, George Mason University, Fairfax, VA. 22030
jaehpark@ise.gmu.edu

Abstract. The term usage control (UCON) is a generalization of access control to cover obligations, conditions, continuity (ongoing controls) and mutability. Traditionally, access control has dealt only with authorization decisions on a subject's access to target resources. Obligations are requirements that have to be fulfilled by the subject for allowing access. Conditions are subject and object-independent environmental requirements that have to be satisfied for access. 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. Traditional authorization decisions are generally made at the time of request but typically do not recognize ongoing controls for relatively long-lived access or for immediate revocation. Moreover, mutability issues that deal with updates on related subject or object attributes as a consequence of access have not been systematically studied. In this paper we motivate the need for usage control, define a family of ABC models as a core model for usage control and show how it encompasses traditional access control, such as mandatory, discretionary and role-based access control, and more recent requirements such as trust management, and digital rights management. In addition, we also discuss architectures that introduce a new reference monitor for usage control and some variations.

1 Introduction

The classic access matrix model has stood fundamentally unchanged for over three decades. The core concept of the access matrix is that a right (or permission) is explicitly granted to a subject to access an object in a specific mode, such as read or write. This core idea has been successfully elaborated in different directions in the familiar models of discretionary, mandatory and role-based access control, to accommodate a diverse range of real-world access control policies. Turing-completeness of the HRU formalization of the access matrix establishes its wide theoretical expressive power [3].

This success and longevity notwithstanding, many researchers have realized that the access matrix needs fundamental enhancements to meet the needs of modern applications and systems. Coming from multiple perspectives, these researchers have given us new concepts such as trust management, digital rights management (DRM), task-based access control, provisional authorization, obligations and more. The focused perspective has led researchers to propose particular extensions to the access matrix model to deal with shortcomings identified in the application or system context in consideration. The net result is a proliferation of point extensions to different flavors of the access matrix model without a unifying theme.

This paper describes a new approach to access control called usage control as a fundamental enhancement of the access matrix. An earlier formulation of usage control models was given in our previous paper [6]. As the core model of usage control, a family of ABC models is built around three decision factors, authorizations (A), obligations (B) and conditions (C). The familiar notion of authorization is based on subject and object attributes. Obligations require some action by the subject so as to gain or sustain access, e.g., clicking ACCEPT on a license agreement. Conditions are environmental or system-oriented factors that predicate access, e.g., time-of-day or overall system load. Further, with respect to authorizations per se, ABC introduces mutable attributes that change as a consequence of access. Finally, ABC recognizes the continuity of access enforcement so the decision to allow access is not only made prior to access, but also during the time interval that access takes place. Given the Turing completeness of the access matrix model it is theoretically possible to represent these enhancements within the access matrix, but that would ignore their fundamental nature in addressing the shortcomings of the classic access matrix identified by numerous researchers. It is time to enhance the core model.

ABC is the first model to address a systematic and comprehensive extension of the classic access matrix. By integrating authorizations, obligations and conditions along with mutable attributes and ongoing enforcement, ABC quite naturally and elegantly encompasses diverse current proposals in the literature. It is shown that ABC encompasses traditional discretionary, mandatory, and role-based access control. It is further shown that ABC encompasses emerging applications such as trust management, digital rights management, etcetera within a unified framework. Strictly speaking ABC is a family of models because each component has a number of options. For example, ABC without obligations, conditions, mutable attributes and ongoing enforcement, is close to the access matrix. ABC is a core model of usage control in that it focuses on the process of access enforcement while leaving other important issues such as administration or delegation aspects for future development.

In architectural perspective, reference monitor is the most important element of classic access control. Among the main differences with respect to classic access control is the requirement for a client-side reference monitor. This is a hallmark of digital rights management. In this paper, we introduce a modified reference monitor for usage control and discuss variations of reference monitor based on its locations. Section 2 examines new aspects that are not covered by classic access control but crucial for modern applications. Section 3 discuss a family of

ABC models for usage control. We further discuss architectural details for usage control in section 4.

2 Beyond Classical Access Control

One commonality of traditional access controls and even trust management is utilization of subjects' attributes and objects' attributes for authorization process. In other words, in traditional access control, authorization decision is made based on subject attributes, object attributes, and requested rights. Attributes include identities, capabilities, or properties of subjects or objects. For example, in mandatory access control, clearance labels of subjects are considered as subject attributes and objects' classification labels as object attributes. Authorization process, then, evaluates the dominance of these labels along with requested access rights (e.g., read, write) to return either 'allowed' or 'not-allowed'. Similarly, in discretionary access control, access control list (ACL) can be viewed as object attributes and capability list as subject attributes. Figure 1 shows this 'attribute-based' traditional access control.

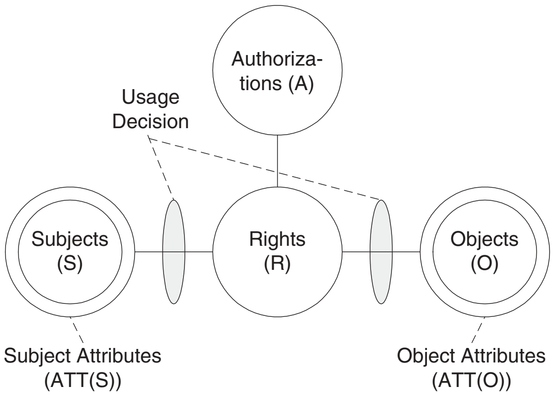

Fig. 1. Traditional Access Control

Although this 'attributed-based' approach of traditional access control can cover many applications, today's digital information systems require more than classical authorizations. For example, suppose Alice has to click 'accept' button for license agreement or has to fill out a certain form to download a company's whitepaper. In this case, certain actions have to be performed by the subject to enable a requested usage. In other words, usage decision is based on fulfillment of required actions, not by existence of subject attributes and object attributes. This decision factor is called as "obligation" and required in addition to authorization to cover modern access control applications.

In addition to authorization and obligation, there are certain situations where access or usage needs to be limited due to certain environmental or system status. For example, usage of certain digital resources may be allowed only during business hours or at certain locations. A system with heavy traffic load may allow only premium users to be serviced. For this requirement, a system needs to check current environmental or system status for usage decision. This decision factor is called as “condition” and required together with authorization and obligation for modern access control.

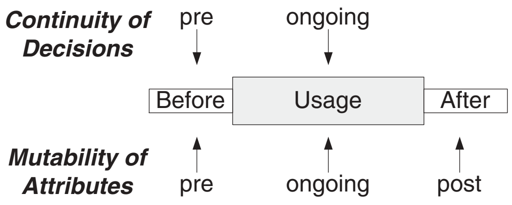

Fig. 2. Continuity and Mutability Properties

In addition to these three decision factors, modern information systems require two other important properties called **“continuity”** and **“mutability”** as shown in Figure 2. In traditional access control, authorization is assumed to be done before access is allowed (pre). However, it is quite reasonable to extend this for continuous enforcement by evaluating usage requirements throughout usages (ongoing). This property is called “continuity” and has to be captured in modern access control for the control of relatively long-lived usage or for immediate revocation of usage.

In traditional access control, attributes are modifiable only by administrative actions. However, in many modern applications such as DRM systems, these attributes have to be updated as side-effects of subjects’ actions. For example, a subject’s e-cash balance has to be decreased by the value of a digital object as the subject uses or accesses the object. This “mutability” property of attributes has been rarely discussed in traditional access control literature. In case attributes are mutable, updates can be done either before (pre), during (ongoing) or after (post) usages as shown in Figure 2. Mutability allows more direct enforcement of various classical policies that require history-based authorizations such as dynamic Separation Of Duty or Chinese Wall policy.

Although some of these issues have been discussed in access control literature, the focus is typically limited to specific target problems, so the discussion is not comprehensive. The notion of usage control (UCON) is developed to cover these diverse issues in a single framework to overcome these shortcomings. In UCON, traditional access control can be extended to include modern access control and digital rights management by integrating obligations and conditions as well as authorizations and by including continuity and mutability properties.

3 ABC Model for Usage Control

The family of ABC model is a core model for usage control. We call this as a core model since it captures the essence of usage control while there are other important issues to be discussed. In this section we briefly discuss eight components of ABC models and a family of models in a systematic manner.

3.1 ABC Model Components

The ABC model consists of **eight components** as follows: subjects, subject attributes, objects, object attributes, rights, authorizations, obligations, and conditions (see figure 3).

Subjects and **objects** are familiar concepts from the past thirty plus years of access control, and are used in their familiar sense in ABC. A **right** enables access of a subject to an object in a particular mode, such as read or write. In this sense the ABC concept of right is essentially similar to the familiar concept of a right in access control. There is a subtle difference in the ABC viewpoint in that ABC does not visualize a right as existing in some access matrix independent of the activity of the subject. Rather the existence of the right is determined when the access is attempted by the subject. The **usage decision functions** indicated in figure 3 make this determination based on subject attributes, object attributes, authorizations, obligations and conditions at the time of usage requests.

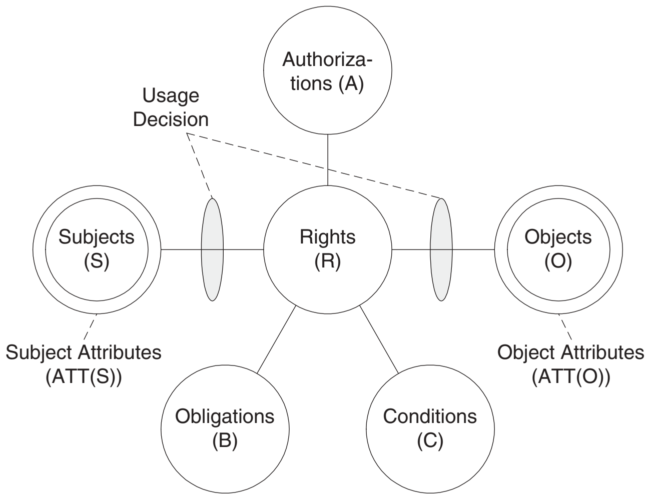

Fig. 3. ABC Model Components

Subject and object attributes are properties that can be used during the access decision process. One of the most important subject attributes in practice is subject identity, but it is not required by the ABC model. Subject identity is often important for determining access, and can be even more important for accountability. However, requiring subject identity as a mandatory attribute precludes anonymous services. Examples of subject attributes include identities, group names, roles, memberships, security clearance, pre-paid credits, account balance, capability lists, etc. Examples of object attributes are security labels, ownerships, classes, access control lists, etc. In e-commerce applications a price-list could be an object attribute, e.g., a particular e-book may stipulate a $10 price for a ‘read’ right and a $15 price for a ‘print’ right. The general concept of attribute-based access control is commonplace in the access control literature and as such this aspect of ABC builds upon familiar concepts.

A significant innovation in ABC is that subject and object attributes can be mutable. **Mutable attributes** are changed as a consequence of access, whereas immutable attributes can be changed only by administrative action. Policies requiring limits on the number of accesses by a subject or reduction of account balance based on access can be easily specified using mutable attributes. More generally, various kinds of consumable authorizations can be modelled in this manner. High watermark policies on subject clearance and Chinese Walls can also be enforced in this way. The introduction of mutable attributes is a critical differentiator of ABC relative to most proposals for enhanced models for access control.

Authorizations, obligations and conditions are decision factors employed by the usage (or access) decision functions to determine whether a subject should be allowed to access an object with a particular right. **Authorizations** are based on subject and object attributes and the specific right in question. Unlike prior models ABC explicitly recognizes that each access has a finite duration. Authorization is usually required prior to the access, but in addition it is possible to require ongoing authorization during the access, e.g., a certificate revocation list (CRL) can be periodically checked while the access is in progress. If the relevant certificate appears on the CRL access can be terminated. Authorizations may require updates on subject and/or object attributes. These updates can be either pre, ongoing, or post. The high watermark policy requires update of the subject’s clearance prior to access. Metered usage payment requires updates after the usage has ended to calculate current usage time. Using pre-paid credits for time-based metering requires periodic updates of the remaining credits while usage is in progress, with possible termination in case of overuse.

Obligations are requirements that a subject must perform before (pre) or during (ongoing) access. An example of a pre-obligation is the requirement that a user must provide some contact and personal information before accessing a company’s white paper. The requirement that a user has to keep certain advertising windows open while he is logged into some service, is an example of an ongoing obligation. Subject and/or object attributes can be used to decide what kind of obligations are required for access approval. The exercise of obligations may update mutable attributes. These updates can affect current or future usage decisions.

Conditions are environmental or system-oriented decision factors. Examples are time of day and system load. They can also include the security status of the system, such as normal, high alert, under attack, etc. Conditions are not under direct control of individual subjects. Evaluation of conditions cannot update any subject or object attributes.

3.2 The ABC Family of Core Models

Based on three decision factors, authorizations, obligations, and conditions, and continuity and mutability properties, we have developed a family of core models for usage control. We say these are core models because, as discussed earlier, they focus on the enforcement process and do not include administrative issues. Also, they will need to be further elaborated for specific applications.

The ABC model assumes there exists a usage request on a target object. Decision-making can be done either before (pre) or during (ongoing) exercise of the requested right. Note that decision-making after the usage does not make sense since there can be no influence on the decision of current usage. Mutability allows certain updates on subject or object attributes as side effects of usages. If usage is immutable, there is no update required for the decision process and denoted as '0'. For mutable usage, updates are required either before (pre), during (ongoing), or after (post) the usage and denoted as '1, 2, and 3', respectively. Based on these criteria, we have developed 16 possible model spaces as a core model for usage control. While there are examples for an individual model, many real world systems are likely to utilize more than one model. In this paper we only consider pure models for simplicity.

	0 (immutable)	1 (pre-update)	2 (ongoing-update)	3 (post-update)
preA	Y	Y	N	Y
onA	Y	Y	Y	Y
preB	Y	Y	N	Y
onB	Y	Y	Y	Y
preC	Y	N	N	N
onC	Y	N	N	N

Fig. 4. The 16 Basic ABC Models

Figure 4 shows all possible detailed models based on these three criteria. Cases that are not likely to be realistic are marked as 'N'. If decision factor is 'pre', updates are likely to occur only before or after the right is exercised and there is little reason to have ongoing updates since without ongoing decision, ongoing-update can influence only decisions on future requests and therefore the updates can be done after the usage is ended. However, if decision factor is 'ongoing', updates are likely to be happen before, during or after the usage. For condition models, evaluation of condition cannot update attributes since it simply checks current environmental or system status.

3.3 ABC Model Definitions and Examples

For simplicity we consider only the "pure" cases consisting of A, B or C alone with pre or ongoing decisions only. In reality these models would often be used in conjunction with each other. It is a valuable thought experiment to consider each model by itself. The goal of this paper is not to develop a logical expression language for the ABC model. Rather, while there can be numerous ways of expressing our ABC model, our focus is to develop comprehensive models for usage control in a unified framework. We believe the ABC model can be used as a reference model for further research in usage control.

$UCON_{preA}$ – Pre-authorizations Models

In $UCON_{preA}$ models, the decision process is performed before access is allowed. We begin with the $UCON_{preA_0}$ which allows no updates of attributes.

Definition 1. The $UCON_{preA_0}$ model has the following components:

- $S, O, R, ATT(S), ATT(O)$ and a usage decision function $pre\mathcal{A}$
- $allowed(s, o, r) \Rightarrow pre\mathcal{A}(ATT(s), ATT(o), r)$.

The $allowed(s, o, r)$ predicate indicates that subject s is allowed to access object o with right r only if the indicated condition is true. This predicate is stated as a necessary condition to allow composition of multiple models, each of which contributes its own conditions. Composition of models requires the conjunction of all relevant $allowed(s, o, r)$ predicates[1].

$UCON_{preA_0}$ corresponds roughly to the classic approach to access control. Examples 1 and 2 show how mandatory and discretionary access controls can be realized within $UCON_{preA_0}$.

Example 1. Mandatory access control stated in $UCON_{preA_0}$:

L is a lattice of security labels with dominance $\geq$
$clearance : S \to L$, $classification : O \to L$
$ATT(S) = \{clearance\}$, $ATT(O) = \{classification\}$
$allowed(s, o, read) \Rightarrow clearance(s) \geq classification(o)$
$allowed(s, o, write) \Rightarrow clearance(s) \leq classification(o)$

Example 2. Discretionary access control using Access Control Lists in $UCON_{preA_0}$:

N is a set of identity names
$id : S \to N$, $ACL : O \to 2^{N \times R}$ (n is authorized to do r to o)
$ATT(S) = \{id\}$, $ATT(O) = \{ACL\}$
$allowed(s, o, r) \Rightarrow (id(s), r) \in ACL(o)$

[1] This is similar to the Bell-LaPadula model [1] where mandatory and discretionary necessary conditions are defined and then applied together.

A capability-based formulation of discretionary access control can be similarly given. For role-based access control, user-role and permission-role assignments can be expressed as subject and object attributes respectively. With mutable attributes we have the following two models[2].

Definition 2. The $UCON_{preA_1}$, and $UCON_{preA_3}$ models are identical to $UCON_{preA_0}$ except they respectively add the following update processes:

- $UCON_{preA_1}$ adds $preUpdate(ATT(s)), preUpdate(ATT(o))$
- $UCON_{preA_3}$ adds $postUpdate(ATT(s)), postUpdate(ATT(o))$

Note that both subject and object attributes can be updated. A Digital Rights Management (DRM) example of *preUpdate* is payment-based access. The *allowed* predicate tests whether the subject s has sufficient $credit(s)$ to access an object o with $price(o)$. The *preUpdate* procedure then decrements $credit(s)$ by the amount $price(o)$. A DRM example of *postUpdate* arises when the price of access depends upon the usage time, i.e., we have metered access. The account balance of the subject needs to be incremented by the rate multiplied by time of use, after access is terminated.

$UCON_{onA}$ – Ongoing-authorizations Models

We begin by formalizing $UCON_{onA_0}$ where no update procedures are included.

Definition 3. The $UCON_{onA_0}$ model has the following components:

- $S, O, R, ATT(S), ATT(O)$ and a usage decision function $on\mathcal{A}$
- $allowed(s, o, r) \Rightarrow true$;
- $stopped(s, o, r) \Leftarrow \neg on\mathcal{A}(ATT(s), ATT(o), r)$.

$UCON_{onA_0}$ introduces the $on\mathcal{A}$ predicate instead of $pre\mathcal{A}$. In absence of pre-authorization, the requested access is always allowed. However, ongoing-authorization is active throughout the usage of the requested right, and the $on\mathcal{A}$ predicate is repeatedly checked for sustaining access. Technically, these checks are performed periodically based on time or event. The ABC model does not specify exactly how this should be done. In case certain attributes are changed and requirements are no longer satisfied, '*stopped*' procedure is performed. We write '$stopped(s, o, r)$' to indicate that right r of subject s to object o is revoked and the ongoing access terminated.

For example, suppose only 10 users can access an object o_1 simultaneously. If a 11th user requests access, the user with the earliest time is terminated. In

[2] It is important to note that the update operations may be nondeterministic. For example, payment for permitting access may be applicable from multiple accounts held by the subject. Which account is debited is not material in enforcement. The exact manner in which the nondeterminism is resolved is not specified as part of the model.

this case, the 11th user is allowed without any pre-authorization decision process. However, $on\mathcal{A}$ monitors the number of current usages on $o_1(ATT(o_1))$, determines which was the earliest to start, and terminates it. Some ongoing authorizations are likely to be occurred only together with pre-authorizations. For example, suppose $on\mathcal{A}$ monitors certain certificate revocation lists periodically to check whether the user's identity certificate is revoked or not. While this is a case of ongoing authorizations, this makes sense only when the certificate has already been evaluated in a 'pre' decision at the time of the request.

$UCON_{onA_1}$, $UCON_{onA_2}$ and $UCON_{onA_3}$ add pre-update $preUpdate(ATT(s))$, $preUpdate(ATT(o))$, ongoing-update $onUpdate(ATT(s))$, $onUpdate(ATT(o))$ and post-update $postUpdate(ATT(s))$, $postUpdate(ATT(o))$ procedures respectively. Suppose the extra user in the above example is revoked based on longest idle time. Monitoring idle time requires ongoing updates of a last activity attribute. This is an example of $UCON_{onA_2}$. Further suppose that revocation of the extra user is based on total usage time in completed sessions since the start of the fiscal year. We would need post-updates to accumulate this time and this is an example of $UCON_{onA_3}$. In both examples, current usage numbers have to be updated at the beginning of each access, hence pre-update (onA_1) is required.

$UCON_{preB}$ – Pre-obligations Models

$UCON_{preB}$ introduces pre-obligations that have to be fulfilled before access is permitted. We model this by the $pre\mathcal{B}$ predicate. Examples of pre-obligations are requiring a user to provide name and email address before accessing a company's white paper, requiring a user to click the ACCEPT box on a license agreement to access a web portal, etc. Note that the pre-obligation action is performed on a different object (e.g., web form, license agreement) than the object that the user is trying to access (e.g., white paper, web portal). More generally, the pre-obligation action may be done by some other subject, e.g., parental consent to access a restricted web site. This complicates formalization of the model given below.

Definition 4. The $UCON_{preB_0}$ model has the following components:

- $S, O, R, ATT(S)$, and $ATT(O)$ as before;
  OBS, OBO, and OB, (obligation subjects, objects, and actions, respectively);
- $preOBL \subseteq OBS \times OBO \times OB$; (pre-obligations elements)
  $preFulfilled : OBS \times OBO \times OB \rightarrow \{true, false\}$;
- $getPreOBL : S \times O \times R \rightarrow 2^{preOBL}$, a function to select pre-obligations for a requested access;
- $pre\mathcal{B}(s,o,r) = \bigwedge_{(obs_i, obo_i, ob_i) \in getPreOBL(s,o,r)} preFulfilled(obs_i, obo_i, ob_i)$
  $pre\mathcal{B}(s,o,r) = true$ by definition if $getPreOBL(s,o,r) = \phi$;
- $allowed(s,o,r) \Rightarrow pre\mathcal{B}(s,o,r)$.

The $getPreOBL$ function represents the pre-obligations required for s to gain r access to o. The $preFulfilled$ predicate tells us if each of the required obligations is true. The $UCON_{preB_1}$ and $UCON_{preB_3}$ add $preUpdate(ATT(s))$, $preUpdate(ATT(o))$ and $postUpdate(ATT(s))$, $postUpdate(ATT(o))$ procedures respectively. The *preUpdate* procedure could be used to mark a subject as registered so the contact information is requested only the first time the subject attempts to access a white paper. The *postUpdate* procedure could be used to monitor total usage of a resource by a subject, e.g., to require periodic reaffirmation of a license agreement.

$UCON_{onB}$ – Ongoing-Obligations Models

The $UCON_{onB}$ models require obligations to be fulfilled while rights are exercised. Ongoing-obligations may have to be fulfilled periodically or continuously. For example, a user may have to click an advertisement at least every 30 minutes or at every 20 Web pages. Alternatively, a user may have to leave an advertisement window active all the time. Note that this concern is about when users have to fulfill obligations, not about when the system actually checks the fulfillments. Actual obligation verification intervals can vary and are not prescribed by the model.

$UCON_{preC}$ – Pre-conditions Model

As described earlier, conditions define environmental and system restrictions on usage. These are not directly related to subjects and objects.

Definition 5. The $UCON_{preC_0}$ model has the following components:

- $S, O, R, ATT(S)$, and $ATT(O)$ are not changed from $UCON_{preA}$;
- $preCON$ (a set of pre-conditions elements);
  $preConChecked : preCON \rightarrow \{true, false\}$;
- $getPreCON : S \times O \times R \rightarrow 2^{preCON}$;
- $pre\mathcal{C}(s,o,r) = \bigwedge_{preCon_i \in getPreCON(s,o,r)} preConChecked(preCon_i)$
- $allowed(s,o,r) \Rightarrow pre\mathcal{C}(s,o,r)$.

Unlike other ABC models, condition models cannot have update procedures. Checking the time-of-day before access is allowed is an example of $UCON_{preC_0}$. Checking the location of the client in cyberspace is another example.

$UCON_{onC}$ – Ongoing-conditions Model

Enforcement of conditions while rights are in active use is supported by the $UCON_{onC}$ model by means of the $on\mathcal{C}$ predicate. For example, if the system status changes to ‘emergency mode’ access by certain kinds of users may be terminated. Likewise if the system load exceeds a specified value access may be aborted.

4 UCON Architectures

In architectural point of view, one of the most critical issues in enforcing UCON is the reference monitor. The reference monitor has been discussed extensively in access control community and is a core concept that provides control mechanisms on access to or usage of digital information. Reference monitor associates decision policies and rules for control of access to digital objects. It is always running and tamper resistant. Subjects can access digital objects only through the reference monitor. In this section, we discuss a conceptual structure of UCON's reference monitor and compare the differences from traditional reference monitor. Also, we discuss some architectural variations of UCON systems based on the utilization of reference monitors.

4.1 Structure of Reference Monitor

ISO has published a standard for access control framework [ISO/IEC 10181-3] that defines reference monitor and trusted computing base [4]. According to the standard, reference monitor consists of two facilities; access control enforcement facility (AEF) and access control decision facility (ADF). Every request is intercepted by AEF that asks an ADF for a decision of the request approval. ADF returns either 'yes' or 'no' as appropriate. Reference monitor is a part of trusted computing base, always running, temper-resistant, and cannot be bypassed.

UCON reference monitor is similar but different in detail from traditional reference monitor of ISO's access control framework. Figure 5 shows the conceptual structure of UCON reference monitors. UCON reference monitor consists

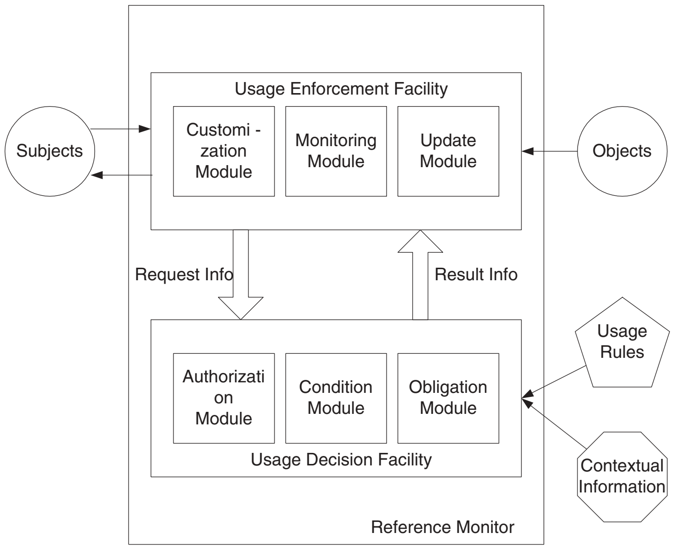

Fig. 5. Conceptual Structure for UCON Reference Monitor

of *Usage Decision Facility (UDF)* and *Usage Enforcement Facility (UEF).* Each facility includes several functional modules. UDF includes conditions and obligations decision modules as well as authorization module. Authorization module takes care of a process similar to traditional authorization process. It utilizes subject and object information (attributes) and usage rules to check whether the request is allowed or not. It may return yes or no. It may return metadata information of authorized portion of requested digital objects along with allowed rights. Then, this metadata information is used for customization of requested digital objects by customization module of UEF. Condition module decides whether the conditional requirements for the authorized requests are satisfied or not by using usage rules and contextual information (e.g., current time, IP address, etc). It may limit rendering devices (e.g., CPU-ID, IP address), rendering time (e.g., business hour, on-duty), etc. Obligation module decides whether certain obligations have to be performed or not before or during the requested usage has been performed. If there exists any obligation that has to be performed, this must be monitored by monitoring module and the result has to be resolved by update module in UEF. Note that usage decision rules may or may not be hardwired into decision facility. Those rules can come along with related digital information or independently [5,2]. Utilization of these modules largely rely on the target application systems' requirements.

4.2 Architectural Classification

Based on the location of reference monitor, there can be *Server-side Reference Monitor (SRM)*, and *Client-side Reference Monitor (CRM).* Here, server is an entity that provides a digital object and client is an entity that receives and uses the digital object. Like a traditional reference monitor, a SRM resides within server system environment and mediates all access to digital objects. On the other hand, a CRM resides in the client system environment and controls access to and usage of digital objects on behalf of a server system. SRM and CRM can coexist within a system. The trustworthiness of CRM is considered relatively lower than that of SRM. Therefore, the main concern here is how reliable and trustworthy the CRM is. In fact, if the client-side computing device is fully functional and general-purpose, CRM is likely to be manipulated with relatively less effort. Therefore, CRM is more suitable to applications with less assurance requirements. This may be improved by using tamper-resistant add-on hardware devices such as dongles, smartcards, etc. On the other hand, if the client device is limited in its functionality and dedicated to specific purposes such as e-book reader or DVD player, CRM is relatively secure from unauthorized manipulations so applications with relatively high assurance requirements are more suitable. After all, the implementation of reference monitors largely depends on business models and their application requirements. For real world implementations, the chances are that both CRM and SRM are likely to be used for better functionality and security. In the following subsections these SRM-only, CRM-only, and SRM & CRM architectures are briefly discussed.

SRM-only Architecture. A system with SRM-only facilitates a central means to control subjects' access to and usage of digital information objects. A subject can be either within same organization/network area or outside this area. In this environment a digital object may or may not be stored in client-side non-volatile storage. If the digital object is allowed to reside in client-side non-volatile storage, it means the saved client copy of the digital object is no longer UCON's target object and doesn't have to be controlled. It can be used and changed freely at client-side. For example, an on-line bank statement can be saved at a customer's local machine for his records and the server system (bank) doesn't care about customer's copy as long as the bank keeps original account information safe. However if the content of digital information itself has to be protected and controlled centrally, the digital information must remain at server-side storage and never be allowed to be stored in cleartext on client-side non-volatile storage. Traditional access control and trust management mainly utilize this kind of system.

CRM-only Architecture. In a system with CRM-only environment, no reference monitor exists on server-side system. Rather, a reference monitor exists at the client system for controlling usage of disseminated digital information. In this environment digital objects can be stored either centrally or locally. The usage of digital objects saved at the client-side is still under the control of CRM in lieu of the server. Without SRM, a digital object cannot be customized for specific users for distribution. Hence, this system is likely to be suitable for B2C mass distribution environments such as e-book systems or MP3 music file distribution. However this doesn't mean that every user will have same usage rights. Distributed digital objects are associated with certain usage rules and users have to prove they have sufficient credentials to exercise certain rights on the objects. At this point users may be limited to perform certain rights on the object under certain conditions such as a specific device identity.

Digital rights management solutions mainly utilize CRM in their systems. In real world implementation, CRM is likely to be embedded within application software where digital objects can be rendered. One example is Acrobat Reader with "Webbuy" plug-in. Webbuy functions as a CRM. Digitally encapsulated PDF files can be viewed through Acrobat Reader with Webbuy. Webbuy controls access to the contents based on a valid license called Voucher. A Voucher may include a specific CPU-ID to restrict rendering devices.

SRM & CRM Architecture. By having SRM in addition to CRM, this architecture can provide two-tier control. SRM may be used for distribution related control while CRM can be used for a finer-grained control on usages. For instance, in SRM, digital objects can be pre-customized for distribution and the distributed, pre-customized digital objects can be further controlled and customized for clients' usages by CRM. As a result, server can reduce or eliminate unnecessary exposure of digital objects that do not have to be distributed. Suppose we have an intelligence system with this architecture. If an unclassified user requests certain digital information that includes some secret information as well, SRM can pre-customize the requested objects before distribution so the distributed version of the objects don't include any secret information. Any finer-control on the distributed objects can be done by CRM at client side. In real world applications, functional specifications of UCON reference monitor can be divided into SRM and CRM in various ways based on the system's functional and security requirements.

5 Conclusion

Classic access matrix based access control has been studied for over thirty years with great attention from the information and computer security community. Nevertheless, there is increasing realization that this model is not adequate for modern application requirements. Researchers have studied various extensions to classic access control concepts. These studies are specific to target problems and thereby seemingly ad-hoc. Unlike these solutions, usage control is comprehensive enough to encompass traditional access control and modern access control such as digital rights management applications. We believe usage control will provide a solid foundations for a robust framework for next generation access control.

References

1. Bell, D. and LaPadula, L.: Secure computer systems: Mathematical foundations and model. MITRE Report, 2(2547) (November 1973)
2. Erickson, J.S.: Fair use, drm, and trusted computing. Communications of the ACM, 46(4) (2003) 34–39
3. Harrison, M.H., Ruzzo, W.L., and Ullman, J.D.: Protection in operating sys- tems. Communications of the ACM, 19(8) (1976) 461–471
4. Security frameworks for open systems: Access control framework. Technical Report ISO/IEC 10181-3, ISO (1996)
5. Jaehong Park, Ravi Sandhu, and James Schifalacqua: Security architectures for controlled digital information dissemination. In Proceedings of 16th Annual Computer Security Application Conference (December 2000)
6. Jaehong Park and Ravi Sandhu. Towards Usage Control Models: Beyond Traditional Access Control, In Proceedings of 7th ACM Symposium on Access Control Models and Technologies (June 2002)