The ABC Core Model for Usage Control: Integrating Authorizations, oBligations, and Conditions

JAEHONG PARK
George Mason University
and
RAVI SANDHU
NSD Security and George Mason University

In this paper, we introduce the family of ABC (*Authorizations, oBligations, and Conditions*) models for *usage control (UCON)*. We call these core models because they address the essence of usage control, leaving administration, delegation and other important but second-order issues for later work. The term usage control 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 users' access to target resources. Obligations are requirements that have to be fulfilled by obligation subjects 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. Although they have been discussed occasionally in recent literature, most authors have been motivated from specific target problems and thereby limited in their approaches. The ABC model integrates these diverse concepts in a unified framework. Traditional authorization decisions are generally made at the time of requests but hardly 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.

Unlike other studies that have targeted on specific problems or issues, the ABC model seeks to enrich and refine the access control discipline in its definition and scope. The ABC model covers traditional access controls such as mandatory, discretionary and role-based access control. Digital rights management and other modern access controls are also covered within the model. We believe our ABC core model for UCON lays the foundation for next generation access controls that are required for today's real world information and systems security. This paper articulates the core of this new area of UCON and develops several detailed models.

Categories and Subject Descriptors: D.4.6 [**Operating Systems**]: Security and Protection—*Access controls*; K.6.5 [**Management of Computing and Information Systems**]: Security and Protection—*Unauthorized access*

This research was partially supported by the National Science Foundation.
An earlier version of this paper appears under the title "Towards Usage Control Models: Beyond Traditional Access Control" in the *Proceedings of 7th ACM Symposium on Access Control Models and Technologies*, Monterey, CA, USA, 2002.
Author's address: Jaehong Park and Ravi Sandhu, Laboratory for Information Security Technology (LIST), ISE Department, George Mason University, Mail Stop 4A4, Fairfax, VA 22030; email: jaehpark@ise.gmu.edu; sandhu@gmu.edu; http://www.list.gmu.edu.

General Terms: Design, Security

Additional Key Words and Phrases: usage control, access control, DRM, trust, privacy

1. INTRODUCTION

Technological innovations in computers and networks have enabled pervasive availability and usability of digital information bringing us opportunities for new business models and personal life styles. Because of these innovations, digital information no longer stays in computer systems only. It is now available on many other devices such as mobile devices (PDA, cell phone, MP3 player, etc), and Internet-integrated home appliances (refrigerator, microwave machine, etc) utilizing various communication methods such as CDs, DVDs, memory cards, mobile messaging, LANs, global Internet networks (either wired or wireless), etc. This pervasive computing phenomenon has raised several new challenging issues for reliable and trusted controls on the usages of digital resources throughout their life cycle.

The issue of usage control on digital resources can be approached from several viewpoints. In the following subsections we examine each of these approaches and their characteristics to motivate our approach. First we explore these issues in traditional access control viewpoint, then in modern access control and digital rights management point of view. Then we discuss our approach to usage control.

1.1 Traditional Access Control Viewpoints

In computer and information security history, there have been many attempts to achieve trusted controls on digital resource usage. The earliest approach was traditional access controls such as mandatory access control (MAC), discretionary access control (DAC), and role-based access control (RBAC). Access control remains a major challenge for computer and information security in modern cyberspace. Providers of services, resources and digital content need to selectively determine who can access these and exactly what access is provided. This is the central objective of access control.

Over the past thirty plus years there has been much progress in access control, but at its core the academic perspective has largely remained unchanged and centered around the access matrix model [Lampson 1971; Landwehr 1997]. The essential concept of the access matrix is that a right is explicitly granted to a subject to access an object in a specific mode, such as read or write. This right exists whether or not the subject is currently accessing the object. Moreover, the presumption is that the right enables repeated access until it is explicitly revoked. In practice the access matrix is never explicitly represented. Instead access control lists (ACLs), capabilities or access relations are used. Groups and roles, possibly in partially ordered seniority relationships, are used to further simplify the actual representation of rights. A variety of discretionary, mandatory and role-based access control models have emerged to accommodate a diverse range of real-world access control policies. In a sense the practice of access control has grown further and further away from the access matrix abstraction. But the core idea that access is driven by rights granted to a subject to access an object remains. On the theoretical side the

seminal work of [Harrison et al 1976] established Turing completeness of the access matrix. While this is a negative result for purposes of safety analysis, it formally establishes the open-ended expressive power of the access matrix.

In recent years several researchers have proposed extensions to the basic access matrix notion of subjects, rights and objects. These have typically come from specific perspectives and the extensions have tended to emphasize the particular system or application focus of the authors. In distributed system the notion of a principal's identity and the meaning of an access control list entry granting particular access to a principal was no longer as simple as in earlier timesharing systems [Abadi et al 1993]. Traditionally, access control has focused on the protection of computer and information resources in a closed system environment. The enforcement of control has been primarily based on identities and attributes of known users by using a reference monitor and specified authorization rules [Sandhu and Samarati 1994]. In today's network-connected, highly dynamic and distributed computing environments, digital information is likely to be used and stored at various locations, hence has to be protected regardless of user location and information location. Relaxing closed system requirement introduces the need to control access by previously unknown users. B2C mass distributions such as e-book systems or music file distributions are also examples of stranger's usages.

1.2 Modern Access Control and Digital Rights Management Viewpoints

With the advent of public-key infrastructure recent research in authorizations for strangers' usages have been pursued under the name of trust management [Blaze et al. 1996; Herzberg et al. 2000; Winsborough et al. 2000; Weeks 2001]. In many cases, trust management utilizes a user's capabilities or properties for authorization in the form of digital credentials or certificates. However, both traditional access controls and trust management have focused on protecting digital resources within server systems and do not deal with client-side controls for locally stored digital information. More recently by utilizing some forms of *client-side reference monitor*, and by focusing on controlling usage of already disseminated digital objects, the arena of Digital Rights Management (DRM) has brought out a significant new perspective on access control problems [Sibert et al. 1995; Kaplan 1996; Rosenblatt et al 2002; Wang et al. 2002]. The DRM requirement for persistent access control [Schneck 1999] and the difficulty of achieving this on a mass-scale is a significant complication. To enable trusted client-side computing there have been industry initiatives such as Microsoft's Palladium and Intel-driven Trusted Computing Platform Alliance (TCPA)[TCPA 2002] which were partly originated from AEGIS [Arbaugh 1997]. Palladium and TCPA have gained serious attention and concern because of their potential impacts on security and privacy issues as well as DRM [Anderson 2002]. DRM is likely to utilize this kind of trusted computing base as a critical enabling technology.

DRM technologies have emerged in mid 90's and gained notable public attention recently. In Jan/Feb. 2001 issue of MIT Technology Review, DRM has been recognized as one of the top 10 emerging technologies that will change the world [MIT 2001]. Because of DRM's potential opportunity for commercial sector, current DRM solutions have been largely driven by commercial entities and are mainly focused on intellectual property rights protection which is based on payment func-

tions. For example, DRM has hardly recognized commercial B2B transactions in their solutions though its underlying technologies can be used for controls on this kind of sensitive information usage. For last several years, many companies have developed various technical solutions for DRM implementations. Many underlying technologies such as watermarking technologies, use-control technologies (including client-side software, server-side encapsulation software, etc.) have been studied. Some rights expression languages (e.g., XrML, ODRL, etc.) also have been developed [ContentGuard 2002; Iannella 2002]. While these DRM techniques and mechanisms have dominated recent DRM studies, many researchers now believe that there is a fundamental unity between DRM and access control. That is DRM is not just a collections of enabling technologies but also about business and security-related policies and models for usage decisions and controls [LaMacchia 2002]. Studies on well-defined, comprehensive models and policies for access control and DRM will provide a foundation for more trusted and secure computing environment.

On a different front several authors have realized that classic access control is inadequate for modern applications. The notion of provisional authorization [Kudo and Hada 2000; Jajodia et al. 2001] states that authorization is not complete until the subject carries out some action to make the authorization effective. The notion of task-based authorization [Thomas and Sandhu 1997] treats all rights as consumable and brought into being just-in-time. In this view a right is a one-time (or k-time) permission obtained in context of enterprise activity. Exercise of a consumable right can enable other rights for different subjects and objects. Access control policy can be seen as one policy amongst many that need to be managed and enforced in distributed system. The Ponder system [Damianou et al. 2001] is a state-of-the-art example of work on policy languages that include but also transcend access control issues. A policy framework directly oriented towards access control but with a number of very useful extensions such as conditions and side-effects has been recently published [Ryutov and Neuman 2001; 2002]. These authors also seek to develop an API and extended ACL based implementation of a portion of their model. From an application perspective the healthcare domain continues to provide significant challenge for traditional access control because of the complexity of its policies and the number of different parties with different interests [Baker et al. 1997; HHS 2002].

1.3 Usage Control Approaches

The research cited above encompasses many significant achievements in specific target problems. At the same time the specific focus has resulted in lack of comprehensiveness and systematic treatment of fundamental issues. Studies on access controls, trust management, and DRM have followed their own tracks and have rarely influenced each other. Although traditional access controls have shown limitations to cover modern digital environments, there has been noteworthy work on security policies and models for controlling digital resources. Similarly though DRM has opened up closed system restrictions, the discipline still lacks well-defined policies and models. Also, current rights expression languages cannot express transaction-level controls including mutability and continuity aspects. Today's DRM technology requires well-defined policies and models that can express usage decisions more

comprehensively and can cover sensitive information protection as well as intellectual property rights protection. Access control and trust management require enlargement of their scope to enable richer, finer and persistent controls on digital objects regardless of their locations. Furthermore, none of these approaches adequately address privacy issues.

Usage Control (UCON) is a conceptual framework that covers these areas in a systematic manner to provide a general-purpose, unified framework for protecting digital resources. UCON is not a substitute for traditional access control, trust management, or digital rights management. Rather, UCON encompasses these three areas and goes beyond in its definition and scope. Also, UCON achieves fine-grained control on digital resources even after the objects have been disseminated. In this paper, we introduce the ABC (Authorizations, oBligations, and Conditions) model family as a core model for usage control that covers these aspects in a single framework systematically and comprehensively. We believe the ABC model of usage control lays the foundation for both next generation access controls and trustworthy digital rights management. Section 2 discusses characteristics of usage control and its scope. Section 3 and 4 identifies core components of the ABC model and develops a family of detailed models. In section 5, we shows how traditional access control, trust management, and digital rights management can be achieved in the ABC model. Then, we discuss administrative aspects of UCON, some UCON examples, and other related issues in section 6. Section 7 discusses related work and section 8 gives our conclusions.

2. USAGE CONTROL

The goal of usage control is to provide a new intellectual foundation for access control. As discussed above, the thirty year old framework of the access matrix has been extended in various different directions as researchers have found it to be inadequate for their needs. The net result is a plethora of seemingly ad hoc extensions without underlying intellectual unity. In this paper we propose a fresh look at the fundamental nature of access control itself. Because we are fundamentally rethinking and extending the essential nature of access control we coin the term **usage control** to convey the broader perspective we are taking. The notion of usage control has been introduced in our previous paper [Park and Sandhu 2002]. The concept of usage control is comprehensive enough to encompass traditional access control, trust management, and digital rights management. Usage control unifies these areas systematically in a single framework and goes beyond in its scope. Figure 1 shows UCON's coverage and its relationships to other research areas. In terms of objectives, sensitive information protection has been one of the most important goals of traditional access control. Recent studies on controlling usages of digital resources have focused on other goals as well, such as Intellectual Property Rights (IPR) protection, and privacy protection. Controlling usage of sensitive information requires protection of digital information that may be critical to nations or organizations. Intelligence community and B2B transaction are good examples for this purpose. IPR protection or digital copyrights protection is relatively a new goal. Content providers' interest largely belongs here so they can realize maximum revenue. Privacy has been rarely studied in the context of con-

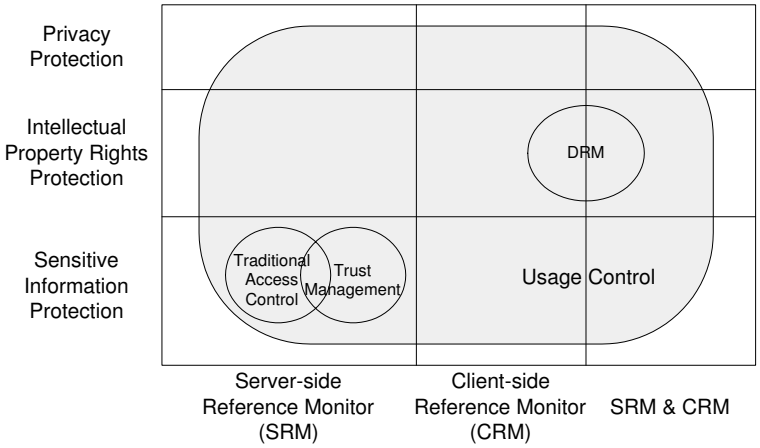

Fig. 1. UCON Coverage

trolling usage of digital information but is beginning to get more public attention. W3C's recent P3P project is one example for privacy support in Web services [P3P 2002]. Healthcare information system is another good example that should consider privacy as a major concern. UCON is objective-neutral and covers all these purposes in a systematic way.

The term usage control has a couple of connotations. In the DRM context it conveys the sense that digital content is provided for use on the end-user's system, but the provider would like to retain some control over what the user does with the bits. In the privacy context the situation is reversed. It is the end-user who often provides personal information to a service provider, and would like to control how the service provider can use that information. Sometimes the personal information is provided by a third-party originator, say a health-care provider, but the individual, called 'identifiee', to whom it pertains would nevertheless like to exercise some control over its use. Usage also has a connotation of duration, so the access may continue for some time. In classic access control the usual viewpoint is that access is enforced before access is granted and then access persists for some duration without any further checks. This is appropriate for traditional access control systems but does not reflect many modern e-business cases.

In usage control, target objects have relationships with consumer, provider, and identifiee subjects. The consumer subject seeks access to a target object provided by a provider subject. The target object may contain privacy-related information of subjects. These subjects are called identifiee subjects and hold certain rights on the object. Usage decision is based on relationships among these different subject parties on target resources. Ideally, these relationships may no longer be one-way control decisions which is the usual case today where provider determines consumer's access. Having multi-way control requires active involvement of each of these three parties in decision-making process. While this may be an ideal approach of usage control, this paper is a first step in this arena and only covers core aspects of usage control without considering relationships among different parties or multi-

way controls. The core part of usage control deals with decision-making aspects of consumer subject usages. In this paper we introduce the ABC model as a core model of usage control. The ABC model mainly discusses basic control issues of consumer subjects' usage on target objects and does not cover any issues of the relationships among different subject parties nor related administrative issues.

Traditionally, access control has dealt with authorizations as the basis for its decision-making process. In the ABC model, the authorization-based decision process utilizes subject attributes and object attributes. Attributes can be identities, security labels, properties, capabilities, etc. The ABC model includes obligations and conditions as well as authorizations as part of usage decision process to provide a richer and finer decision capability. The necessity of obligations and conditions has been recognized in modern business systems such as B2C mass distribution systems as well as B2B transactions and interactions between business partners. Obligations are requirements that have to be fulfilled for usage allowance. Conditions are environmental requirements that are independent from individual subjects and objects. These decision predicates can be evaluated before or during exercise of a request. In addition, usage of target object may require certain updates on subject or object attributes before, during or after a usage exercise (e.g., reducing a requester's account balance by the value of an e-book). UCON covers these issues within its ABC core model in a systematic manner.

In architectural point of view, traditional access controls have focused on server-side controls only, hence barely considered *client-side controls* which gives an ability to control usages persistently even after the digital resources are distributed. The ABC model can be utilized in both server-side and client-side control architectures though some functional details are likely to be different. Client-side control requires the existence of client-side trusted computing base and reference monitor. Recent studies on client-side controls include Palladium and TCPA as mentioned previously. Trustworthiness of client-side reference monitor is a relative issue and largely dependent on requirements of business models including privacy issues. The detail of client-side reference monitor is not discussed in this paper. The main focus of this paper is on model level discussion and does not consider architectural or mechanistic details.

3. ABC MODEL COMPONENTS

The ABC models consist of eight core components (see Figure 2). They are subjects, subject attributes, objects, object attributes, rights, authorizations, obligations, and conditions. Authorizations, obligations and conditions are functional predicates that have to be evaluated for usage decision. Each predicate can be divided into detailed predicates. Subjects, objects and rights can be divided into several detailed components with different perspectives. Traditional access controls utilize only authorizations for decision process. Obligations and conditions are new concepts that have been discussed recently to resolve certain shortcomings shown in traditional access controls. These three decision factors will be used for the development of various detailed models.

A significant innovation in ABC is that subject and object attributes can be mutable. **Mutable attributes** are changed as a consequence of access, whereas

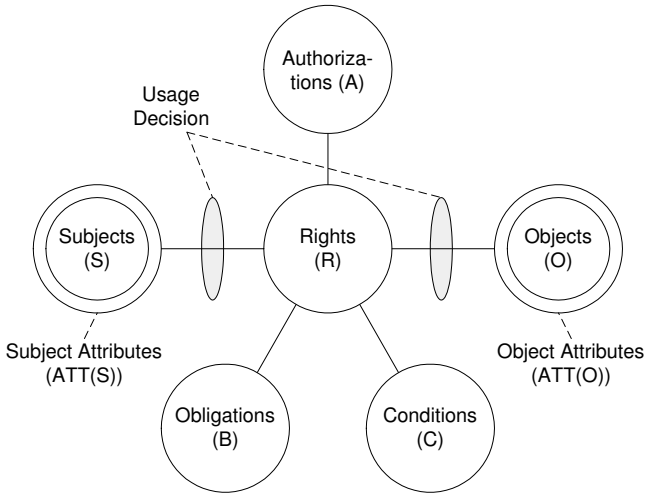

Fig. 2. ABC Model Components

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. Mutable attributes add further complication to the requirement for obtaining timely values of attributes from a trusted source, since now the attributes must also be modifiable in a trusted way.

3.1 Subjects (S) and Subject Attributes (ATT(S))

A subject is an entity associated with attributes, and holds or exercises certain rights on objects. For simplicity, a subject in usage control can be regarded as representing an individual human being. A subject is defined and represented by its attributes. Subject attributes are properties or capabilities of a subject that can be used for the usage decision process. A subject may or may not have a unique identity. If authorization is done with a user's unique identity, accountability can be provided. If not, anonymity can be supported. Some attributes such as pre-paid credits, or usage capabilities can be used without unique identity for anonymous usages or transferable rights. Examples of subject attributes include identities, group names, roles, memberships, security clearance, etc. A group is a set of users who holds same rights as a group. A role is a named collection of users and relevant permissions [Sandhu et al. 1996]. Groups and roles may have hierarchical relationships. 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.[1]

If an attribute is *immutable*, it cannot be changed by the user's activity. Only administrative actions can change such an attribute. A *mutable* attribute can be modified as a side effect of subjects' access to objects. Many examples of trust management and DRM are likely to utilize mutable attributes. Some examples of mutable attributes are credits/capabilities (eg., $10 worth usage, five times per day, print twice), security clearance with relaxed (weak) or no tranquility, usage log (eg., already read portion cannot be read again), etc.

In usage control, the subjects can be *consumer subjects (CS)*, *provider subjects (PS)*, or *identifiee subjects (IS)*. Consumer subjects are entities who exercise the rights to access the objects. An e-book reader, MP3 music listener and even a distributor of digital objects can be a consumer subject. Provider subjects are entities who provide an object and hold certain rights on it. Examples of provider subjects include an author of an e-book, a distributor of the book, a primary physician, etc. The identifiee subjects are entities who are identified in digital objects that include their privacy-sensitive information. A patient in health care system is an example of an identifiee subject. Although the concept of identifiee subjects always exists in case of privacy-sensitive information, identifiee subjects may or may not be included within UCON systems based on the system requirements or policies.

3.2 Objects (O) and Object Attributes (ATT(O))

Objects are a set of entities that subjects hold rights on, whereby the subjects can access or use objects. Objects are also associated with attributes, either by themselves or together with rights. As for subjects, object attributes include certain properties that can be used for access decisions. Examples of object attributes that are associated with objects are security labels, ownerships, classes, etc. Object classes can be used to categorize objects so authorization can be done based not only on individual objects but also sets of objects that belong to same class [Sandhu and Samarati 1994]. Examples of attributes for objects with rights are values, role permissions, etc. The values may be used to define how many credits are required to obtain a certain right on a specific object. For example, "Harry Potter" e-book together with a 'read' right may require $10 or the book with an additional 'print' right may require $15. Object attributes also can be mutable (eg., the number of play time on each item of music).

In UCON, objects can be either privacy sensitive or privacy non-sensitive. A privacy-sensitive object includes individually identifiable information that can cause privacy problems if not used properly. An UCON object can be either original or derivative. The derivative object in UCON is different from that of other DRM literature. DRM's derivative objects are more like "reused" or "reproduced" objects. In DRM, the term "derivative" means derived (cited, quoted, or copied) from an

[1]Using attributes for access decisions requires a trusted source for the values and their timeliness. There are many challenges to achieving this. The details of how this would be accomplished is an important implementation and architectural issue. Nonetheless, in our view it is important to keep the model separate from these implementation concerns, however important they may be. Keeping the model separate from implementation is critical to separating concerns and reaching fundamental understanding of disparate issues [Sandhu 2000]. This viewpoint will be sustained throughout the paper.

original work to create another digital work that includes parts of the original work. In UCON, however, the derivative object is an object that is created in consequence of obtaining or exercising rights on an original object. For example, playing MP3 music file can create usage log information. This log data file is called a derivative object in UCON. To provide mutual protection on the rights of all involved subjects (consumer, provider and/or identifiee subjects), just like the original object, these derivative objects also have to be considered as target objects and must hold UCON properties and relations with other components. Based on their format, objects can be documents (e.g., .doc, .pdf, .ps), audio (e.g., .mp3, .wav), video (e.g., JPEG, DVD, MPEG), executable files (e.g., games), etc. Each may require its own application tools to be used.

3.3 Rights (R)

Rights are privileges that a subject can hold and exercise on an object. Rights consist of a set of usage functions that enables a subject's access to objects. Rights may or may not have a hierarchy. Like subjects and objects, rights can also be divided into consumer rights (CR), provider rights (PR), and identifiee rights (IR). In an access control viewpoint, rights enable 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. However 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.[2] The **usage decision functions** indicated in figure 2 make this determination based on subject attributes, object attributes, authorizations, obligations and conditions. In general, rights include rights for direct use of objects (such as read), delegation of rights and rights for administering access (such as modify subject and object attributes that in turn determine access rights). In this paper we do not consider delegation rights and administrative rights. Rights can be divided into many functional categories. The two most fundamental rights categories might be view and modify, possibly augmented with creation and deletion. We can also distinguish direct access rights that are used by a subject to access an object from administrative rights that are used to administer access as well manage the object. The Digital Rights Management community has published several studies on functional rights [ContentGuard 2002; Gunter et al. 2001; Iannella 2002], categorizing them as render rights, transport rights, derivative works rights and utility

[2]One could argue that the access matrix is a conceptual entity and does not exist as such. Nonetheless, the traditional position has been that the access matrix is what we are enforcing. The embodiment of the access matrix by Access Control Lists (ACLs), Capabilities or a Access Relation is a means of representing a sparse data structure efficiently. In actual practice the rights of a subject to an object are often determined when the access is attempted, e.g., the ACL may authorize a group to read an object and the subject's membership in the group is determined by subject's attributes at access time. So the ABC viewpoint is more accurate with respect to actual practice. The ABC viewpoint has consequences for access review. Predicting which rights will be available when access is attempted becomes a problem more akin to safety analysis with respect to leakage of rights [Harrison et al 1976], than a simple lookup of relevant data structures. The ABC view is more accurate with respect to real-world access control systems where the actual representation of rights is rarely as straightforward as in the access matrix.

rights [Rosenblatt et al 2002]. More generally, rights can be defined by specific applications such as credit and debit in an accounting application.

3.4 Authorizations ($\mathcal{A}$)

Authorizations are functional predicates that have to be evaluated for usage decision and return whether the subject (requester) is allowed to perform the requested rights on the object. Authorizations evaluate subject attributes, object attributes, and requested rights together with a set of authorization rules for usage decision. Authorizations can be either pre-authorizations ($preA$) or ongoing-authorizations (onA). $preA$ is performed before a requested right is exercised and onA is performed while the right is exercised. onA may be performed continuously or periodically during the time span of access. In general, most traditional access control policies including MAC, DAC, RBAC and Trust Management (TM) utilize some form of pre-authorization for their decisions. Also, some DRM decision processes are pre-authorizations. Although rarely implemented, an example of ongoing authorization would be the continued checking of revocation status during the exercise of usage. Thereby usage can be immediately terminated to enforce immediate revocation.

Certain authorizations may require updates on subject attributes and/or object attributes. These updates can be either pre, ongoing, or post. Security clearance with high watermark property requires updates before usage is performed. Metered usage payment requires updates after the usage is ended to calculate current usage time. Using pre-paid credits for usage time based metered payment requires periodic updates of the credits during the access to prevent overuse.

3.5 oBligations ($\mathcal{B}$)

Obligations are functional predicates that verify mandatory requirements a subject has to perform before or during a usage exercise. Obligations can be either pre-obligations ($pre\mathcal{B}$) or ongoing-obligations ($on\mathcal{B}$). $pre\mathcal{B}$ is a predicate that utilizes some kind of history functions to check if certain activities have been fulfilled or not and returns either 'true' or 'false'. A user may have to fill out some personal information before reading a company's white paper. Similarly, a user may have to agree to provide usage log information before listening to music files. $on\mathcal{B}$ is a predicate that has to be satisfied continuously or periodically while the allowed rights are in use. A user may have to keep watching certain advertisements while he has logged in. Obligations may or may not utilize subject or object attributes. Attributes can be used to determine what kind of obligations are required for usage approval. Obligations may require certain updates on subject attributes. These updates are likely to affect either current or future usage decisions. Note that attributes are not used for decision making with respect to obligations, but only for choosing what obligations apply.

One of the basic assumptions in the ABC model is that its decision-making process is transaction-based. This means that decision predicates are evaluated upon each usage request and the decision influences usages of that request. A *pre* decision predicate decides approval or denial of the request. An *ongoing* predicate may revoke or continue to allow current exercise of the requested usage. [3]

[3]Unlike authorizations or conditions, there can be transaction-independent, global obligations

Our obligations are different from duties in that duties are assigned to subjects regardless of the subjects' requests, and essential for an organization, whereas our obligations are requirements that have to be fulfilled and checked before or during the usage of certain rights. Traditional access control has hardly recognized the obligation concept. Some DRM solutions include obligation functions though many of them implement the obligation functions only partially or implicitly. The ABC model does not include duties. We feel that including duties within the models will distract our original purpose and cause unnecessary complexity.

3.6 Conditions ($\mathcal{C}$)

Conditions are environmental or system-oriented decision factors. Condition predicates evaluate current environmental or system status to check whether relevant requirements are satisfied or not and return either 'true' or 'false'. Subject attributes or object attributes can be used to select which condition requirements have to be used for a request. However no attribute is included within the requirements themselves. Unlike authorizations or obligations, condition variables cannot be mutable since conditions are not under direct control of individual subjects. Evaluation of conditions cannot update any subject or object attributes. Some examples of condition requirements include current local time for accessible time period (e.g., business hours), current location for accessible location checking (e.g., area code, device, CPU-ID), security status of the system (e.g., normal, high alert, under attack), system load, etc.

In the ABC model, one may separate a device from conditions and consider it as a different component of the model just like other components such as subjects, objects, and rights. Intuitively, since majority of current computing systems are quite mobile and network-connected, and digital information is virtually available anywhere, usage rules can specify allowed devices explicitly along with subjects, objects, and rights. While this may be true, we prefer to include device component within conditions since there are other subject and object independent factors such as time periods, and system load.

Conditions are different from authorizations in that conditions mainly focus on evaluations of environmental, system-related restrictions that have no direct relationship with subject and object attributes for usage decision (that is, subject and object attributes are not included within condition requirements hence not required for usage decision process) whereas authorizations evaluate attributes that are related to subjects (requesters) or requested objects for usage decision. [4]

---

where the obligations influence only future requests and have no affects on the current request decision. For example, a user may have to fill out monthly evaluation reports for continuous subscription of a digital library, or a user may have to provide usage log information to a provider. These global obligations have to be fulfilled for future usages in timely manner (either time-based or event-based). Such global obligations are outside the scope of ABC core models.

[4]It should be noted there are some ambiguities as to how specific items should be treated. The IP address of a client can be viewed as a subject attribute, but can also be viewed as a condition indicating the location of the client. Whether the IP address is viewed as a subject attribute or a condition element is a choice that the system architect has to make. Rather than trying to provide air-tight boundaries between these concepts, we recognize them as somewhat fuzzy. It is generally true that a rich model will accommodate multiple ways of specifying a given policy.

	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. 3. The 16 Basic ABC Models

4. THE ABC FAMILY OF CORE MODELS

All of these components makes for a fairly complex model. Nonetheless, we believe, the recognition of three distinct factors, authorizations, obligations and conditions, along with mutability of attributes and continuity of enforcement is critical to supporting modern access control requirements. The resulting complexity is appropriate for modern cyberspace. Based on the eight components discussed above we develop a framework for classifying ABC models. We say these are our core models because, as discussed earlier, they focus on the enforcement process and do not include administrative constructs. Also they will need to be further elaborated for specific applications, as will be discussed later.

Our classification is based on the following three criteria: *decision factors* that consist of authorizations, obligations, and conditions, *continuity* of decision being either pre or ongoing with respect to the access in question, and *mutability* that can allow updates on subject or object attributes at different times. If all attributes are immutable, no updates are possible as a consequence of the decision process. This case is denoted as '0'. With mutable attributes, updates are possible before (pre), during (ongoing), or after (post) the right is exercised, denoted as '1, 2, and 3', respectively. Based on these criteria, we enumerate the model space shown in Figure 3.

Cases that are not likely to be useful in practice are marked as 'N'. If decision factor is 'pre', updates can occur before or after the right is exercised but there is little reason to have ongoing updates. Without ongoing decision, ongoing-update can only influence decisions on future requests and therefore the updates can be done after the usage is ended. For example, suppose Alice is a member of a digital music library. Suppose she has to pay $1/hour of music play. This example can be handled as a pre-authorization with post update case. Her usage time is accumulated on her usage log file as each play ends. This case does not require any updates during the playing of a music track. However, if decision factor is 'ongoing', updates can happen before, during or after the right is exercised. These updates are used for current usage decision. This explains the top four rows of Figure 3. For the bottom two rows the only decision factor is conditions. Evaluation of conditions cannot update attributes by definition. The resulting 16 Y's in Figure 3 define the 16 basic ABC models. The A and B models have 7 Y's each whereas the C model has only 2 Y's. In practice, many real-world systems will use some combination of these models.

This is no different in the case of ABC.

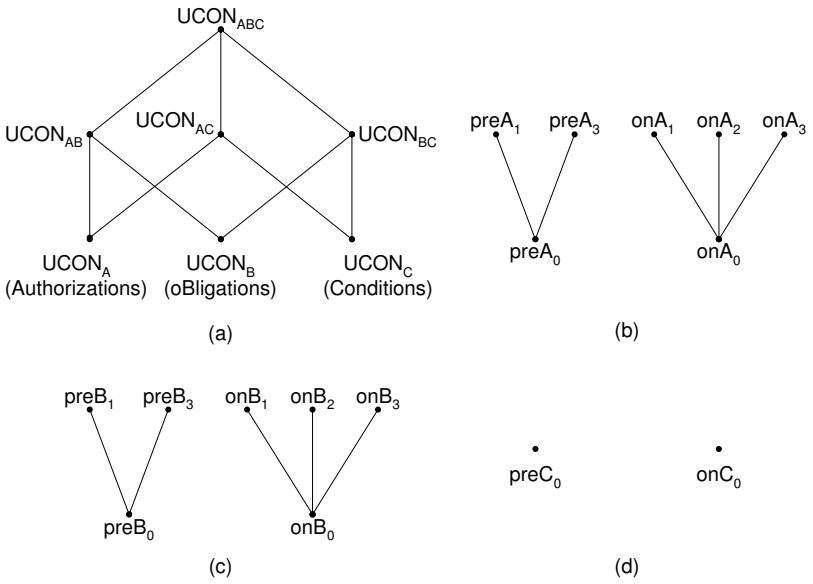

Fig. 4. The ABC Family of Core models

Figure 4(a) shows possible combinations of ABC models and their relationships. We regard each of A, B or C to be on equal footing, hence the three base models at the bottom. At the next level up we have combinations of two of these, and further combination of all three. In this way we can succinctly represent which combination of A, B and C is being used in a given context. Each of the A, B and C models is divided into several cases as respectively shown in Figures 4 (b), (c) and (d). In totality these comprise the 16 Y's of Figure 3. The pre and ongoing cases are regarded as being on equal footing. The case of mutable attributes (1, 2 and 3) always dominates immutable attributes (0), but there is no ordering between the 1, 2 and 3 cases. Figure 4 graphically demonstrates the richness of the model space available in the ABC family.

The ABC model definitions do not express the mechanistic details such as how updates can be enforced. Actual pre-updates may have to be performed either before the requested rights are exercised (e.g., obtaining a lock for mutual exclusion) or right after the rights are started (e.g., e-cash decrease) [Ryutov and Neuman 2001; 2002]. However we do not include such implementation issues in our models following our practice of keeping the model and architecture-mechanism distinct.

In this paper we are not trying to develop a logical expression language for the ABC model. Rather, while there can be numerous ways of expressing the ABC model, our focus is to develop comprehensive models for usage control that can support modern access control requirements as well as DRM in a single framework and discuss the detailed characteristics of these in a systematic manner. Many current DRM solutions utilize obligations and conditions for decision process though they may not define these factors explicitly.

4.1 $UCON_{preA}$ - pre-Authorizations Models

Authorizations have been considered as the core of access control and extensively discussed since the beginning of access control discipline. Traditionally, access control research has focused on pre-authorizations in which a usage decision is made before a requested right is exercised. $UCON_{preA}$ models utilize these pre-authorizations for their usage decision processes. In $UCON_{preA}$ models, an authorization decision process is done before usage is allowed. There are three detailed models based on mutability variations. $UCON_{preA_0}$ is immutable pre-authorization model that requires no update. $UCON_{preA_1}$ is pre-authorization model with an optional pre-update procedure. A pre-update includes update functions that modify attributes before usage is started. $UCON_{preA_3}$ is pre-authorization model with an optional post-update procedure. A post-update utilizes update functions to modify certain attributes after usage is terminated.[5]

The following definitions formalize the $UCON_{preA}$ models. Although some definitions are quite similar and the expressions can be reduced to a certain degree, we explicitly express each of the detail models for completeness of the models. This is done throughout the model definitions in this paper.

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

—$S, O, R, ATT(S), ATT(O)$ and $pre\mathcal{A}$ (subjects, objects, rights, subject attributes, object attributes, and pre-authorizations respectively);

—$allowed(s, o, r) \Rightarrow pre\mathcal{A}(ATT(s), ATT(o), r)$.

Definition 2 The $UCON_{preA_1}$ model is identical to $UCON_{preA_0}$ except it adds following pre-update processes:

—$preUpdate(ATT(s)), preUpdate(ATT(o))$, an optional procedure to perform update operations on $ATT(s)$ and $ATT(o)$, respectively. Note that $preUpdate$ can include non-deterministic operations.

Definition 3 The $UCON_{preA_3}$ model is identical to $UCON_{preA_0}$ except it adds following post-update processes:

—$postUpdate(ATT(s)), postUpdate(ATT(o))$, an optional procedure to perform update operations on $ATT(s)$ and $ATT(o)$, respectively. Note that $postUpdate$ can include non-deterministic operations.

$UCON_{preA_0}$ consists of subjects (S), objects (O), rights (R), subject attributes ($ATT(S)$), object attributes ($ATT(O)$), and pre-authorizations ($pre\mathcal{A}$). $pre\mathcal{A}$ is a functional predicate that utilizes $ATT(S)$, $ATT(O)$, and R for usage decision making. $pre\mathcal{A}$ examines usage requests using $ATT(S)$, $ATT(O)$, and R then decides whether the request is allowed or not. We write $allowed(s, o, r)$ to indicate that

[5]Note that the model does not show the detailed enforcement mechanisms of how updates have to be performed.

subject s is allowed right r to object o. Note that the ABC model formulates 'implies' connectives rather than 'if'. This means that righthand-side of the connective is not sufficient to allow usages but is necessary. A similar approach can be found in Bell LaPadula (BLP) model. In BLP, mandatory access control is formulated as 'necessary condition' and used together with discretionary access control to enforce additional information flow policies [Bell and LaPadula 1973; Sandhu 1993]. Throughout the models, we formulate 'necessary condition' rather than 'sufficient condition' so decision process can include other rules that might be necessary for finer and richer controls.

The meaning of $preUpdate(ATT(s))$ is that subject attributes are updated. Exactly what values can be used in computing the update is left unspecified in the model. These could be subject, object attributes and other variables. Similarly for the other update processes in Definition 2 and 3. Traditional access controls such as MAC, DAC, and RBAC are likely to belong to $UCON_{preA_0}$. Following Examples 1 and 2 respectively show how traditional MAC and DAC can be realized within $UCON_{preA_0}$.

Example 1 MAC policies, $UCON_{preA_0}$:

L is a lattice of security labels with dominance relation $\geq$
$clearance : S \rightarrow L$
$classification : O \rightarrow L$
$ATT(S) = \{clearance\}$
$ATT(O) = \{classification\}$

$allowed(s, o, read) \Rightarrow clearance(s) \geq classification(o)$
$allowed(s, o, write) \Rightarrow clearance(s) \leq classification(o)$

Example 2 DAC closed policies using ACL with an individual ID, $UCON_{preA_0}$:

N is a set of identity names
$id : S \rightarrow N$, one to one mapping
$ACL : O \rightarrow 2^{N \times R}$
$ATT(S) = \{id\}$
$ATT(O) = \{ACL\}$

$allowed(s, o, r) \Rightarrow (id(s), r) \in ACL(o)$

In case of MAC, security labels (clearance and classification) are used as a subject attribute ($ATT(S)$) and an object attribute ($ATT(O)$), and Bell-LaPadula's security properties (simple and star property) are utilized for pre-authorizations ($pre\mathcal{A}$). Here, if the clearance of subject s dominates the classification of object o, 'read' requests are allowed. Similarly, 'write' is allowed if clearance(s) is dominated by classification(o). In case of DAC example, individual (or group) identities and access control lists (ACL) are subject attributes and object attributes, respectively. ACL is a functional mapping of object to multiple ids and rights. If the

subject's identity name together with the requested right exists in ACL, the request is allowed. Although it has been understood that ACL and capability list are used to achieve similar control functionalities, they can actually provide quite different results when they are used with mutable attributes. This is discussed in next section. In RBAC, a user-role and a permission-role can be considered as a $ATT(S)$ and a $ATT(O)$ respectively and compared for authorizations before allowing access. Details of MAC, DAC, RBAC, and other related areas to ABC model are discussed in next section. Also, certain authorization processes of DRM (e.g., membership-based digital library) can be expressed within $UCON_{preA_0}$ as follows. We illustrate two examples of DRM with pre-updates and post-updates respectively.

Example 3 DRM pay-per-use with a pre-paid credit, $UCON_{preA_1}$:

M is a set of money amount
$credit : S \rightarrow M$
$value : O \times R \rightarrow M$
$ATT(s) : \{credit\}$
$ATT(o, r) : \{value\}$

$allowed(s, o, r) \Rightarrow credit(s) \geq value(o, r)$
$preUpdate(credit(s)) : credit(s) = credit(s) - value(o, r)$

Example 4 DRM membership-based metered payment, $UCON_{preA_3}$:

M is a set of money amounts
ID is a set of membership identification numbers
$TIME$ is a current usage minute
$member : S \rightarrow ID$
$expense : S \rightarrow M$
$usageT : S \rightarrow TIME$
$value : O \times R \rightarrow M$ (a cost per minute of r on o)
$ATT(s) : \{member, expense, usageT\}$
$ATT(o, r) : \{valuePerMinute\}$

$allowed(s, o, r) \Rightarrow member(s) \neq \phi$
$postUpdate(expense(s)) : expense(s) = expense(s) + (value(o, r) \times usageT(s))$

In Example 3, if the credit of a subject s is not less than the value of the requested usage, the request is allowed. Once the request is allowed, the subject's credit is reduced by the value of the usage. Example 4 shows membership-based usage controls with metered payment. In this case, a request is allowed if the subject is a member. However, a total expense has to be updated at the end of each usage by using current usage time and value of the usage so it can be paid periodically. [6]

$UCON_{preA_1}$ and $UCON_{preA_3}$ are unchanged from $UCON_{preA_0}$ except the ad-

[6]Enforcement of periodic payment of accumulated expense is not considered as part of the core ABC model since it is not related to a request-based decision process but is part of larger workflow.

ditional update procedures. $UCON_{preA_1}$ and $UCON_{preA_3}$ introduce '*preUpdate*' and '*postUpdate*' procedures to modify subject attributes and object attributes. Many B2B and B2C applications require some form of update functionalities. Recent DRM solutions have also dealt with these. For example, pre-paid credits have to be reduced at the time a user is allowed to exercise requested rights on an object ($UCON_{preA_1}$).

4.2 $UCON_{onA}$ - ongoing-Authorizations Models

In $UCON_{onA}$ model, usage requests are allowed without any 'pre' decision-making. However, authorization decisions are made continuously or repeatedly while usage rights are exercised. If certain requirements become dissatisfied, the currently allowed usage right is revoked and its exercise is stopped. Ongoing authorizations have been seldom discussed in access control literature. By utilizing ongoing authorizations, monitoring is actively involved in usage decisions while a requested right is exercised. This kind of continuous control is especially useful for relatively long-lived usage rights. In $UCON_{onA}$, we develop four detailed models. $UCON_{onA_0}$ is immutable ongoing-authorization model that has no update procedure included. $UCON_{onA_1}$ is ongoing-authorization model with pre-updates. $UCON_{onA_2}$ and $UCON_{onA_3}$ include ongoing updates and post updates, respectively.

The following definitions formalize the $UCON_{onA}$ models.

Definition 4 The $UCON_{onA_0}$ model has the following components:

—$S, O, R, ATT(S), and ATT(O)$ are not changed from $UCON_{preA}$;

—$on\mathcal{A}$ (ongoing-authorizations);

—$allowed(s, o, r) \Rightarrow true$;

—$stopped(s, o, r) \Leftarrow \neg on\mathcal{A}(ATT(s), ATT(o), r)$.

Definition 5 The $UCON_{onA_1}$ model is identical to $UCON_{onA_0}$ except it adds following pre-update processes:

—$preUpdate(ATT(s)), preUpdate(ATT(o))$, an optional procedure to perform update operations on $ATT(s)$ and $ATT(o)$, respectively.

Definition 6 The $UCON_{onA_2}$ model is identical to $UCON_{onA_0}$ except it adds following ongoing-update processes:

—$onUpdate(ATT(s)), onUpdate(ATT(o))$, an optional procedure to perform update operations on $ATT(s)$ and $ATT(o)$, respectively.

Definition 7 The $UCON_{onA_3}$ model is identical to $UCON_{onA_0}$ except it adds following post-update processes:

—$postUpdate(ATT(s)), postUpdate(ATT(o))$, an optional procedure to perform update operations on $ATT(s)$ and $ATT(o)$, respectively.

$UCON_{onA_0}$ model introduces $on\mathcal{A}$ predicate instead of $pre\mathcal{A}$. Since there is no pre-authorization, the requested access is always allowed. However, ongoing-authorizations are active throughout the usage of the requested right, and certain requirements are repeatedly checked for continuous access. Semantically these requirements have to be true all the time while the right is exercised. Technically, these checking processes are likely to be performed periodically based on time or event. In the ABC model, we do not specify these technical and implementation details. In case certain attributes are changed and requirements are no longer satisfied, '*stopped*' procedure is performed. We write '$stopped(s, o, r)$' to indicate rights r of subject s to object o is revoked. In many cases, ongoing authorizations are likely to occur together with pre-authorizations though ABC model does not require this. For example, suppose $on\mathcal{A}$ screens 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 at the time of the request. This can be an example of $UCON_{onA_0}$ model. $UCON_{onA_1}$, $UCON_{onA_2}$ and $UCON_{onA_3}$ are unchanged from $UCON_{onA_0}$ but add pre-updates, ongoing-update and post-update procedures respectively. Some examples of $UCON_{onA}$ models are given below.

Example 5 A limited number of simultaneous usages, revocation using usage start time, $UCON_{onA_{13}}$:

T is an ordered set of current usage start times
UN is a set of concurrent usage numbers
N is a set of identification names
$id : S \rightarrow N$
$usageNum : O \rightarrow UN$
$startT : O \rightarrow 2^{N \times T}$
$ATT(s) : \{id\}$
$ATT(o) : \{startT, usageNum\}$

$allowed(s, o, r) \Rightarrow true$
$stopped(s, o, r) \Leftarrow (usageNum(o) > 10) \wedge$
$(id(s), t) \in startT(o)$ where $t = \min\{t' | \exists s', (id(s'), t') \in startT(o)\}$
$preUpdate(startT(o)) : startT(o) = startT(o) \cup \{(id(s), t)\}$, where s is currently requesting subject of usage
$preUpdate(usageNum(o)) : usageNum(o) = usageNum(o) + 1$
$postUpdate(startT(o)) : startT(o) = startT(o) - \{(id(s), t)\}$, where s is a subject of stopped usage
$postUpdate(usageNum(o)) : usageNum(o) = usageNum(o) - 1$

Example 6 A limited number of simultaneous usages, revocation using longest idle time, $UCON_{onA_{123}}$:

T is an ordered set of last activity times
UN is a set of concurrent usage numbers

N is a set of identification names
$id : S \rightarrow N$
$usageNum : O \rightarrow UN$
$lastActiveT : O \rightarrow 2^{N \times T}$
$ATT(s) : \{id\}$
$ATT(o) : \{lastActiveT, usageNum\}$

$allowed(s, o, r) \Rightarrow true$
$stopped(s, o, r) \Leftarrow (usageNum(o) > 10) \wedge$
$(id(s), t) \in lastActiveT(o)$ where $t = \min\{t' | \exists s', (id(s'), t') \in lastActiveT(o)\}$
$preUpdate(usageNum(o)) : usageNum(o) = usageNum(o) + 1$
$onUpdate(lastActiveT(o))$, repeated updates on $lastActiveT(o)$
$postUpdate(usageNum(o)) : usageNum(o) = usageNum(o) - 1$.

Example 7 A limited number of simultaneous usages, revocation using total usage time, $UCON_{onA_{13}}$:

T is an ordered set of current usage times
TT is an ordered set of total usage times
UN is a set of concurrent usage numbers
N is a set of identification names
$id : S \rightarrow N$
$totalT : O \rightarrow 2^{N \times TT}$, A functional mapping of object to a set of total usage times of active subjects
$usageNum : O \rightarrow UN$
$ATT(s) : \{id\}$
$ATT(o) : \{usageT, totalT, usageNum\}$

$allowed(s, o, r) \Rightarrow true$
$stopped(s, o, r) \Leftarrow (usageNum(o) > 10) \wedge$
$(id(s), tt) \in totalT(o)$ where $tt = \max\{tt' | \exists s', (id(s'), tt') \in totalT(o)\}$
$preUpdate(usageNum(o)) : usageNum(o) = usageNum(o) + 1$
$postUpdate(usageNum(o)) : usageNum(o) = usageNum(o) - 1$
$postUpdate(totalT(o)) : (id(s), tt) = (id(s), tt+t)$, where s is a subject of stopped usage and t is current usage time of the s

In Example 5, 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 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. Here, starting time of each request has to be added before the beginning of the requested usage and has to be taken out after the usage is stopped. Also current usage number of o_1 is increased by 1 at the time of access and decreased by 1 at the end of the access, hence a $UCON_{onA_{13}}$ model. 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 as shown

in Example 6. Further suppose that revocation of the extra user is based on total usage time in completed sessions since the start of the fiscal year. Post-updates would be needed to accumulate current usage time as shown in Example 7.

4.3 $UCON_{preB}$ - pre-oBligations Models

$UCON_{preB}$ introduces pre-obligations that have to be fulfilled at the time of a request and before access is allowed. $pre\mathcal{B}$ is a kind of history function that checks whether certain obligations have been fulfilled or not and return true or false for the usage decision. Suppose a user has to provide his name and email address to download a company's white papers or suppose a user has to click 'O.K.' on a license agreement to access a web portal. Here, the user has to fulfill the required actions before access is allowed. $UCON_{preB}$ models consist of 2 steps. First step is to select required obligation elements for the requested usage. This selection may utilize subject and/or object attributes. Second step is to evaluate whether the selected obligation elements have been fulfilled without any error (e.g., invalid e-mail addresses). In $UCON_{preB}$ models, a request may require multiple pre-obligation elements to be fulfilled. The $pre\mathcal{B}$ predicate evaluates if all the required pre-obligation elements ($preOBL$) are fulfilled by using $preFulfilled$ and returns either true or false.

The following definitions formalize the $UCON_{preB}$ models.

Definition 8 The $UCON_{preB_0}$ model has the following components:

—$S, O, R, ATT(S)$, and $ATT(O)$ are not changed from $UCON_{preA}$;
—OBS, OBO, and OB, (obligation subjects, obligation objects, and obligation actions, respectively);
—$pre\mathcal{B}$, and $preOBL$, (pre-obligation predicates and pre-obligation elements, respectively);
—$preOBL \subseteq OBS \times OBO \times OB$;
—$preFulfilled : OBS \times OBO \times OB \rightarrow \{true, false\}$;
—$getPreOBL : S \times O \times R \rightarrow 2^{preOBL}$, a function to select pre-obligations for a requested usage;
—$pre\mathcal{B}(s, o, r) = \bigwedge_{(obs_i, obo_i, ob_i) \in getPreOBL(s,o,r)} preFulfilled(obs_i, obo_i, ob_i)$;
$pre\mathcal{B}(s, o, r) = true$ by definition if $getPreOBL(s, o, r) = \phi$;
—$allowed(s, o, r) \Rightarrow pre\mathcal{B}(s, o, r)$.

Definition 9 The $UCON_{preB_1}$ model is identical to $UCON_{preB_0}$ except it adds following pre-update processes:

—$preUpdate(ATT(s)), preUpdate(ATT(o))$: an optional procedure to change certain attributes as a consequence of pre-obligations.

Definition 10 The $UCON_{preB_3}$ model is identical to $UCON_{preB_0}$ except it adds following post-update processes:

—$postUpdate(ATT(s)), postUpdate(ATT(o))$: an optional procedure to change certain attributes as a consequence of pre-obligations.

Decisions on what kind of obligations are required for requests ($getPreOBL$) are rather complicated and have various patterns. Subject or object attributes may or may not be used for the decisions. If no attribute is used, a user is likely to be required to fulfill same obligations at each request. For example, without a subject attribute, we cannot recognize previous users who have previously provided name and email address. We therefore have to ask for same obligations again. Selection of required obligation elements can be based on subject attributes only, object attributes only, both subject and object attributes, rights only, or all three of subject attributes, object attributes and rights. Suppose a subject has to read a license agreement and click 'OK' button before he exercises any rights on Web Services, or suppose a subject has to provide his personal information (age, gender, organization, e-mail address, etc) to download a company's white paper. The first obligation case can be decided based on target objects regardless of subject attributes or rights. The second one can be based on object attributes and rights. In Web Services example, suppose members have to report a monthly usage history, and guests have to provide their names and e-mail addresses. These obligations are decided based on subject attributes. Again with the Web Service example, suppose guests who only want to surf (read) contents have to provide name and e-mail address, and who want to participate and write some messages on discussion board have to provide their unique ID number such as a Social Security Number for accountability. These obligations are determined based on rights.

Obligation elements consist of OBS, OBO, and OB. The entity (*obs*) who has to perform obligation may or may not be the same subject as the requester (s). For example, to be a member of a Web community, children may need parents' agreements. In this case, the parents (*obs*) are different entities from the child (s) who wants to be a member. The entity (*obo*) on which the obligation has to be performed can be either a constant or a function of the subject attributes, object attributes and/or rights. Suppose whoever wants to access a digital library has to provide name and address. Here, name and address (*obo*) is a constant regardless of subject attributes, object attributes and rights. If a non-member has to provide something different from what a member has to provide, we can say *obo* is different for different subject attributes. Similarly, what has to be performed (*ob*) can be either a constant or a function of *obo*. For example, suppose Alice has to read and agree a license agreement for certain services by clicking 'yes' button. Here, the *ob* is always the 'click' action because the *obo* is the license agreement.

The ABC model does not specify these details. Rather, we use an abstract function called '$getPreOBL$' to obtain required obligations leaving the detail and internal decision functions as an implementation decision. After all, it can be said that obligations are decided based on requests that consist of s, o, and r. Some examples are given below.

Example 8 A license agreement obligation, every time (without attribute), $UCON_{preB_0}$:

$OBS = S$
$OBO = \{license\_agreement\}$

$OB = \{agree\}$
$getPreOBL(s, o, r) = \{(s, license\_agreement, agree)\}$

$allowed(s, o, r) \Rightarrow preFulfilled(s, license\_agreement, agree)$

Example 9 High or low license agreements for high/low objects (with object attributes), $UCON_{preB_0}$:

$OBS = S$
$OBO = \{high\_license\_agreement, low\_license\_agreement\}$
$OB = \{agree\}$
$level : O \rightarrow \{high, low\}$
$ATT(o) = \{level\}$
$getPreOBL(s, o, r) = \begin{cases} (s, high\_license\_agreement, agree), & \text{if } level(o) = \text{'high'}; \\ (s, low\_license\_agreement, agree), & \text{if } level(o) = \text{'low'}. \end{cases}$
$allowed(s, o, r) \Rightarrow preFulfilled(getPreOBL(s, o, r))$

Example 10 Selective license agreements for first time users only, $UCON_{preB_1}$:

$OBS = S$
$OBO = \{license\_agreement\}$
$OB = \{agree\}$
$registered : S \rightarrow \{yes, no\}$
$ATT(s) = \{registered\}$
$getPreOBL(s, o, r) = \begin{cases} (s, license\_agreement, agree), & \text{if } registered(s) = \text{'no'}; \\ \phi, & \text{if } registered(s) = \text{'yes'}. \end{cases}$
$allowed(s, o, r) \Rightarrow preFulfilled(getPreOBL(s, o, r))$
$preUpdate(registered(s)) : registered(s) = yes$

In Example 8, a license agreement is required from every request regardless of the user's previous agreements. This example does not require any subject and object attributes. Suppose a Web service requires a license agreement. In this case, *obs* is same as *s* and *obo* and *ob* are constants. We can extend this case to require different obligations for different object attributes or different subject attributes. Example 9 shows a case that requires two different license agreements for high and low objects. Note that these attributes are used for obtaining required obligations not for making usage decisions. Example 10 requires a license agreement only once. To do this, a subject attribute called '*registered*' is used. Once a user has agreed on a license agreement, the user is registered (*preUpdate*) for future requests. Still, this attribute is not directly used for authorizations. To be an authorization model, this attribute has to influence usage decision results. No $UCON_{preB_3}$ example is shown in this paper. However we can easily modify Example 10 to include post updates. If a user has to agree on a licence agreement at every 5 hours of accumulated usage, the total usage time has to be updated at the end of each usage for future requests.

4.4 $UCON_{onB}$ - ongoing-oBligations Models

$UCON_{onB}$ models are similar to $UCON_{preB}$ models except that obligations have to be fulfilled while rights are exercised. Ongoing-obligations may have to be fulfilled periodically or continuously. For this, we introduce a time parameter T as part of obligation elements $onOBL$. Here, T is likely to define certain time intervals that are either time-based or event-based. For example, a user may have to click an advertisement within every 30 minute interval or within every 20 Web pages accessed, or a user may have to keep an advertisement window active all the time. Note that our concern is about when users have to fulfill obligations, not when a system actually checks the fulfillments. The model assumes that $on\mathcal{B}$ has to be true all the time though actual obligation verification intervals can vary. In $UCON_{onB}$ models, there are four detailed models based on mutability issues. $UCON_{onB_0}$ includes ongoing-obligations predicate instead of pre-obligations predicate. $UCON_{onB_1}$, $UCON_{onB_2}$ and $UCON_{onB_3}$ are same as $UCON_{onB_0}$ except that they add pre-updates, ongoing-updates and post-updates, respectively.

The following definitions formalize the $UCON_{onB}$ models.

Definition 11 The $UCON_{onB_0}$ model has the following components:

—$S, O, R, ATT(S), ATT(O), OBS, OBO$, and OB are not changed from $UCON_{preB}$;

—T, a set of time or event elements;

—$on\mathcal{B}$ and $onOBL$, (ongoing-obligations predicates and ongoing-obligation elements, respectively);

—$onOBL \subseteq OBS \times OBO \times OB \times T$;

—$getOnOBL : S \times O \times R \rightarrow 2^{onOBL}$, a function to select ongoing-obligations for a requested usage;

—$onFulfilled : OBS \times OBO \times OB \times T \rightarrow \{true, false\}$;

—$on\mathcal{B}(s, o, r) = \bigwedge_{(obs_i, obo_i, ob_i, t_i) \in getOnOBL(s,o,r)} onFulfilled(obs_i, obo_i, ob_i, t_i)$;

—$on\mathcal{B}(s, o, r) = true$ by definition if $getOnOBL(s, o, r) = \phi$;

—$allowed(s, o, r) \Rightarrow true$;

—$stopped(s, o, r) \Leftarrow \neg on\mathcal{B}(s, o, r)$.

Definition 12 The $UCON_{onB_1}$ model is identical to $UCON_{onB_0}$ except it adds following pre-update processes:

—$preUpdate(ATT(s)), preUpdate(ATT(o))$: an optional procedure to change certain attributes as a consequence of pre-obligations.

Definition 13 The $UCON_{onB_2}$ model is identical to $UCON_{onB_0}$ except it adds following ongoing-update processes:

—$onUpdate(ATT(s)), onUpdate(ATT(o))$: an optional procedure to change certain attributes as a consequence of pre-obligations.

Definition 14 The $UCON_{onB_3}$ model is identical to $UCON_{onB_0}$ except it adds following post-update processes:

—$postUpdate(ATT(s)), postUpdate(ATT(o))$: an optional procedure to change certain attributes as a consequence of pre-obligations.

The following Example 11 shows a simple example for $UCON_{onB_0}$.

Example 11 Watch advertisement windows while s exercise r, $UCON_{onB_0}$:

$OBS = S$
$OBO = \{ad\_window\}$
$OB = \{keep\_active\}$
$T = \{always\}$
$getOnOBL(s, o, r) = \{(s, ad\_window, keep\_active, always)\}$

$allowed(s, o, r) \Rightarrow true$
$stopped(s, o, r) \Leftarrow \neg onFulfilled(s, ad\_window, keep\_active, always)$

Here, there is only one ongoing obligation is required. Suppose a free Internet service provider requires users to watch an advertisement while they are connected to the server. In this case, there is no requirement that has to be completed before using the service. As long as the advertisement window is active, the usage is allowed. Again how frequently the system checks the status of the advertisement window is not considered. Although examples for $UCON_{onB_1}$, $UCON_{onB_2}$ and $UCON_{onB_3}$ are not shown here, they can be developed without much effort. For example, consider free Internet Services. Suppose every user has to watch ad for first 10 minutes of connection to Internet, but every 10th user has to watch ad for 20 minutes. Here every time a user connect Internet, usage number has to be increased or reset to 0 at the beginning of access to decide which ongoing obligation has to be fulfilled. This is an example of $UCON_{onB_1}$. Suppose a user has to click an advertisement within every 30 minutes. Here a last click time has to be updated throughout usage. This is an example of $UCON_{onB_2}$. Further suppose a user has to watch advertisement after first 10 hours every month. Here, current connection time has to be accumulated at the end of each connection. This is an example of $UCON_{onB_3}$.

4.5 $UCON_{preC}$ - pre-Conditions Model

As described earlier, conditions define certain environmental restrictions that have to be satisfied for usages. In general, *preCON* includes certain environmental restrictions that are not directly related to subjects and objects. Current environmental or system-oriented status is retrieved each time a condition is evaluated. By utilizing conditions in usage decision process, $UCON_C$ can provide finer-grained controls on usages. Unlike authorization and obligation models, condition models cannot be mutable. Note that this is different from the fact that the value of conditional status can be changed as the environmental situation is being changed (e.g., current time is changed as time goes, or a wireless access point is changed as a user moves around a building). Although subject or object attributes are not

used for usage decision process, they can be used to decide what kind of condition elements ($preCON$) have to be enforced for usage decision. $UCON_{preC}$ introduces pre-conditions predicate ($pre\mathcal{C}$) that has to be evaluated before requested rights are exercised.

The following definitions formalize the $UCON_{preC}$ model.

Definition 15 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);
—$getPreCON : S \times O \times R \rightarrow 2^{preCON}$;
—$preConChecked : preCON \rightarrow \{true, false\}$;
—$pre\mathcal{C}(s, o, r) = \bigwedge_{preCon_i \in getPreCON(s,o,r)} preConChecked(preCon_i)$
—$allowed(s, o, r) \Rightarrow pre\mathcal{C}(s, o, r)$.

In $UCON_{preC_0}$, $pre\mathcal{C}$ is utilized in usage decision process along with S, O and R. A set of relevant condition elements $preCON$ is selected based on a request possibly using subject or object attributes. To allow a request, all of the selected condition restrictions have to be evaluated ($pre\mathcal{C}$). For example, suppose there are requirements to restrict locations where usages can be exercised. Checking a CPU-id or an IP address before a usage allowance is an example of $UCON_{preC_0}$. Example 12 checks the current location of a user at the time of a request. Allowed locations for student and faculty can be different and have to be selected accordingly. This example assumes either there is no location change while the request is exercised or there is no restriction of location changes during the usages once the original location has been approved.

Example 12 Location limitation, $UCON_{preC_0}$:

$studentAREA, facultyAREA$ (allowed area codes for student and faculty)
$curArea$ is a current rendering device's area code
$ATT(s) = \{member\}$
$preCON = \{(curArea \in studentAREA), (curArea \in facultyAREA)\}$

$$getPreCON(s, o, r) = \begin{cases} (curArea \in studentAREA), & \text{if } member(s) = \text{'student'}; \\ (curArea \in facultyAREA), & \text{if } member(s) = \text{'faculty'}. \end{cases}$$

$allowed(s, o, r) \Rightarrow preConChecked(getPreCON(s, o, r))$

4.6 $UCON_{onC}$ - ongoing-Conditions Model

In many cases, environmental restrictions have to be satisfied while rights are in active use. This could be supported within $UCON_{onC}$ model. In $UCON_{onC}$, usages are allowed without any decision process at the time of requests. However, there is an ongoing conditions predicate to check certain environmental status repeatedly throughout the usages. As mentioned earlier, the $UCON_{onC_0}$ model is intrinsically immutable.

The following definitions formalize the $UCON_{onC}$ model.

Definition 16 The $UCON_{onC_0}$ model has the following components:

—$S, O, R, ATT(S)$, and $ATT(O)$ are not changed from $UCON_{preA}$;
—$onCON$ (a set of ongoing-conditions elements);
—$getOnCON : S \times O \times R \rightarrow 2^{onCON}$;
—$onConChecked : onCON \rightarrow \{true, false\}$;
—$on\mathcal{C}(s, o, r) = \bigwedge_{onCon_i \in getOnCON(s,o,r)} onConChecked(onCon_i)$
—$allowed(s, o, r) \Rightarrow true$;
—$stopped(s, o, r) \Leftarrow \neg on\mathcal{C}(s, o, r)$.

The $UCON_{onC_0}$ model introduces an ongoing conditions predicate ($on\mathcal{C}$) for monitoring selected condition elements ($getOnCON(s, o, r)$). If any current environmental status violates any of the restrictions, the allowed right is revoked and the exercise is stopped. In Example 13 below, allowed time period limitation is required throughout usage exercises. For example, suppose a day-time user can access objects during day time (say 8am to 4pm), and a night-shift user can access objects during night time (say 4pm to 12pm). Note that, $currentT$ is a current status (time) of local time, not an attribute of subject or object. Here, $currentT$ is evaluated throughout the usage and if its value is no longer within the allowed period, the usage is stopped. This example is likely to use both pre-condition and ongoing-condition since current time is also likely to be checked at the time of request, hence a combined model $UCON_{preC_0onC_0}$.

Example 13 Time limitation, $UCON_{preC_0onC_0}$:

$dayH, nightH$ (day-shift and night-shift office hours, mutually exclusive)
$currentT$ is current time
$preCON : \{(currentT \in dayH), (currentT \in nightH)\}$

$onCON : \{(currentT \in dayH), (currentT \in nightH)\}$

$$getPreCON(s, o, r) = \begin{cases} (currentT \in dayH), & \text{if } shift(s) = \text{'day'}; \\ (currentT \in nightH), & \text{if } shift(s) = \text{'night'}. \end{cases}$$

$$getOnCON(s, o, r) = \begin{cases} (currentT \in dayH), & \text{if } shift(s) = \text{'day'}; \\ (currentT \in nightH), & \text{if } shift(s) = \text{'night'}. \end{cases}$$

$allowed(s, o, r) \Rightarrow preConChecked(getPreCON(s, o, r))$
$stopped(s, o, r) \Leftarrow \neg onConChecked(getOnCON(s, o, r))$

4.7 Global Obligations

In ABC model, obligations are used for usage decision of current request. However, there are other cases where the evaluation of obligations' fulfillment is postponed for some time and used for future usage decision rather than usage decision of current request. Suppose a member has to pay monthly metered payment for continuous music services. Though total usage time has to be updated at each service, this does not influence usage decision of service requests until the payment due.

This kind of obligation is called global obligation and does not fit into the ABC model structure though it is also an important aspect in usage control. UCON considers these global obligations as an exceptional case of UCON obligation models. Global obligations are unique in their characteristics. Like other obligations, they have to be fulfilled by obligation subjects. However, they do not affect any usage decisions on the originated requests. Rather, what they can do is to influence future requests. Although certain updates can be required for global obligations, these updates do not influence any decision for current usages. Since there is no influence on decision-making for current usages, this feature can't belong to $UCON_{B_0}$, hence can't belong to $UCON_{B_1}$,$UCON_{B_2}$, or $UCON_{B_3}$. Note that the actual updates still can happen either before, during, or after the usages. In case of an update after usage, unlike post-update, it does not have to be done right after the end of usage. For example, a user may have to fulfill usage log report at the end of each usage, each day, or each month. Monthly metered payment, or monthly subscription payment are also examples of global obligations. The actual influence on decision making is postponed to certain point. These global obligations are independent from the originated usage transactions and have impact only on future decisions.

5. APPLICATIONS IN ABC MODEL

Usage control encompasses traditional access controls, trust management, and digital rights management and goes further in its definition and the scope. In this section, we show how MAC, DAC, RBAC, trust management, and DRM can be realized within the ABC model. We further discuss some possible extensions of these policies for better understanding of their characteristics and richer controls. Most traditional access controls and trust management can be realized by using $UCON_{preA_0}$ model. Some extensions may require $UCON_{preA_1}$ model. Ongoing decisions are rarely discussed in literature. Mutability issues are not common in previous work. Most of the research that deals with continuity or mutability issues still lack systematic perspective and comprehensiveness, being narrowly focused. Discussions on some of this prior work in terms of the ABC model provides solid evidence of comprehensiveness and richness of our models. We also show some healthcare system examples that require obligations, conditions as well as authorizations.

5.1 Mandatory Access Control

In mandatory access control, security labels are used for usage decisions. In UCON point of view, clearance is a subject attribute and classification is an object attribute. These security labels of subjects and objects are compared to enforce simple security property (no read up) and star property (no write down). Example 1 in previous section shows this MAC policy using the $UCON_{preA_0}$ model.

Traditional access controls have rarely supported an update property. In MAC, strong tranquility property which belongs to our immutable authorizations is normally assumed. In other words, security labels of subjects and objects cannot be changed by users' actions. Only administrative actions can change the labels. With a weak tranquility property, security labels can be changed by users' autonomous actions but only without violating defined security policies. This has been known as a high watermark property. Example 14 shows this high watermark property of

BLP as an example of $UCON_{preA_1}$. Here, a subject always start with the lowest possible clearance label. The clearance of the subject is raised to higher labels until it reaches its maximum label as the subject accesses higher objects. LUB denotes least upper bound.

Example 14 MAC policies with high watermark property, $UCON_{preA_1}$:

L is a lattice of security labels with dominance relation $\geq$
$clearance : S \rightarrow L$
$maxClearance : S \rightarrow L$
$classification : O \rightarrow L$
$ATT(S) = \{clearance, maxClearance\}$
$ATT(O) = \{classification\}$

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

5.2 Role-based Access Control

The $UCON_{preA_0}$ model also can support RBAC in its authorization process. In RBAC96 model [Sandhu et al. 1996], a role is a collection of users and a collection of permissions. The permission is a collection of object-right pairs. In $UCON_{preA_0}$, user-role assignment can be viewed as subject attributes and permission-role assignment as attributes of object and rights. Example 15 shows how RBAC_1 can be viewed in our ABC models. RBAC_1 includes hierarchy in its definition. Only activated roles are used for authorization decision. Here, if there exists an active role ($actRole(s)$) that dominates any of the permission roles ($Prole(o, r)$), a request is allowed. Example 16 and 17 shows examples that include high watermark property. Example 16 does not have a role hierarchy while Example 17 does. In both cases, active roles are updated if other than currently active roles are required for a request. Since the role hierarchy is not a lattice (so LUB is not always defined), Example 17 also has a selection issue. Having an automated high watermark property in RBAC eliminates the least privilege principle that is supported in original models.[7] It also causes a selection problem in case there are multiple roles available for a request. In Example 17, 'UB' denotes upper bounds.

Example 15 RBAC_1 with activation, $UCON_{preA_0}$:

$P = \{(o, r)\}$
$ROLE$ is a partially ordered set of roles with dominance relation $\geq$
$actRole : S \rightarrow 2^{ROLE}$
$Prole : P \rightarrow 2^{ROLE}$
$ATT(S) = \{actRole\}$
$ATT(O) = \{Prole\}$

[7] Least Privilege can be supported if role activation is done manually by users.

$allowed(s, o, r) \Rightarrow \exists role \in actRole(s), \exists role' \in Prole(o, r), role \geq role'$

Example 16 RBAC$_0$ with high watermark property, $UCON_{preA_1}$:

$P = \{(o, r)\}$
$ROLE$ is an unordered set of roles
$srole : S \rightarrow 2^{ROLE}$
$prole : P \rightarrow 2^{ROLE}$
$actRole : S \rightarrow 2^{ROLE}$
$ATT(S) = \{srole, actRole\}$
$ATT(O) = \{prole\}$

$allowed(s, o, r) \Rightarrow srole(s) \cap prole(o, r) \neq \phi$
$preUpdate(actRole(s)) : actRole(s) =$
$\begin{cases} actRole(s), & \text{if } actRole(s) \cap Prole(o, r) \neq \phi; \\ actRole(s) \cup \lambda(srole(s) \cap prole(o, r)), & \text{if } actRole(s) \cap Prole(o, r) = \phi, \end{cases}$
where λ is a nondeterministic selection function of an element from a set.

Example 17 RBAC$_1$ with high watermark property, $UCON_{preA_1}$:

$P = \{(o, r)\}$
$ROLE$ is a partially ordered set of roles with dominance relation $\geq$
$srole : S \rightarrow 2^{ROLE}$
$prole : P \rightarrow 2^{ROLE}$
$actRole : S \rightarrow 2^{ROLE}$
$ATT(S) = \{srole, actRole\}$
$ATT(O) = \{prole\}$
$\overline{ROLE} = \{role \mid \exists role' \in srole(s), \exists role'' \in prole(o, r), role' \geq role \geq role''\}$
$\widetilde{ROLE} = \{role \mid \exists role' \in actRole(s), \exists role'' \in prole(o, r), role' \geq role \geq role''\}$
$\widehat{ROLE} = \{role \mid \exists role' \in actRole(s), \exists role'' \in prole(o, r), role \in \overline{ROLE},$
$role \in UB(role', role'')\}$

$allowed(s, o, r) \Rightarrow \overline{ROLE} \neq \phi$
$preUpdate(actRole(s)) :$
$actRole(s) = \begin{cases} actRole(s), & \text{if } \widetilde{ROLE} \neq \phi; \\ actRole(s) \cup \lambda(\widehat{ROLE}), & \text{if } \widetilde{ROLE} = \phi, \end{cases}$
where λ is a nondeterministic selection function of an element from a set.

5.3 Discretionary Access Control

Discretionary access control also can be supported in $UCON_{preA}$. DAC policies govern the access of users to an object based on individual or group identities of users and objects. The access modes such as read, write, or execute are granted to a user if the user has privilege to use a specific access mode on an object. Traditionally access matrix has been realized by using either access control list (ACL), or capability list. In $UCON_{preA}$, DAC can be expressed by using either ACLs or capability lists as shown below. In many examples, ACLs and capability lists can

be used to achieve same control functions. However in certain cases such as group-based usage number restrictions, ACLs and capability lists have different control functionalities. This distinction has been rarely discussed in previous literature.

Following Examples 18 and 19 utilize ACL and capability list respectively. In Example 18, if any of the subject's group names is listed in a requested object's ACL, the request is allowed. On the other hand, in Example 19, capability list is used to check whether a subject holds any dominant group ID for the requested right. Although these two cases seem to accomplish similar functionality, the functional distinction is much clearer when an update procedure is required. Suppose each group of a subject has a limited number of usage times. In this case, the available number of a subject's usage has to be reduced. To do this we have to select one (or some) of the subject's group(s) and update the current usage number(s) of the selected subject group(s). Here, the number of allowed usage count is assigned to each group of subjects so that the usage can be controlled on a subject group basis. On the other hand, if capability lists are used, the allowed usage count number is assigned to each group of objects and the update of the number is controlled on an object group basis.

Example 18 utilizes ACL and allows multiple group IDs of a subject (assuming no group hierarchy). If one of the subject's group IDs exists in ACL(o), the request is allowed. Example 19 utilizes capability lists and includes a hierarchy of object group IDs. Here, if there exists a object group IDs that is lower than or equal to a group ID of $CL(s)$, the request is allowed. Example 20 utilizes capability lists and includes usage count k to limit the number of usages for each object group. Here, for the sake of simplicity, we assume there is only one k for each (g, r). In this example, since an object can have multiple group IDs, an update requires a selection of one group ID among the object's group IDs.

Example 18 DAC using ACL w/ multiple group ID, $UCON_{preA_0}$:

G is a set of groups of subject s
$groupId : S \rightarrow 2^G$, many to many mapping
$ACL : O \rightarrow 2^{G \times R}$, g is authorized to do r to o
$ATT(S) : \{groupId\}$
$ATT(O) : \{ACL\}$

$allowed(s, o, r) \Rightarrow \{(g, r) \mid g \in groupId(s)\} \cap ACL(o) \neq \phi$

Example 19 DAC using Capability List w/ group hierarchy, $UCON_{preA_0}$:

G is a partially ordered set of groups of o
$groupId : O \rightarrow 2^G$
$CL : S \rightarrow 2^{G \times R}$, s is authorized to do r to g or lower g's
$ATT(S) = \{CL\}$
$ATT(O) = \{groupId\}$

$allowed(s, o, r) \Rightarrow \exists g \in groupId(o), \exists (g', r) \in CL(s), g \leq g'$

Example 20 DAC using Capability List w/ multiple group IDs and usage count, $UCON_{preA_1}$:

G is a set of group names
$groupId : O \rightarrow 2^G$, many to many mapping
$CL : S \rightarrow 2^{G \times R \times K}$, s is authorized to do r to g for k times
$ATT(S) = \{CL\}$
$ATT(O) = \{groupId\}$

$allowed(s, o, r) \Rightarrow \overline{GR} \neq \phi$,
$\overline{GR} = \{(g, r) \mid g \in groupId(o)\} \cap \{(g', r) \mid (g', r, k) \in CL(s), k \geq 1\}$

$\lambda : \overline{GR} \rightarrow G$, a functional mapping to select a group for update
$preUpdate(CL(s)) : k = k - 1, (\lambda(\overline{GR}), r, k) \in CL(s)$

5.4 Trust Management

Trust management has mainly focused on authorization decisions of previously known users. Most of the related research has discussed architectural and mechanistic aspects of authorizations. Although the ABC model also covers authorizations of strangers, our focus is not how to get a credential to authorize a stranger's usage request. Rather, we focus on usage decision policies and models. In Example 21, a doctor can be mapped to multiple specialities. If a requester holds any speciality, then he can read the object. However if a requester want to write on an object, one of his specialities should be same as the object's speciality.

Example 21 Hospital information usages of doctor by specialty, $UCON_{preA_0}$:

$SPECIALITY$ is a set of medical specialty names
$cert : S \rightarrow 2^{SPECIALITY}$
$groupID : O \rightarrow SPECIALITY$
$ATT(s) : \{cert\}$
$ATT(o) : \{groupID\}$

$allowed(s, o, read) \Rightarrow (cert(s) \neq \phi)$
$allowed(s, o, write) \Rightarrow (cert(s) \neq \phi) \wedge groupID(o) \in cert(s)$

5.5 Digital Rights Management

Usage decisions in commercial DRM solutions usually utilize user-defined, application-level, payment-based security policies, and do not include traditional access control policies. Typical examples include pay-per-view, metered payment, membership-based monthly subscriptions, etc. These DRM examples can be realized within our ABC model. In Example 22, a usage request is allowed if a subject holds enough pre-paid credits to use certain rights on specific objects. In this case, the credit is considered as a subject attribute and the value of the requested usage as an at-

tribute of the object and right.

Example 22 DRM pay-per-use, $UCON_{preA_1}$:

M is a set of money amount
$credit : S \rightarrow M$
$value : O \times R \rightarrow M$
$ATT(s) : \{credit\}$
$ATT(o, r) : \{value\}$

$allowed(s, o, r) \Rightarrow credit(s) \geq value(o, r)$
$preUpdate(credit(s)) : credit(s) = credit(s) - value(o, r)$

In general, payment based authorization requires certain update procedures to resolve usage expenses. In Example 22, a user's credit has been reduced by the value of usages at the time of a request approval. In case of metered payment, post-updates are likely to be required. In Example 23, since a usage on an object holds more than one value, a system has to select a value for update. The system may have to select a value based on subject's membership ranks, sale period, or multiple purchases. Because the selection policies can vary, Example 23 simply utilizes a non-deterministic selection function to describe this functionality.

Example 23 DRM pay-per-use, one credit, multiple values, $UCON_{preA_1}$:

M is an ordered set of money amount
$credit : S \rightarrow M$
$value : O \times R \rightarrow 2^M$
$ATT(s) : \{credit\}$
$ATT(o, r) : \{value\}$
$\overline{M} = \{m \mid m \in value(o, r), m \leq credit(s)\}$, a set of available values for selection.

$allowed(s, o, r) \Rightarrow \exists m \in value(o, r), m \leq credit(s)$
$preUpdate(credit(s)) : credit(s) = credit(s) - \lambda(\overline{M})$,
where λ is a selection function to select a value for update.

In case a user holds more than one credit account, if the sum of credits is more than a value, the request is allowed. Here, some or all of the credit accounts have to be reduced in total by the usage value (Example 24).

Example 24 DRM pay-per-use, multiple credits, one value, $UCON_{preA_1}$:

M is an ordered set of money amount
$credit : S \rightarrow 2^M$
$value : O \times R \rightarrow M$
$ATT(s) : \{credit\}$
$ATT(o, r) : \{value\}$

$\overline{M} = \{\overline{m} \mid \overline{m} \in credit(s)\}$
$\widehat{M} = \{\widehat{m} \mid \sum \widehat{m} = value(o)\}$
$\lambda : \overline{M} \rightarrow \widehat{M}, \overline{m} \geq \widehat{m}$

$allowed(s, o, r) \Rightarrow \sum credit(s) \geq value(o, r)$
$preUpdate(credit(s)) : \forall \overline{m}, \overline{m} = \overline{m} - \lambda(\overline{m})$

Many DRM solutions and studies have included some form of obligations and conditions because of DRM's distributed and payment-based nature. Some objects can only be used at certain locations or time durations. A user may have to provide certain personal information or usage log information for further use. Some DRM related examples for obligations and conditions are presented in previous sections.

5.6 Modern Access Control (Healthcare Examples)

Generally, modern access control requires more than authorizations for usage decision. In this section, we show several healthcare information system examples that require authorizations, obligations and conditions. We also show how one example can be expressed in various ways using different decision predicates. Example 25 shows an example that requires an authorization for usage decision. Here, the number of doctor's previous operations is considered as a subject's attribute and used for authorization.

Example 25 A medical doctor (s) can perform (r) an operation (o) only if he has performed operations more than 3 times, $UCON_{preA_1}$:

$ROLE$ is an unordered set of roles
$SPECIALITY$ is a set of medical speciality names
N is a set of subject's total operation numbers
$exp : S \rightarrow N$
$sRole : S \rightarrow 2^{ROLE}$
$sArea : S \rightarrow 2^{SPECIALITY}$
$oArea : O \rightarrow 2^{SPECIALITY}$
$ATT(s) : \{sRole, sArea, exp\}$
$ATT(o) : \{oArea\}$

$allowed(s, o, operate) \Rightarrow$ 'doc' $\in sRole(s), sArea(s) \cap oArea(o) \neq \phi, exp(s) \geq 3$
$preUpdate(exp(s)) : exp(s) = exp(s) + 1$

In Example 26, suppose medical operations can be allowed only when patients agree on a consent form. This requires additional obligation predicate. *obs* is selected based on an object' attribute, *opid*. In this example, obligation actions (*ob*) have to be performed by patient of the requested operation. Authorization is also used for medical speciality verification.

Example 26 A medical doctor can perform an operation only if patients agree on consent form, $UCON_{preA_0 preB_0}$:

$ROLE$ is an unordered set of roles
$SPECIALITY$ is a set of medical speciality names
$PATIENTid$ is a set of patients' identification numbers
$sRole : S \rightarrow 2^{ROLE}$
$sArea : S \rightarrow 2^{SPECIALITY}$
$oArea : O \rightarrow 2^{SPECIALITY}$
$spid : S \rightarrow PATIENTid$
$opid : O \rightarrow PATIENTid$
$ATT(s) : \{sArea, spid\}$
$ATT(o) : \{oArea, opid\}$
$OBS = \{s' | \text{`PATIENT'} \in sRole(s')\}$
$OBO = \{consent\}$
$OB = \{agree\}$
$getPreOBL(s, o, operate)$
$= \{(s', consent, agree)\}$ where $s' \in OBS, spid(s') = opid(o)$

$allowed(s, o, operate) \Rightarrow$ 'doctor' $\in sRole(s), sArea(s) \cap oArea(o) \neq \phi$
$allowed(s, o, operate) \Rightarrow preFulfilled(getPreOBL(s, o, operate))$

Suppose there are junior doctors and senior doctors ($sRole(o)$). In case a junior doctor wants to perform operations, the operation is allowed only with the presence of any of his senior doctors. This can be realized by using either authorization, obligation, or condition predicates. Example 27 utilizes condition predicate that checks current local time and decides whether any of the senior doctors is on-duty at the time of operation request. Authorization is used together with condition predicate to check doctor's speciality. In Example 28, obligation predicate is used to check whether any of senior doctor has agreed to be available. Alternatively, as shown in Example 29, same example can be also realized by using authorization predicate. Here, senior doctor's presence is treated as an attribute of the subject. In Example 27, 28, and 29, 'sDoc' and 'jDoc' are labels for senior doctor and junior doctor, respectively.

Example 27 A junior medical doctor can perform an operation only with the presence of a senior doctor, $UCON_{preA_0 preC_0}$:

$ROLE$ is an unordered set of roles
$SPECIALITY$ is a set of medical speciality names
$DOCid$ is a set of doctors' identification numbers
$curT$ is a current local time, T is a set of time
$sRole : S \rightarrow 2^{ROLE}$
$sArea : S \rightarrow 2^{SPECIALITY}$
$dId : S \rightarrow DOCid$, a functional mapping of subject to a doctor's ID number
$sdId : S \rightarrow 2^{DOCid}$, a functional mapping of subject to a set of senior doctors
$dutyS : S \rightarrow T$, a functional mapping of subject to duty start time
$dutyE : S \rightarrow T$, a functional mapping of subject to duty end time

$oArea : O \rightarrow 2^{SPECIALITY}$
$ATT(s) : \{sRole, sArea, dId, sdId, dutyS, dutyE\}$
$ATT(o) : \{oArea\}$
$getPreCON(s, o, operate)$
$= \begin{cases} (\exists s', dId(s') \in sdId(s), dutyS(s') \leq curT \leq dutyE(s')), & \text{if } sRole(s) = \text{'jDoc'}; \\ \phi, & \text{if } sRole(s) = \text{'sDoc'}. \end{cases}$
$allowed(s, o, operate) \Rightarrow$ 'doctor' $\in sRole(s), sArea(s) \cap oArea(o) \neq \phi$
$allowed(s, o, operate) \Rightarrow preCondChecked(getPreCON(s, o, operate))$

Example 28 A junior medical doctor can perform an operation only with the presence of a senior doctor, $UCON_{preA_0preB_0}$:

$ROLE$ is an unordered set of roles
$SPECIALITY$ is a set of medical speciality names
$DOCid$ is a set of doctors' identification numbers
$sRole : S \rightarrow 2^{ROLE}$
$sArea : S \rightarrow 2^{SPECIALITY}$
$dId : S \rightarrow DOCid$, a functional mapping of subject to a doctor's ID number
$sdId : S \rightarrow 2^{DOCid}$, a functional mapping of subject to a set of senior doctors
$oArea : O \rightarrow 2^{SPECIALITY}$
$ATT(s) : \{sRole, sArea, dId, sdId\}$
$ATT(o) : \{oArea\}$
$OBS = \{s' | \text{'sDoc'} \in sRole(s')\}$
$OBO = \{presence\}$
$OB = \{agree\}$
$getPreOBL(s, o, operate)$
$= \begin{cases} ((s', presence, agree) \text{ where } s' \in OBS, dId(s') \in sdId(s)), & \text{if } sRole(s) = \text{'jDoc'}; \\ \phi, & \text{if } sRole(s) = \text{'sDoc'}. \end{cases}$
$allowed(s, o, operate) \Rightarrow$ 'doctor' $\in sRole(s), sArea(s) \cap oArea(o) \neq \phi$
$allowed(s, o, operate) \Rightarrow preFulfilled(getPreOBL(s, o, operate))$

Example 29 A junior medical doctor can perform an operation only with the presence of a senior doctor, $UCON_{preA_0}$:

$ROLE$ is an unordered set of roles
$SPECIALITY$ is a set of medical speciality names
$DOCid$ is a set of doctors' identification numbers
$sRole : S \rightarrow 2^{ROLE}$
$sArea : S \rightarrow 2^{SPECIALITY}$
$sdId : S \rightarrow 2^{DOCid}$, a mapping of subject to a set of **on-duty** senior doctors.
$oArea : O \rightarrow 2^{SPECIALITY}$
$ATT(s) : \{sRole, sArea, sdId\}$
$ATT(o) : \{oArea\}$
$allowed(s, o, operate) \Rightarrow$ 'jDoc' $\in sRole(s), sArea(s) \cap oArea(o) \neq \phi, sdId(s) \neq \phi$

ABC model is comprehensive enough to include various access control policies in a single framework. The goal of ABC family model is not to make an air-tight distinction among the detailed models. In ABC model, policies or requirements can be resolved in multiple ways. Although we have shown many examples, how one actually implements access (or usage) control policies and requirements is not the issue of ABC model. These issues are to be considered at the architecture and mechanism layers, not at the model layer. ABC model provides possible ways to realize or express various policies and requirements in a formal framework. It is this richness and robustness of the expressive power that makes ABC model significant.

6. DISCUSSION

In the previous section we have demonstrated how traditional access control policies, trust management, DRM, and healthcare examples can be expressed in the ABC model. In this section we discuss mutability issues of ABC model briefly. Then we discuss administrative issues on the usage control and examine some DRM and Health-care examples in terms of UCON administration. We also discuss reversed control aspects of usage control as an extension of UCON administration.

6.1 Mutability Issues in ABC Model

Separation of Duty (SoD) is one popular issue of classical access control and has been studied extensively in previous literature. SoD can be considered as a constraint of attribute assignment. Suppose purchase_clerk can only 'prepare' a check and account_clerk can only 'issue' a check. This is a typical static SoD example. To enforce static SoD, a system requires a subject cannot be assigned to both purchase_clerk and account_clerk attributes simultaneously. Here both purchase_clerk and account_clerk are immutable attributes. In ABC model immutable attributes are assumed to be previously assigned to subjects and objects and managed by administrative actions. In other words, in ABC model, constraints that have to be applied to the assignments of immutable attributes are already enforced.

Although we can make ABC model more complex to enforce static SoD by including 'Constraints' property within the model, since assignment of immutable attribute to subject or object is administrative issue, and since ABC model is the first step in this arena, we consider only the core issues of usage control and keep the core model as simple as possible. By simplifying immutable aspects of attributes in ABC model, we can focus our discussion on mutable aspects of attributes.

Mutability is a crucial property for history-based policies (i.e., dynamic SoD, Chinese Wall) and policies that require consumable attributes (most of commercial B2C DRM policies). Unlike immutable attributes, mutable attributes are modifiable as a consequence of subjects' actions and do not require any administrative action for updates. Therefore, in ABC model, policies that require mutable attributes are enforced within the core model and not considered as an administrative issue. While most classical access control models focus on the management of immutable attributes, ABC model separates core models from administrative issues and deals with mutable aspects of attributes by assuming that immutable attributes are already taken cared.

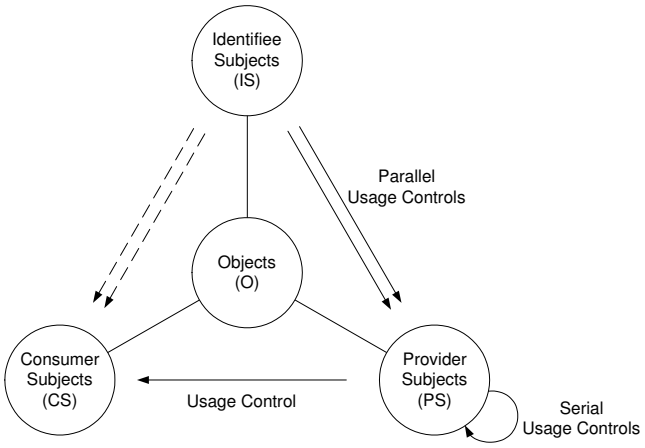

Fig. 5. Administrative UCON Triangle

6.2 Administrative Issues in UCON

In the ABC model, we have mainly focused on usages of consumer subjects on objects. In this section we assume that usage rules such as authorization, obligation, and condition rules are already provided. As mentioned earlier, in UCON, subjects can be either consumer, provider, or identifiee. Each subject party has close relationships with other parties. One party may influence other parties' usage decisions. Each party holds its own rights on objects. Exercising rights on an object may require certain obligations that have to be fulfilled before, during or after the rights are exercised. Fulfillment of these obligations may create other objects (called derivative objects) that have to be protected and controlled from usages. This series of relationships has to be resolved seamlessly in UCON as an administrative issue. Here, we discuss some fundamental issues of UCON administration briefly. We believe that the further work on administrative UCON is crucial for the success of usage control.

Figure 5 shows an administrative triangle for usage control decisions. Here, a consumer subject is an end-user who is the last beneficiary of an object content in a supply chain. If allowed by a provider, a consumer can hand over the object to another consumer and can control usage of the new consumer. In this case, the original consumer becomes a provider of the object to the new consumer. Note that this is different from that a consumer passes an object or a copy of an object to another consumer on behalf of the previous provider of the object while the previous provider controls usage of the disseminated object. Normally, a consumer's usage on an object is likely to be controlled by a single provider. Although there can be multiple providers who actually provide same object copies to a consumer, these copies are considered as separate objects and may have different control policies. If a provider is not an originator of an object, the provider's ability to control consumers's usages on the object is likely to be limited by another provider. If an object o_1 includes other objects o_2 and o_3, a provider subject s_1 of o_1 is considered

as a consumer of the included objects o_2 and o_3 and the o_1 as a separate object. In this case, s_1's ability on usages of o_1 is also limited by the providers of o_2 and o_3. In Figure 5, this chain of usage controls is denoted as 'serial usage control'. Unlike provider subjects, there exists no control chain of identifiee subjects. Identifiee subjects are subjects whose individually identifiable information is included within an object, therefore hold certain rights to control usages on the object. Credit card information or DNA information are some of the examples of individually identifiable information. The usages of an object that includes these privacy-related information of multiple subjects are controlled by the identifiee subjects. Such multiple controls on usages are denoted as 'parallel usage control'. In general, identifiee subjects are likely to limit provider's usages on the object to control consumer's usages (dotted arrows) on their privacy-related information.

As a summary, UCON has to be viewed as a comprehensive approach to protecting and controlling usages of three subject parties and their relationships and influences on each other. In today's dynamic, distributed digital environment, traditional one-way control no longer provides adequate trustworthiness. Eventually, unlike previous one-way (from provider to consumer) approaches, control decision of UCON has to be multi-directional for mutual controls and privacy protections. We believe these issues are no longer just technical matters. Business commitment and legal and social support are also crucial for the success of usage control.

6.3 DRM and Healthcare Applications in UCON Administration

By distinguishing subject parties, UCON emphasizes relationships between subjects and objects and between subjects themselves. This distinction is shown in figure 6 and 7. Figure 6 is a UCON diagram for privacy non-sensitive objects and Figure 7 is for privacy sensitive objects. The UCON model for privacy sensitive objects includes an additional subject called identifiee and relevant rights. Figure 6 and 7 are based on the following legend.

PNO: Privacy Non-sensitive Object
PSO: Privacy Sensitive Object
Cx: Consumer x
Px: Provider x
Ix: Identifiee x
yR: y Rights
yA: y Authorization
yC: y Condition
yB: y oBligation
where $x = \{x|R, A, C, B\}, y = \{y|C, P, I\}$

We will use two examples and demonstrate how UCON models can be applied for privacy non-sensitive and privacy sensitive digital information. One simple example is a popular MP3 music file distribution. This example can be explained with Figure 6 that has provider and consumer subjects sides. Suppose a music composer (say Bob) wants to sell his new song through a distributor, and a buyer (say Alice) wants to buy the song from the distributor. In case of the relations between Bob and the distributor, Bob will be a provider subject (PS) and the

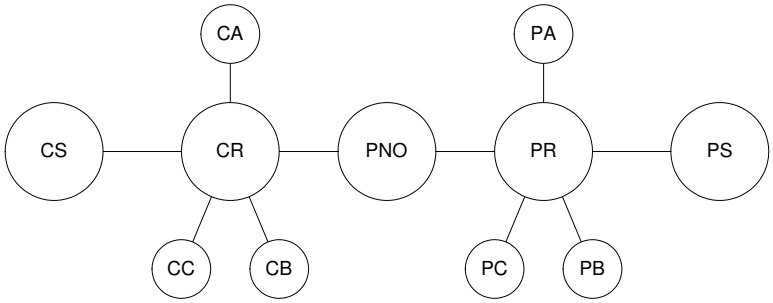

Fig. 6. Two Sides of UCON Model for Privacy Non-sensitive Objects

distributor will be a consumer subject (CS). Bob will have certain provider rights (PR) that are agreed at the time of a contract with the distributor. The distributor will have rights (CR) to distribute the MP3 song (PNO) and get certain profits from the sales. Likewise, in case of Alice and the distributor, Alice will be a consumer subject and the distributor will be a provider subject. Then Alice has rights (CR) such as play right for the song and the distributor will have rights (PR) such as copy and disseminate rights on the object. In this case, Alice may be required to pay ahead (CA) to obtain a play right but only on a specific player (CC) which is selected by her. In addition, she may have to agree on submission of her usage log report to the provider (CB). On the other hand, the distributor can have rights to collect consumers' usage log information. This shows that in UCON system, a consumer's obligation is likely to be a provider's right and vice versa.

One good example for the control of privacy sensitive objects might be a healthcare system. We consider a healthcare system called PCASSO to demonstrate the UCON model for privacy sensitive objects. The PCASSO project was developed by UC San Diego and SAIC under the support of NIH [Baker et al. 1997]. The main purpose of the project is to develop a healthcare system that provides secure access to highly sensitive patient information over Internet. Access control of PCASSO mainly utilizes labels and roles. Patient records are labeled with one of 5 security levels including Low, Standard, Deniable, Guardian Deniable, and Patient Deniable. As a provider subject, the primary care provider provides patient medical record (PSO). In addition, the primary care provider decides security level of patient medical information. Care providers (primary, emergency or others), guardians, researchers, and even patients can be consumer subjects. In PCASSO, the patient role can be either a consumer subject or an identifiee subject. As a consumer subject, a patient can read his medical record if it is not patient deniable. As an identifiee subject, the patient can review (IR) access log information on his record. Note that the patient doesn't have rights to decide use and disclosure of his medical information in PCASSO.

According to recent regulation called the Privacy Rule from the US Department of Health and Human Services (HHS), healthcare providers such as doctors and hospitals are required to obtain a patient's written consent before using or disclosing the patient's personal healthcare information to carry out treatment, payment, or healthcare operations (TPO) [HHS 2002]. To use or disclose the patient's med-

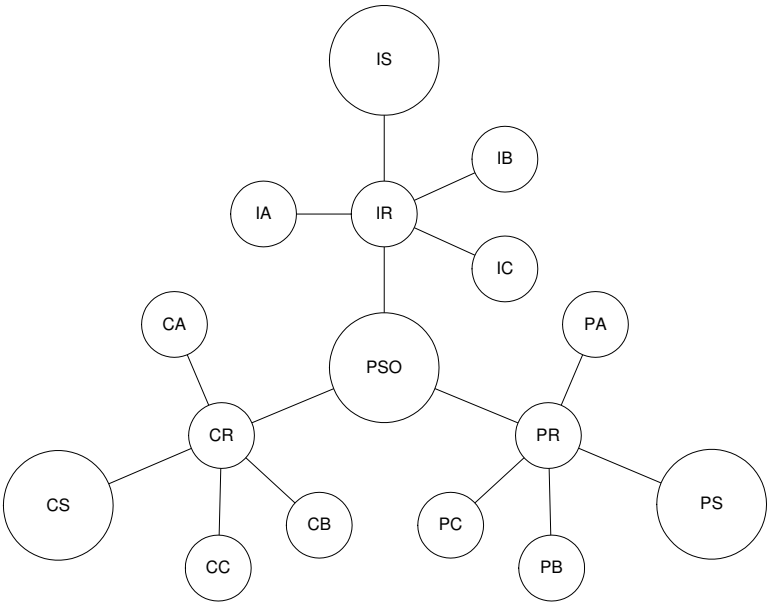

Fig. 7. Three Sides of UCON Model for Privacy Sensitive Objects

ical information for other reasons than TPO, healthcare providers are required to obtain written authorization documents. In Privacy Rule, authorization is more detailed and specific than consent. In PCASSO, neither consent nor authorization is included in the system. Moreover, usage and disclosure of patient medical information is entirely up to a primary care provider. For better control of all parties on patients' healthcare information and for better privacy protection, these consent and authorization should be part of identifiee rights in UCON model. Also, it should be the patient who holds those identifiee rights.

6.4 Reverse UCON

As mentioned above, obtaining or exercising usage rights on a digital object may create another digital information object (derivative object) which also needs controls for its access and usage. Some examples are payment information, usage log, etc. The usage control on these derivative objects is reversed in its control direction in such a way that the provider subject becomes the consumer subject and vice versa. This reversed usage control is called reverse UCON and the rights are called reverse rights. Furthermore, obtaining or exercising the reverse rights on these derivative objects may also create other derivative objects and reverse (more correctly inverse) rights on it.

Figure 8 shows an example of reverse UCON. Some components are omitted in this diagram for the sake of simplicity. Suppose Alice wants to listen to a MP3 music file. To obtain play rights, she as a consumer subject (CS) may have to agree on payment-per-play (OB: obligation) and provide credit card information. Upon her exercise of the play rights, she has to report her usage log on the MP3 file

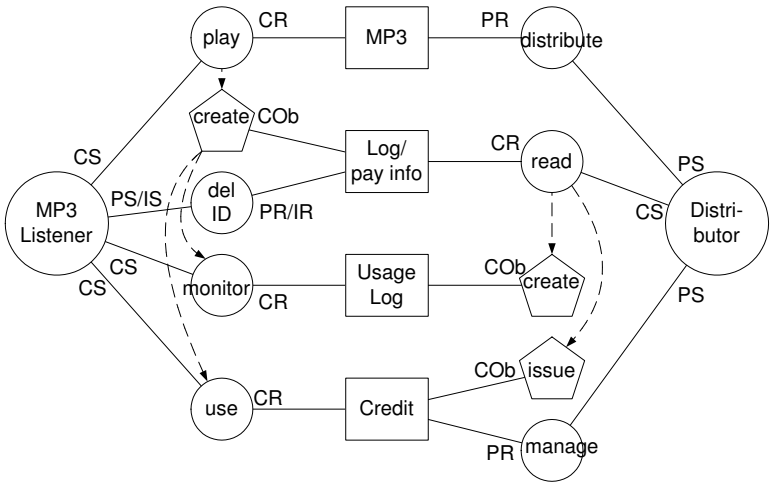

Fig. 8. An Example of Reverse UCON

(OB). In UCON, this payment information and log information are also considered as objects (derivative objects) and as part of UCON model. Now Alice becomes both a provider subject (PS) and an identifiee subject (IS) of the log/payment information and may hold certain rights (PR and IR) on them such as the right that she can delete her ID from log information. The distributor may have rights to collect log information either by putting an obligation on consumer rights or by giving consumer rights to get some store credits on log reports. If Alice has rights to get some store credit based on her play time, then it is now distributor's obligation as a provider subject to issue certain credit to Alice.

Control and protection of rights and usage of rights on the derivative objects have been hardly recognized or discussed in information security literature. In UCON, reverse UCON can be viewed as part of the UCON model and is not different from ordinary UCON in its model specifications. In general, derivative objects are likely to include privacy-related information. Adequate controls on derivative objects will be crucial for better privacy treatment. By handling derivative objects in UCON system, at least security and privacy issues can be discussed systematically within a common framework.

UCON systems are likely to be implemented and managed under the control of one of three subject sides: consumer, provider or identifiee. This implies it's hard to guarantee availability of adequate control mechanisms implemented for the other two sides on the rights and usage of rights. There can be also a third party who develops/manages UCON system on behalf of all of PS, CS and IS sides. Therefore, to make a sound reverse UCON system available, there should be either a voluntary commitment from a development/management group or legal enforcement. In its implementation, UCON system may have to include following mechanisms for reverse UCON.

—To provide ability to review detail of derivative objects which are going to be created.

—To provide ability to refuse creation of derivative objects (the consumer may have to give up or reduce exercising original rights).

—To provide ability to restrict reverse usage by blocking certain part of derivative objects (i.e., identity) or by allowing only aggregated information of individual objects.

—To provide ability to monitor reverse usage on derivative objects (this may cause another round of reverse UCON).

7. RELATED WORK

Several lines of related research were discussed in the introduction. Of these the policy-based authorization representation and enforcement model of [Ryutov and Neuman 2001; 2002] is possibly the closest to our ABC models. This model builds authorizations from objects, rights and conditions. A possible implementation by means of extended access control lists (EACLs) is outlined. Subjects are not explicitly recognized in EACLs but are rather embedded into conditions. Similar to ABC, this model also recognize pre- and mid-conditions. We feel that ABC pre and ongoing decisions are more precisely and systematically defined. In addition to pre- and mid-conditions, the model also identifies "post-conditions" which may have "side-effects" as part of the model to resolve update timing issue. However its definition of post-condition is closer to an implementation level aspect. In ABC model point of view, it may be viewed as a special case of pre-authorization with pre-update model ($UCON_{preA_1}$) implementation. This is largely because of its lack of systematic treatment on mutability issue. At this point the ABC model is more mature and comprehensive.

The term 'obligation' has been used with different meanings in the literature. Damianou et al. introduced the Ponder policy specification language [Damianou et al. 2001]. Ponder policies consists of authorization, obligation, refrain, and delegation policies. Schaad and Moffett have discussed further on the obligation part of Ponder [Schaad and Moffett 2002]. In both case, obligations are duties that have to be done independently from users' access requests. For example, software developer of a project A in an organization may have duties to provide weekly progress report to project manager. These duties are given to him not because he has requested certain accesses, but because he has assigned to a software developer role in the organization. Schaad and Moffett argue that obligations require authorizations so the required actions can be performed. By definition, UCON obligation is different from Ponder obligations (or duties). In ABC model, obligations are what subjects have to perform before or during (or even after in case of global obligation) obtaining or exercising usages. If an obligation is required, it just has to be done and does not require any authorization process for obligation fulfillment. Fundamentally, obligations in ABC model are different from duties. Also, ABC model does not include any concept of duties in its model.

The notion of 'provisional authorization' has been introduced in recent literature [Kudo and Hada 2000; Jajodia et al. 2001]. In a narrow definition, provision is what has to be performed prior to the authorization of usage requests. Provision is similar to UCON pre-obligations. Bettini et al. have discussed the notion of 'obligation' [Bettini et al. 2002]. Here, obligation is what has to be performed

after authorization decisions. This is similar to our global obligations. Neither has defined a notion for ongoing-obligations which have to be fulfilled continuously or regularly while the requested action is being performed. In the ABC model, obligations are defined and discussed in systematic manner so that they can be used for various situations with finer-grained controls. What really sets ABC apart from other research efforts is its systematic and comprehensive effort to provide a new intellectual foundation for access control. No prior effort has this reach and scope.

In terms of industry trends two ongoing efforts are worth mentioning. ContentGuard's eXtensible rights Markup Language (XrML), evolved from Xerox PARC's DPRL, has emerged as an OASIS based standard for rights expression languages [ContentGuard 2002; Wang et al. 2002]. As defined in its XML schema-based specification version 2, 'grant' consists of four entities called principal, rights, resource, and condition [ContentGuard 2002; Wang et al. 2002]. XrML conditions include terms, conditions and obligations. However their definition of terms, conditions and obligations are different from our conditions and obligations and not as precise as ours. Furthermore, while XrML may express various rules and policies for rights, it fails to resolve a transaction-based decision-making process. For example, XrML can express 'student can play a MP3 file 5 times' but assumes usage history of 'play' rights is supported by applications. Hence, it fails to resolve mutable cases such as 'after being played 2 times, now the MP3 file can be played only 3 more times'. Also there is no attempt to express ongoing decision-making. Similar shortcomings can be found also in OASIS eXtensible Access Control Markup Language (XACML) [Godik and Moses 2002]. Although authorization in XACML is based on transaction or request, it fails to cover mutable cases and ongoing cases.

8. CONCLUSION AND FUTURE WORK

In this paper we have introduced a novel concept called usage control and its core model called ABC model for controlling access to and usage of digital resources. Usage control encompasses traditional access control, trust management, and digital rights management and goes beyond them in its scope. By unifying these diverse areas in a systematic manner, the ABC model offers a promising approach for the next generation of access control.

We have given a description of the ABC family of models for usage control. The models, and their relationships, are summarized in Figure 4. We emphasize that we have only described the "pure" models corresponding to individual points in these figures. In practice we would expect real systems to use composite models which combine several of these together. Space does not permit us to explore the expressive power of combined models. Nonetheless, we have shown by example that a wide range of policies can be easily expressed in these models.

It has been our explicit goal to accommodate all the ideas we have seen in the access control literature in the past decade in a single unified framework. In particular we have looked to the Digital Rights Management community for inspiration to take us beyond the usual bounds of access control.

We believe the ABC models achieve the desired unification at an appropriate level of abstraction and provide a solid foundation for further research. ABC leaves

open the architecture and mechanisms for providing trusted attributes. This is one of the important challenges as we look ahead. We notice that delegation of rights is among the crucial issues that should be covered within UCON framework. In addition, there should be a clear description of administration issues. We believe further studies on these issues will provide more comprehensive solution approaches for the area of usage control.

There is increasing realization that traditional access control is not adequate for modern application needs. Many researchers have published possible extensions to the basic access control concepts. This paper is the first effort to overhaul the underlying foundation of access control itself. It provides a robust and integrated framework for access control models and systems of the future. We hope this ABC approach will re-unify a discipline that is starting to get fragmented at a time when the importance of access control is being increasingly appreciated.

REFERENCES

Abadi, M. and Burrows, M. and Lampson, B. 1993. A Calculus for Access Control in Distributed Systems. *ACM TOPLAS*, 15(4), pp 706–734

Anderson, Ross 2002. TCPA / Palladium Frequently Asked Questions *Online Available:http://www.cl.cam.ac.uk/~rja14/tcpa-faq.html*.

Arbaugh, William 1997. A Secure and Reliable Bootstrap Architecture *Proceedings of the IEEE Symposium on Security and Privacy* pp 65-71.

Baker, Dixie , Barnhart, Robert. and Buss, Teresa. 1997. PCASSO: Applying and Extending State-of-the-Art Security in the Healthcare Domain. *Proceedings of the Annual Computer Security Applications Conference*.

Bell, D. and LaPadula, L. 1973. Secure Computer Systems: Mathematical Foundations and Model. *MITRE Report, MTR 2547, v2, November*.

Bettini et al. 2002. Obligation Monitoring in Policy Management. *Proceedings of the Workshop on Policies for Distributed Systems and Networks*.

Blaze, M., Feigenbaum, J., and, Lacy, J. 1996. Decentralized Trust Management. *Proceedings of IEEE Symposium on Security and Privacy*.

ContentGuard 2002. XrML: Extensible rights Markup Language Core 2.1 Specification. *Online, Available: http://www.xrml.org*.

Damianou et al. 2001. The Ponder Policy Specification Language. *Proceedings of the Workshop on Policies for Distributed Systems and Networks*.

Godik, Simon and Moses, Tim 2002. OASIS eXtensible Access Control Markup Language (XACML) Specification 1.0. *Online, Available: http://www.oasis-open.org/committees/xacml/docs/*.

Gunter, Carl, Weeks, Stephen ,and Wright, Andrew 2001. Models and Languages for Digital Rights. *Proceedings of the Hawaii International Conference on System Sciences*.

Harrison, M.H., Ruzzo, W.L. and Ullman, J.D. 1976. Protection in Operating Systems. CACM, 19(8):461–471

Herzberg, A., Mass, Y., Mihaeli, J., Naor, D., and, Ravid, Y. 2000. Access Control Meets Public Key Infrastructure, Or: Assigning Roles to Strangers. *Proceedings of IEEE Symposium on Security and Privacy*.

HHS 2002. Standards for Privacy of Individually Identifiable Health Information. *Online, Available: http://www.hhs.gov/ocr/hipaa/finalreg.html*.

Iannella, Renato 2002. Open Digital Rights Language V1.1. *Online, Available: http://odrl.net*.

Jajodia, S, Kudo, M., and Subrahmanian, V.S. 2001. Provisional Authorizations. *E-Commerce Security and Privacy* Anup Gosh (Ed.) Kluwer Academic Press, 2001.

Kaplan, Marc 1996. IBM Cryptolopes, Superdistribution and Digital Right Management. *Online, Available:http://www.research.ibm.com/people/k/kaplan*.

KUDO, MICHIHARU AND HADA, SATOSHI 2000. XML Document Security based on Provisional Authorization. *Proceedings of the ACM Conference on Computer and Communications Security.*

CARL LANDWEHR 1997. Protection (Security) Models and Policy The Computer Science and Engineering Handbook, CRC Press, pp 1914–1928

LAMPSON, B.W. 1971. Protection. 5th Princeton Symposium on Information Science and Systems, Reprinted in *ACM Operating Systems Review* 8(1):18–24, 1974

B. LAMACCHIA 2002. Key Challenges in DRM: An Industry Perspective. Proc. 2nd ACM DRM Workshop (in conjunction with ACM CCS Conference) November.

MIT 2001. Ten emerging technologies that will change the world. *MIT Technology Review* (Jan/Feb).

P3P 2002. The Platform for Privacy Preferences 1.0 (P3P1.0) Specification. *Online, Available: http://www.w3.org/P3P/.*

PARK, JAEHONG AND SANDHU, RAVI 2002. Towards Usage Control Models: Beyond Traditional Access Control. *Proceedings of Seventh ACM Symposium on Access Control Models and Technologies.*

B. ROSENBLATT, B. TRIPPE AND S. MOONEY 2002. *Digital Rights Management: Business and Technology.* M&T Books.

RYUTOV, TATYANA AND NEUMAN, CLIFFORD 2001. The Set and Function Approach to Modeling Authorization in Distributed Systems. *Proceedings of the Workshop on Mathematical Methods and Models and Architecture for Computer Networks Security.*

RYUTOV, TATYANA AND NEUMAN, CLIFFORD 2002. The Specification and Enforcement of Advanced Security Policies. *Proceedings of the Workshop on Policies for Distributed Systems and Networks.*

SANDHU, RAVI 1993. Lattice-Based Access Control Models. *IEEE Computer* (November), pp 9 - 19.

SANDHU, RAVI AND SAMARATI, PIERANGELA 1994. Access Control: Principles and Practice. *IEEE Communication Magazine* (September), pp 40 - 48.

SANDHU, R., COYNE, E., FEINSTEIN, H. AND YOUMAN, C. 1996. Role-Based Access Control Models. *IEEE Computer* (February), pp 38 - 47.

RAVI SANDHU 2000. Engineering Authority and Trust in Cyberspace: The OM-AM and RBAC Way. Proc. 5th ACM Workshop on Role-Based Access Control, Berlin, Germany, July 26-28, 2000, pages 111-119.

SCHAAD, ANDREAS AND MOFFETT, JONATHAN 2002. Delegation of Obligation. *Proceedings of the Workshop on Policies for Distributed Systems and Networks.*

SCHNECK, PAUL 1999. Persistent Access Control to Prevent Piracy of Digital Information. *Proceedings of the IEEE* Vol. 87, No. 7, (July).

SIBERT ET AL. 1995. The DigiBox: A self-Protecting Container for Information Commerce. *Proceedings of USENIX Workshop on Electronic Commerce.*

TCPA 2002. Trusted Computing Paltform Alliance, Main Specification V1.1b *Online Available:http://www.trustedcomputing.org/docs.*

ROSHAN THOMAS AND RAVI SANDHU 1997. Task-based Authorization Controls (TBAC): Models for Active and Enterprise-Oriented Authorization Management Database Security XI: Status and Prospects,North-Holland

WANG ET AL. 2002. XrML - eXtensible rights Markup Language. *Proceedings of ACM Workshop on XML Security.*

WEEKS, STEPHEN. 2001. Understanding Trust Management Systems. *Proceedings of IEEE Symposium on Security and Privacy.*

WINSBOROUGH, W., SEAMONS, K., AND, JONES, V. 2000. Automated Trust Negotiation. *Proceedings of the DARPA Information Survivability Conference and Exposition.*