This is the accepted version of this conference paper. Published as:


© Copyright 2010 Australian Computer Society, Inc.
An Administrative Model for $UCON_{ABC}$

Farzad Salim  Jason Reid  Ed Dawson

Information Security Institute, Queensland University of Technology, GPO Box 2434, Brisbane Queensland 4001, Australia
Email: \{farzad, reid, e.dawson\}@isi.qut.edu.au

Abstract

$UCON_{ABC}$ is an emerging access control framework that lacks an administration model. In this paper we define the problem of administration and propose a novel administrative model. At the core of this model is the concept of attribute, which is also the central component of $UCON_{ABC}$. In our model, attributes are created by the assertions of subjects, which ascribe properties/rights to other subjects or objects. Through such a treatment of attributes, administration capabilities can be delegated from one subject to another and as a consequence $UCON_{ABC}$ is improved in three aspects. First, immutable attributes that are currently considered as external to the model can be incorporated and thereby treated as mutable attributes. Second, the current arbitrary categorisation of users (as modifiers of attributes), to system and administrator can be removed. Attributes and objects are only modifiable by those who possess administration capability over them. Third, the delegation of administration over objects and properties that is not currently expressible in $UCON_{ABC}$ is made possible.

Keywords: Access Control, Trust Management, Usage Control, Authorisation, Administration.

1 Introduction

The advent of the Internet and large scale Intranet has led to the emergence of a plethora of new applications (e.g., resource sharing, electronic commerce, health care systems) in which authorisation is significantly different from that of more traditional centralised or closed systems, from two main perspectives: the absence of a central administrator and the lack of prior knowledge held by resource providers and access requesters about each other.

$UCON$ is a new and emerging abstract\(^1\) authorisation framework that attempts to combine features from traditional access control, trust management and digital rights management. The concept of $UCON$ was introduced by Park and Sandhu (Sandhu & Park 2003, Park & Sandhu 2002b) and further refined into a formal model of $UCON_{ABC}$ that specifically focuses on the authorisation, obligation and condition aspects of access control (Park & Sandhu 2004).

At the heart of $UCON_{ABC}$ lies the concept of attributes and the primary contribution of the model is the manipulation of attributes throughout an access process. In this model, attributes are considered abstractly as the properties (e.g., role, classification, credit, clearance) of subjects or objects in the model. They are also conceptually divided into two categories, immutable and mutable. The former is left out of the $UCON_{ABC}$ model, as they are admin-controlled, meaning only an administrator can modify them. Mutable attributes are those whose value can change throughout an access process, usually as a consequence of subjects’ actions. To eliminate complexities such as who has a right to modify the attributes and how rights are to be assigned and enforced, mutable attributes are considered to be modified by the system (i.e., system-controlled), an abstraction for a trusted process or user (Park & Sandhu 2004, Park et al. 2004).

The need for a clear definition of an administration and attribute management model for $UCON_{ABC}$ has been mentioned several times in the literature (Park & Sandhu 2002b, Sandhu & Park 2003, Park & Sandhu 2004, Park et al. 2004, Zhang et al. 2004) but left aside for future research. Park and Sandhu (Park & Sandhu 2004), the originators of the $UCON$ concept suggest 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.”. This need has also been noticed by others (Zhang et al. 2007, Luo et al. 2008, Wang & Wang 2007) and attempts have been made to address it through annotating the $UCON_{ABC}$ model with ad-hoc delegation or administration elements. These approaches and their shortcomings will be discussed in more detail in Section 3.

We argue that administration is a separate matter, orthogonal to $UCON_{ABC}$ itself. Administration deals with the question of who is supposed to provide the required policies for an authorisation. We refer to this as the Administrative Model. The latter determines who must be authorised/denied given the relevant authorisation policies, which we refer to as the Usage Model. We believe that in the context of $UCON_{ABC}$, administration must be addressed through the management of attributes. Therefore, an administration model defines what attributes are, where they come from and who can manipulate them. Further, since objects have attributes, through the administration of attributes, such an administrative model will allow the administration of objects as well. Hence, the connection between the administration model and the $UCON_{ABC}$ model stems from the fact that the former defines attributes and the latter employs them for an authorisation process.

This paper proposes a novel administrative model for $UCON_{ABC}$, where the properties/rights of a sub-

---

\(^1\)Note that abstractness is a major design principle for $UCON$. By abstract we mean it must be independent of any specific existing language, policy or model.

Copyright ©2010, Australian Computer Society, Inc. This paper appeared at the Australasian Information Security Conference (AISC), Brisbane, Australia. Conferences in Research and Practice in Information Technology (CRPIT), Vol. 105, Colin Boyd and Willy Susilo, Ed. Reproduction for academic, not-for-profit purposes permitted provided this text is included.
ject or an object are defined by attributes which are formed through assertions made directly by a subject or indirectly by others to whom he/she delegated the administration capability. A key contribution of the proposed model is its introduction of a two layer structure which comprises a peer model and authoriser model. The peer model provides an expressive unrestricted environment where every subject may state their beliefs about properties and rights of other subjects/objects. Through this, attributes become source centric. Initially the issuer of an attribute is the sole administrator of the attribute, until they assert otherwise. The implication of this is the ability to identify who can modify or delegate properties and rights. The authoriser model, sits on top of the peer model and is responsible for determining whose assertions are to be taken into account for an access request. In order to make such an adjudication the model depends on the system/application wide administrative policy which we assume subjects operating within the model have consented to. We suggest no specific policy but provide some examples of well known policies such as owner-based, used in Digital Rights Management (DRM) or privacy aware applications, and administrator-based, for traditional access control systems.

Through such a formulation of the administration problem we make the adequacy of trust management techniques for the administration of UCONABC evident. In order to communicate our ideas more concretely we use the syntax of the SecPal language (Becker et al. 2007). We use some of the basic constructs that exist in almost any trust management language to show its function in the proposed administrative model. However, the model is not defined or limited by SecPal, since it is not part of the proposed extensions. This is necessarily in keeping with the abstract nature of UCONABC.

Our administrative framework specifically improves UCONABC in three aspects. First, it lifts the current assumption of a single administrator who issues attributes and the authorisation policies. Second, it removes the arbitrary division of attributes into mutable (those that are system-controlled) and immutable attributes (those that are admin-controlled). Hence, all attributes are conditionally mutable and can be treated within the current UCONABC model. Consequently, it would allow the construction of a model with either top-down or bottom-up propagation of administration, in which, administration capabilities over attributes and objects are delegatable and the administration root is dynamically determined with respect to a specific application policy. Such dynamism is actually one of the strong advantages of the proposed model. Third, the arbitrary categorization of users as modifiers of attributes into system and administrator is removed. There only exist subjects, who can modify the attributes they administer or for which they have been delegated administration capabilities.

The rest of this paper is organised as follows. Section 2 outlines the main concepts of UCONABC, on which our paper is based. This is followed by Section 3 that reviews related approaches to the administration problem. Section 4 introduces a motivating example. The actual administration model is described in Section 5 and this is followed by Section 6, where we make concluding remarks.

2 UCONABC Components

The UCONABC model (Park & Sandhu 2004) extends traditional access control to address the problem of authorisation not only at the time of access to a resource but also during its usage. The main components of the model as shown in Figure 1, are the Subjects (S) that wish to use their rights (R) over certain Objects (O). Subjects and objects are endowed with Attributes (A) that capture the properties of these entities. Authorisation (AU) is a functional predicate that evaluates usage requests based on the subjects’ and objects’ attributes, the requested rights, the policy model, and returns either yes or no. In addition to authorisation there are two other decision factors, obligations (B) which is a functional predicate that ensures certain obligation actions are performed by the subject and Condition (C) predicates, where environmental requirements that have to be satisfied are checked as a part of the usage decision process. The question of how the propositional value of these predicates are determined is currently external to the model.

Two innovations for UCONABC are attribute mutability and continuity. Different from other access control models, subject or object attributes in this model can not only be modified by the administrator, but can also be changed as a side effect of a subject’s usage of an object. However, there is a special user called system who is assumed to observe these actions and make proper updates. The concept of continuity proposes that the system repeatedly checks the validity of subjects’ rights during the access. When a right is revoked, the access will be terminated on time. These two properties uniquely identify UCONABC and are indispensable in an open network environment.

3 Related Works

In UCONABC (Park & Sandhu 2004) attributes are the central component of the model as access decisions are based on the state of subjects’ and objects’ attributes, which may change and subsequently influence the usage control decision. Despite the important role of attributes in usage control, UCONABC does not address challenges regarding the management of attributes, focusing instead on user authorisation issues. However, the need for an attribute management mechanism has been noted by UCONABC designers (Sandhu & Park 2003, Park & Sandhu 2002b, 2004).

In the original UCONABC papers of Park et al. (Park & Sandhu 2002b, 2004), attributes are divided into ‘admin-controlled’ and ‘system-controlled’. The admin-controlled attributes are said to be immutable in that they are only assigned to subjects and objects by administrator actions and cannot be modified by the system automatically, whilst system-controlled attributes can be updated as side effects of user’s us-
age of objects. Park et al. (Park et al. 2004) further refine admin-controlled attributes to be either 'security-officer-controlled' or 'user-controlled', where the user could be either the subject possessing the property, 'self-controlled', or some other user within the model, 'non-self-controlled'. They leave further details on administration issues for future work. As we will discuss in Section 5, such a syntactic classification of attributes and users is rather arbitrary and meaningless without being able to define attributes and access control policies and by them they can be updated. Furthermore, the above papers always imply the need for a central administrator who assigns attributes and rights to users. This is inconsistent with one of the primary design goals of usage control, to address open environments. The issues regarding delegation are also not considered.

In addition to the above mentioned papers there are a number of other proposals that attempt to provide an administration/delegation for UCONABC. Zhang et al., (Zhang et al. 2007) proposed UCONπ, a delegation model for UCONABC. Their view of delegation is limited to access level permissions, where a user grants some of their own rights (e.g., read/write) to another user. They introduce several entities specifically for delegation purposes: ‘delegator’, ‘delegated’ ‘delegation context’ and ‘permission’, as attributes, referred to as ‘delegation attributes’. The contribution of their proposal is to suggest the importance of expressing the relationships between subjects. However, the proposed extension arguably violates the generalised and abstract nature of the original UCONABC model. Further, there is no clear link between the proposed model and the UCONABC model.

Luo et al. (Luo et al. 2008) attempt to integrate UCONABC with ideas from trust management to introduce a distributed delegation model, referred to as UTCDM. They use directed graphs as a representation tool to express delegation relationships and to provide a credential discovery algorithm. While credential discovery is one of the main areas in trust management, the link between UCONABC and their proposed mechanism is unclear. Many well studied trust management and credential discovery frameworks already exist (Blaze et al. 1998, Li et al. 2003) that could be used to address the issues proposed in this paper. However, UCONABC is meant to be an abstract framework, thus their proposal arguably violates the abstractness principle inherent in UCONABC.

Wang et al. (Wang & Wang 2007) use UCONABC’s abstract concept of attribute as ‘credibility’ to express the trustworthiness of subjects who attempt to access an object. From this perspective their proposal is trivial as the concept of attribute in UCONABC is clearly abstract enough to express trust, credibility, role, etc. They also attempt to establish that, by using attributes as certificates, one can reduce certain complexities such as certificate revocation and certificate discovery that exist in certificate based access control models. However, it is not clear how their proposal would reduce such complexities. There are several theoretical and practical issues that the paper fails to address, such as: who is able to assign attributes to subjects regardless of what they represent, and how and by whom these attributes are to be modified.

Originator Control (ORCON) is a proposal for an access control policy in which recipients of information need to gain the originator’s approval for the further re-dissemination of the information (Park & Sandhu 2002a). They put emphasis on the importance of adapting the originator control policy within UCON as a means for it to go beyond traditional access control, trust management models and DRM. However, the paper leans toward implementation rather than an abstract theoretical work. They describe several ways in which licenses and delegation tickets could be used to control redistribution of a resource. As we will describe in Section 5, explicit description of ownership and owner-based control are two examples of a system policy that an abstract framework like UCON must be capable of modelling, but it must not be limited only to this.

On a similar ground to our work, Firozabadi et al. (Firozabadi et al. 2002) and Wood et al. (Wood & Fernandez 1979) address the requirements of a decentralised administration/authorisation model and distinguish between two kinds of delegations: delegation of authority at management level and delegation of permission at request level. The delegation of authority allows an entity to hand off authorities to another entity such that the receiver can express authorisation policies on behalf of the sender. On the other hand, permissions are privileges to exercise the rights of a specific entity - their delegation allows the receiving entity to access resources on behalf of the other. Drawing this distinction allows them to introduce constraints required for each category. However, the main contribution of these proposals is in dissecting the distinction between the delegation types rather than providing a formal language to express them. These works have inspired the main ideas that underlie our proposal.

In the following sections we will introduce a general administrative model that is policy agnostic and therefore adheres to the abstraction level inherent in UCONABC, while addressing its limitations in defining, issuing, delegating and dealing with the modification of rights and properties.

4 Motivating Example

To clarify the problem that our administrative model seeks to address, consider the following simple example, taken from (Park & Sandhu 2004) which addresses ‘DRM pay-per-use’ by using UCONπpreA3.2. In this model, two attributes are assumed to exist:

- credit(s): subject’s credit (measurement unit is money).
- value(o,r): object’s value (the amount of money for a given right on object).

There are also two policies:

- allowed(s,o,r) ⇐ credit(s) ≥ value(o,r).
- update(credit(s)): credit(s) = credit(s) - value(o,r).

The policies are intuitively read as: to accept a request the subject (requesting) must have enough credit. In that case the subject’s credit is modified by reducing the value of the object that is being requested and access is granted.

Given the above example, there are three main points that are assumed and left outside the UCONABC model.

1. It is not clear who determines the properties such as subject’s credit and object’s value. Such properties (in UCONABC term, attributes) belong to a subject or an object.

2. It is not clear who determines the rights for a subject on an object. In the above example, allowed(s,o,r).

3The subscript refers to the pre-authorisation with pre-update policy - interested readers refer to the original paper.
3. Although the modification of such properties is at the core of UCONABC, policy governing modification of properties is not explicit. Therefore it is not practical to determine who can modify these properties. In the context of the above example, it is not clear who can update the credit property of a subject (i.e., update(credit(s))).

UCONABC simply assumes that update(credit(s)) is performed by system, an entity outside the model, trusted to do so. By the same token, rights that specify the relationship between subjects and objects are assumed to exist and the model does not care about the origin of the rights.

It is important to appreciate that properties and rights are subjective by nature. In the real world, attributes and rights are acquired from the sources that have the authority to provide them. For example, a subject may acquire credit from a bank or a driving license from the traffic authority; those dealing with the subject may trust the bank to honour the provided credit or not. However, the only entity capable of modifying a subject’s credit must be the bank or those somehow appointed by the bank to do so. The objective of the administrative model for UCONABC is to address these issues.

5 Administrative Model (M)

Here we introduce a novel approach in representing administrative functions. We conceptually divide the UCONABC Administrative Model (M) into, the Peer Model (M_P), that defines an unrestricted basis for establishing relationships between subjects and objects through assertions, and the Authoriser Model (M_A), that provides a means for selecting and honouring some of the existing assertions with respect to an administrative policy of the system that implements the UCONABC model.

![Figure 2: Administrative Model (M)](image)

The administrative model, shown in Figure 2, is abstracted into four major components: Subjects (S) that are inter-connected, Objects (O) that are administered, owned or used\(^3\) by subjects, Assertions (A) that specify the relationships between subjects and objects, and finally the Administrative Policy (P), denoting system policy.

We use a specific name when we are addressing an element of the finite set of S or O. For example Alice, Bob, Carol ∈ S or File ∈ O. Further, we use a lower-case letter (e.g., s) when we are addressing a variable.

5.1 Peer Model (M_P): Subjects, Objects, Assertions

At the heart of the UCONABC model is the concept of attributes and at the center of M_P is the concept of assertion shown in Figure 3. The relationship between assertions and attributes stems from the fact that assertions in our model generate the attributes of the UCONABC model. Assertions are also used to express rights, conditions and obligations of UCONABC as well.

We generalise an assertion to be a belief statement expressed by a subject about the property (T) or the rights (R)\(^4\) of another subject (including themselves) or an object. An assertion could state various things such as what is the subject’s role, age, credit balance, clearance or object’s classification, value or trustworthiness from the issuer’s perspective.

![Figure 3: Assertions A](image)

In relation to expressing conditions and obligations within M_P, we argue that both of these are objects with properties under the control of a specific subject. This is different view from the former access control models, including UCONABC that consider these to be a special (type) entity, other than a common object and considers the subject modifying them to be external to the model. Their approach results in a simpler model as they do not need to deal with the manipulation of these entities by users (subjects).

For example, in UCONABC, local time is considered as a condition under the control of external subject, environment. However, in reality, the local time is the property of an object, system clock, which is under the control of a subject, root in a Unix operating system. By a similar token, in UCONABC, a user’s assent to a privacy policy by ticking a policy form is considered as the discharge of an obligation. In our model, the policy form could be considered as an object with a property, filled, that could be assigned values true or false by a subject (user).

In all the above cases an assertion states that its issuer believes\(^5\) that a subject or an object has a property or a right. Through such a treatment we inherit a subjective view, where properties and rights are always formed from their issuer’s perspective and are always valid from that aspect. We refer to such claims that a subject makes in an assertion as (subjective) fact. The grammar of facts are shown in Table 1.

There are two types of assertion that could be written over a fact: a direct assertion and an indirect/delegation assertion. Direct assertions are the basis of the model and take the form:

\[ s \text{ says} \text{ fact} (\text{if fact}_1 \ldots \text{fact}_n) \]

Informally, a direct assertion states that the subject believes (says) a fact. The assertion may also be conditional upon the existence of some other facts asserted by the issuer. In the above assertion s ∈ S is the issuer of the statement, says is a keyword, fact encodes a specific fact based on the grammar of facts.

---

\(^3\)In this paper we consider “use” and “access” to be synonyms.

\(^4\)Note that rights, permission or capability can be viewed as a property; for example granting a right r to a subject can be viewed as making an assertion that the subject has property r. However, here for clarity and to be aligned with the UCONABC approach we consider them to be separate.

\(^5\)Note that in our paper, believe, assert, state are synonyms, all mean a subject has made an assertion.
The if is an optional keyword, fact1, ..., factn are facts upon which the fact is conditional. The conditional facts must already exist within the set of assertions made by s, Assertion Context of s (Ac,s), in order to say: s says fact.

Following is an example of three direct assertions made by three subjects: Bank, Administrator and Alice in an imaginary UCON model. They intuitively mean Administrator believes that Bob has the clearance level 1: Bank states (believes) that Bob has a credit limit of $50, and Alice believes the object Book has a value of $10 for reading purposes.

\[\begin{align*}
\text{Admin says Bob hasClearance[1]} & \quad \text{(1)} \\
\text{Bank says Bob hasCredit[50]} & \quad \text{(2)} \\
\text{Alice says Book hasValue[10,read]} & \quad \text{(3)}
\end{align*}\]

Direct assertions are less expressive as they are unable to capture the dependency of a subject on another in order to make an assertion that brings about the propositional content of a fact. For example, only using direct assertions, it is not possible for Alice to depend on Administrator to determine the clearance property for Bob by:

\[\begin{align*}
\text{Alice says Bob hasClearance[1] if } & \quad \text{Administrator says Bob hasClearance[1]} \\
& \quad \text{(4)}
\end{align*}\]

This is because the conditional facts in a direct assertion must be deducible from the issuer's assertion context, in this case \(\text{Ac}_{Alice}\). Expressing such a dependency is the primary requirement in open environments, where subjects have limited knowledge about the properties or the rights of others. This need is addressed by delegation assertions that take the following form and allow a subject to state its willingness to believe certain types of facts asserted by other subjects:

\[s \text{ says } s' \text{ can say } D \text{ fact (if fact}_1, \text{... fact}_n)\]

The above delegation assertion introduces an extra keyword \text{can say} \(D\) which introduces the willingness of the issuer for accepting (believing) the assertions made by another subject \(s'\) about the fact. The delegations have arbitrary but specified depth, where \(D\) defines the possible depth of the delegation and takes the values \(n \in N \ldots \infty\) where \(D = 0\) means no delegation and \(\infty\) means an unbounded delegation. The depth of \(D = 2\) means that a subject (e.g., \(s\)) may delegate the assertion of a fact to another subject (e.g., \(s_1\)) and allow \(s_1\) to delegate to others but not allow these other delegates to further delegate.

\[\begin{align*}
\text{Alice says Admin can say}_0 s \text{ hasClearance} & \quad \text{(4)} \\
\text{Alice says Bank can say}_0 s \text{ hasCredit} & \quad \text{(5)}
\end{align*}\]

For example, given the direct assertions \(\{1,2\}\) and delegation assertions \(\{4,5\}\), one (i.e., Alice or any one having access to assertions) can deduce:

\[\begin{align*}
\text{Alice says Bob hasCredit[50]} & \quad \text{(6)} \\
\text{Alice says Bob hasClearance[1]} & \quad \text{(7)}
\end{align*}\]

As a result the assertion context of Alice \(\text{Ac}_{Alice}\) would consist of direct assertions \(\{3,6,7\}\). Notice that here we assumed subjects (e.g., Alice) are informed about the assertions made by other subjects in the model. The details of how such knowledge is shared and complexities regarding chains of assertion are directly related to the application employing our model and several approaches exist to address these issues e.g., (Li et al. 2003, Blaze et al. 1998).

Notice that through the above treatment, subjects within a UCON_{ABC} model are enabled to make assertions about properties and rights of other subjects and objects within the model. Further, they can delegate such assignments to other subjects. Here, despite UCON_{ABC}'s approach where attributes belong to subjects or objects, and where it is not clear who has specified the policy, properties and rights only exist from the perspective of their issuers. This does not necessarily mean that it is universally believed (by other subjects in the model) that the subject/object actually has the property/right in question. This leads us to the question, how could the UCON_{ABC} model decide whose (subject) perspective should be relied upon for making an authorisation decision? This question is answered by the authoriser model.

### 5.2 Authoriser Model (\(M_A\)): System policy, Assertions

The peer model introduces a flexible anarchic model where no authority is assumed. A subject may make an assertion about itself, other subjects or objects and these assertions could be honoured by others through delegation. However, such an anarchic model is meaningless if there are no means to determine whose assertions are to be taken into account for a UCON_{ABC} usage control decision, which must ultimately involve one or more subjects who act as the authority root—whose set of assertions is denoted as \(\text{Ac}_{Root}\).

The concept of authority root is inherent in, and the basis of, all the existing access control models. However, it is usually assumed fixed and defined outside the model itself. For example, in most RBAC models the authority root is a trusted central administrator who is assumed to assign permissions to roles and roles to users. Based on these assignments, the model determines whether an access request is
to be granted or denied. ORCON and privacy-oriented models consider the owner of the resource as the authority root and make access control decisions based on this assumption. Trust Management systems consider the authority root to be the principal called *local*, who writes the local policies. Since these models are based on predefined views about the root of authority, none is general enough to model the others.

To ensure the flexibility of our administrative model we bring the concept of authority root into the model and allow it to be explicitly defined through an *Administrative Policy*. The authoriser is defined as a function shown in Figure 4:

```
   p ∈ P  =>  Authoriser (M_A)  =>  AC_Root
```

Figure 4: System View: Determining Authority Root

where $P$ is a set of possible administrative policies. An administrative policy $p ∈ P$ precisely states the rules necessary to determine which subject(s) are to be considered as the authority root. The $AC$ is a set of all assertions made by subjects in $M_P$. In other words, it is the universal set of assertion contexts. Given $p$ and $AC$, $M_A$ determines the assertion context of the authority root, denoted as $AC_Root$.

Note that when more than one authority root is specified by $p$, there is a potential for conflict between their assertions. The nature of such conflicts depend on the language used for expressing administrative policies. Such a language could allow the expression of hierarchies and would ideally provide mechanisms for detecting inconsistencies or redundancies. Whilst the discussion of such conflicts and their resolution is important for the applicability of our proposal, the focus of this paper is on introducing an abstract administrative model and allow it to be explicitly defined through an *Administrative Policy*. The authoriser is defined as a function shown in Figure 4:

```
   p ∈ P  =>  Authoriser (M_A)  =>  AC_Root
```

Figure 4: System View: Determining Authority Root

where $P$ is a set of possible administrative policies. An administrative policy $p ∈ P$ precisely states the rules necessary to determine which subject(s) are to be considered as the authority root. The $AC$ is a set of all assertions made by subjects in $M_P$. In other words, it is the universal set of assertion contexts. Given $p$ and $AC$, $M_A$ determines the assertion context of the authority root, denoted as $AC_Root$.

Note that when more than one authority root is specified by $p$, there is a potential for conflict between their assertions. The nature of such conflicts depend on the language used for expressing administrative policies. Such a language could allow the expression of hierarchies and would ideally provide mechanisms for detecting inconsistencies or redundancies. Whilst the discussion of such conflicts and their resolution is important for the applicability of our proposal, the focus of this paper is on introducing an abstract administrative model and allow it to be explicitly defined through an *Administrative Policy*. The authoriser is defined as a function shown in Figure 4:

```
   p ∈ P  =>  Authoriser (M_A)  =>  AC_Root
```

Figure 4: System View: Determining Authority Root

where $P$ is a set of possible administrative policies. An administrative policy $p ∈ P$ precisely states the rules necessary to determine which subject(s) are to be considered as the authority root. The $AC$ is a set of all assertions made by subjects in $M_P$. In other words, it is the universal set of assertion contexts. Given $p$ and $AC$, $M_A$ determines the assertion context of the authority root, denoted as $AC_Root$.

Note that when more than one authority root is specified by $p$, there is a potential for conflict between their assertions. The nature of such conflicts depend on the language used for expressing administrative policies. Such a language could allow the expression of hierarchies and would ideally provide mechanisms for detecting inconsistencies or redundancies. Whilst the discussion of such conflicts and their resolution is important for the applicability of our proposal, the focus of this paper is on introducing an abstract administrative model and allow it to be explicitly defined through an *Administrative Policy*. The authoriser is defined as a function shown in Figure 4:

```
   p ∈ P  =>  Authoriser (M_A)  =>  AC_Root
```

Figure 4: System View: Determining Authority Root

where $P$ is a set of possible administrative policies. An administrative policy $p ∈ P$ precisely states the rules necessary to determine which subject(s) are to be considered as the authority root. The $AC$ is a set of all assertions made by subjects in $M_P$. In other words, it is the universal set of assertion contexts. Given $p$ and $AC$, $M_A$ determines the assertion context of the authority root, denoted as $AC_Root$.

Note that when more than one authority root is specified by $p$, there is a potential for conflict between their assertions. The nature of such conflicts depend on the language used for expressing administrative policies. Such a language could allow the expression of hierarchies and would ideally provide mechanisms for detecting inconsistencies or redundancies. Whilst the discussion of such conflicts and their resolution is important for the applicability of our proposal, the focus of this paper is on introducing an abstract administrative model and allow it to be explicitly defined through an *Administrative Policy*. The authoriser is defined as a function shown in Figure 4:

```
   p ∈ P  =>  Authoriser (M_A)  =>  AC_Root
```

Figure 4: System View: Determining Authority Root

where $P$ is a set of possible administrative policies. An administrative policy $p ∈ P$ precisely states the rules necessary to determine which subject(s) are to be considered as the authority root. The $AC$ is a set of all assertions made by subjects in $M_P$. In other words, it is the universal set of assertion contexts. Given $p$ and $AC$, $M_A$ determines the assertion context of the authority root, denoted as $AC_Root$.

Note that when more than one authority root is specified by $p$, there is a potential for conflict between their assertions. The nature of such conflicts depend on the language used for expressing administrative policies. Such a language could allow the expression of hierarchies and would ideally provide mechanisms for detecting inconsistencies or redundancies. Whilst the discussion of such conflicts and their resolution is important for the applicability of our proposal, the focus of this paper is on introducing an abstract administrative model and allow it to be explicitly defined through an *Administrative Policy*. The authoriser is defined as a function shown in Figure 4:

```
   p ∈ P  =>  Authoriser (M_A)  =>  AC_Root
```

Figure 4: System View: Determining Authority Root

where $P$ is a set of possible administrative policies. An administrative policy $p ∈ P$ precisely states the rules necessary to determine which subject(s) are to be considered as the authority root. The $AC$ is a set of all assertions made by subjects in $M_P$. In other words, it is the universal set of assertion contexts. Given $p$ and $AC$, $M_A$ determines the assertion context of the authority root, denoted as $AC_Root$.

Note that when more than one authority root is specified by $p$, there is a potential for conflict between their assertions. The nature of such conflicts depend on the language used for expressing administrative policies. Such a language could allow the expression of hierarchies and would ideally provide mechanisms for detecting inconsistencies or redundancies. Whilst the discussion of such conflicts and their resolution is important for the applicability of our proposal, the focus of this paper is on introducing an abstract administrative model and allow it to be explicitly defined through an *Administrative Policy*. The authoriser is defined as a function shown in Figure 4:

```
6 Conclusion and Future Work

The contributions of this paper are three fold. First, we introduced a novel administrative model based on two layers of abstraction, capable of representing both centralised as well as distributed administrative requirements to address different application domains. One layer introduces an anarchic environment where every subject, in addition to being an access requester, can be an administrator and specify rights and properties for other subjects or objects. Another layer introduces constraints based on application requirements to identify who can actually administer and update rights or properties. In this aspect, we made the concept of authority root that is inherent but external to the existing access control models, explicit and internal. Through this, given an application’s requirements, theoretically any number of authority roots could be defined, whom can specify any policy type or delegate such tasks. Thus the design delivers a desirable flexibility.

Second, we analysed the administrative problem in UCON$_{ABC}$ and showed how it can be addressed using existing trust management techniques, as a result, taking advantage of all the already developed functionalities (e.g., certificate delegation, revocation, etc.) that comes with them.

Finally, through concrete examples we showed how our proposed administrative model can address the specific problems identified within UCON$_{ABC}$ model. Precisely, the administrator who was considered a special entity external to the UCON$_{ABC}$ model can now be any subject(s) within the model. An arbitrary division of attributes into mutable, modified by subjects, and immutable, only to be modified by the administrator, is no longer necessary, since all attributes are made mutable. The arbitrary attribute modifier system is removed; attributes are source centric and can only be updated through their issuer or by those whom have been delegated relevant administrative authority over them.

We envisage the immediate future direction is to adopt one of the existing trust management languages to develop an administrative toolkit based on the concepts introduced in this paper. The developed framework is aimed towards a current application of UCON$_{ABC}$ in collaborative environments, such as the one proposed by Zhang et al., (Zhang et al. 2006). Further, we would like to examine the potential conflicts that may arise due to assertions made by multiple authority roots and introduce approaches to deal with them.

7 Acknowledgement

This research has been partially funded by the Australian Research Council - Project DP0773706.

References


