Extended RBAC with Role Attributes

Jianming Yong
Department of Information Systems, Faculty of Business
University of Southern Queensland, Australia
yongj@usq.edu.au

Elisa Bertino
CERIAS, Purdue University
West Lafayette,IN 47907, USA
bertino@cerias.purdue.edu

Mark Toleman Dave Roberts
Department of Information Systems, Faculty of Business
University of Southern Queensland, Australia
{markt,roberts}@usq.edu.au

Abstract

Though RBAC has been researched for many years as a current dominant access control technology, there are few researches to be done to address the further extension of the role which is the fundamental entity of RBAC. This paper tries to extend the role to a further level, the role attributes. Through the attributes, the function and operation on the role can be enhanced and extended. Through the attributes, ANSI RBAC is significantly extended. In the inheritance of hierarchical role, the privacy of its parental role can be kept by using HA (Hidden Attribute).

Key words: Access Control, RBAC, Role attributes

1 Introduction

After the Internet becomes one of most important infrastructures for modern societies, all systems have to have an access control policy in order to effectively protect the system security and privacy. There are a lot of access control technologies in the history. So far it is obvious that RBAC has become a domain access control technology. RBAC model is mainly based on the different roles which are assigned authorized permissions. Then the users use the system through obtaining needed roles. The fundamental entity of RBAC is the role. For many years, few researches have been done to further extend the role into a further step. This paper tries to expand the role into a next deep level, role attributes. The impact of the attributes on RBAC is thoroughly discussed in later sections.

This paper is organised as the follows: Section 2 introduces basic concepts of ANSI RBAC and its components. In Section 3, some research attempts towards the attributes are discussed although there are not many. In section 4, a further expansion of the role attributes is discussed, including the fundamental concepts, attribute operation, attribute types, basic attribute modelling. In Section 5, the basic relationships between the

attributes and all entities of ANSI RBAC are discussed. In Section 6, relationship comparisons between attributed-extended RBAC and ANSI RBAC from the perspectives of the core RBAC, hierarchical RBAC and constrained RBAC are briefly illustrated. In Section 7, a new diagram of attribute-extended RBAC is demonstrated. Section 8 addresses the conclusion remarks and future works.

2 RBAC

In February 2004, ANSI INCITS 359-2004 [ANSI, 2004] was approved. It is a milestone for RBAC research and development. It has a significant impact on the access control. So far RBAC has been accepted as a primary technology for system access control. It is a result of co-efforts by both academia and industry.

2.1 In Academia

From 1995 to present, there are 10 ACM symposiums held on Access Control Models and Technologies. Most scholarly papers published at the symposiums are more or less related to RBAC. If any search is done on ACM or IEEE digital libraries for RBAC, there will bring hundreds of papers which are relevant to RBAC. The fundamental concepts of RBAC were addressed by Sandhu, et al [1996]. In [Sandhu, 1996], RBAC was categorized as four sub-components, $RBAC_0$, $RBAC_1$, $RBAC_2$, and $RBAC_3$.

$RBAC_0$ is base model of RBAC. In ANSI INCIT 359-2004, $RBAC_0$ is called Core RBAC. Core RBAC has four fundamental entities: Users (U), Permissions (P), Sessions (S), and Roles (R). Users are human beings or their agents who have ability to use information systems. Roles are duties and responsibilities in an organisation or a system. Permissions are concerning access to roles with suitable users. Sessions are responsible for establishing dynamical relationships between users and roles. The set of relationships of U, P, S, and R is defined at Core RBAC as $U \times R$, $P \times R$, $S \rightarrow U$, and $S \rightarrow Set(R)$. $U \times R$ means the relationship mapping between users and roles. $P \times R$ means the relationship mapping between permissions and roles. $S \rightarrow U$ means that a specific user gets a required session or sessions. $S \rightarrow Set(R)$ means that a session is associated with a set of roles.

$RBAC_1$ is defined as role hierarchies. In ANSI INCIT 359-2004, $RBAC_1$ is called Hierarchical RBAC. Hierarchical RBAC is actually defined the relationships among different roles, especially role inheritance. The main purpose of hierarchical RBAC is to simplify the system administration and management.

$RBAC_2$ is defined as Constraints Model. In ANSI INCIT 359-2004, $RBAC_2$ is called Constrained RBAC. Constrained RBAC actually puts constrains on users what roles and permissions can be assignment to. It can prevent some users from abusing the system. Furthermore static constraint and dynamic constraint are introduced by Ferraiolo, et al [2001].

$RBAC_3$ is actually a combination of $RBAC_1$ and $RBAC_2$. It has not been listed as a separate component in ANSI INCIT 359-2004.

2.2In Industry

Many popular operating systems, like, Windows, Linux, Unix, are widely using roles to manage system resources. Major DBMS products, like Oracle, Sybase, etc, all support RBAC. Major enterprise integration solutions, like SAP, PeopleSoft, widely use the concept and mechanism of roles which have the similar functions as being defined in RBAC.

3 Related Works

The most fundamental concept of RBAC is the role. The role decides the security properties of information systems. It is important to do further research on the role. So far there are some further researches on the role, like environment role [Covington et al, 2003], parameterized role [Giuri and Iglio, 1997], and attributed role [Byun et al, 2005].

In [Covington et al, 2003], the environment role is defined as a role which is use to capture environmental conditions. The environment role has extended the traditional RBAC with a new type of role. When the users get their roles, they have to get support from relevant environment roles so that they can successfully execute their assigned roles. In [5], a role is associated with a template. A role template is represented as follows:

$$r(\chi_1, \chi_2, \chi_3, \chi_4, \chi_5, \ldots \chi_n)$$

Where r is the role name and $\chi_1, \chi_2, \chi_3, \chi_4, \chi_5, \ldots \chi_n$ are properties bound to the role, also called parameters of the role. In [Byun et al, 2005], the role attributes are defined, the notion of role attributes is very similar to the role parameters [Giuri and Iglio, 1997]. Unfortunately none of further research has been done to extend role attributes into RBAC. We are realising that the role attributes can be used to rebuild RBAC and to further enhance the central concept of the role. In the following sections, we will address the role attribute in details. The role attributes will have a big impact to current RBAC mechanism. Also the individual role's privacy can also be protected by an effective implementation of role attributes.

4 Attributes

A role consists of various attributes which demonstrate the property of the role and the permissions the role holds. In order to clearly descript how the role attributes to have the impacts on current RBAC mechanism, we need a further discussion on the role attributes as follows.

4.1 Definition of Attribute

Definition 1 (Attribute) An attribute (Attr) is a property of a role (r). A set of relevant attributes consists of a role. A role has a fix set of attributes. The relationships between the role and the attribute are presented as follows:

$$r(Attr) = r(Attr_0, Attr_1, \ldots, Attr_n)$$

$$\forall Attr_i, Attr_j \in r,\ 0 \le i \le n,\ 0 \le j \le n : \text{if } i \ne j, \text{ then } Attr_i \cap Attr_j = \Phi$$

$$\bigcup_{i=0}^{n} Attr_i = r(Attr)$$

If a system has a set of roles, R, R=R(r_0, r_1, r_2, …, r_m), and ATTR is as a full set of attributes for the system, we will have the following relationship:

$$ATTR = r_0(Attr) \bigcup r_1(Attr) \bigcup ... \bigcup r_m(Attr)$$

In order to use the role attributes to implement access control, we need to define more concepts to support this mechanism.

4.2 Attributes types

As the attributes are properties of a role, we define four types of attributes: compulsory, optional, obvious, and hidden.

4.2.1 Compulsory

Definition 2 (Compulsory Attribute (CA)) A compulsory attribute is an essential part of a role. If a compulsory attribute of a role is deactivated or a compulsory attribute has an empty value, the role cannot be assigned to any user and the role is also in a non-active situation.

$$\forall\ Attr_k \in ATTR,\ 0 \le k \le z\text{, z is the cardinality of ATTR}$$
$$\text{If } Attr_k \in r(Attr) \text{ and } Attr_k \text{ is a CA of r}$$
$$\text{Then } Attr_k \leftrightarrow r(Attr)$$

4.2.2 Optional

Definition 3 (Optional Attribute (OA)) An optional attribute is not an essential part of a role. If an OA of a role is deactivated or an OA has an empty value, the role can still be assigned to the users for a normal execution.

$$\forall\ Attr_k \in ATTR,\ 0 \le k \le z\text{, z is the cardinality of ATTR}$$
$$\text{If } Attr_k \in r(Attr) \text{ and } Attr_k \text{ is an OA of r}$$
$$\text{Then } Attr_k \mapsto r(Attr)$$

4.2.3 Obvious

Definition 4 (Obvious Attribute (ObA)) An obvious attribute of a role is visible to all other roles which have relationships with this role. ObA can be useful in the hierarchical RBAC and constraint RBAC. ObA can be formulized as the follows:

$$\forall\ Attr_k \in ATTR,\ 0 \le k \le z\text{, z is the cardinality of ATTR}$$
$$\text{If } Attr_k \in r_i(Attr) \text{ and } Attr_k \text{ is an ObA of } r_i \text{ and}$$
$$\text{If } r_i \times r_j \neq \Phi\ \&\ i \neq j \text{ Then } Attr_k \text{ is visible to } r_j$$

4.2.4 Hidden

Definition 5 (Hidden Attribute (HA)) A hidden attribute of a role is only visible to the role itself. It can be expressed as follows:

$\forall$ $Attr_k \in ATTR$, $0 \le k \le z$, z is the cardinality of ATTR

If $Attr_k \in r_i(Attr)$ and $Attr_k$ is an ObA of r_i and

If $r_i \times r_j \neq \Phi \ \& \ i = j$ Then $Attr_k$ is visible to r_j

Other $Attr_k$ is never visible to r_j

4.2.5 Role conformation

Based on four role types, a role can be formed by only CA or CA as well as OA. Any CA can be ObA or HA. Any OA can also be ObA or HA. The role formation is shown at Figure 1.

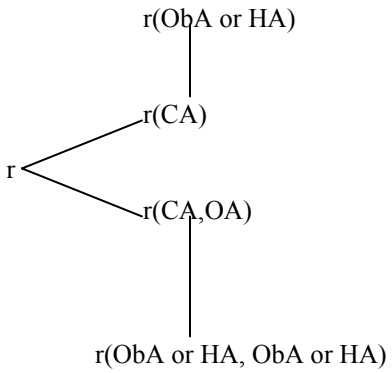

Figure 1 Role conformation

4.2.6 Usage of role types

We use a university hierarchy, shown in Figure 2, to demonstrate the usage of CA and OA. In Figure 2, we have the following roles: university role, department role, academics role, administration role, research role, teaching role, etc. At the top of the hierarchy, the university role has the attributes: University ID, students, faculty, company management. It is clear that the attributes of University ID, students and faculty are CA. If the university has no university ID, students, faculty members, there is no way to have the university. There is an attribute of university called Company which means the university has a function to manage a company. It is clear that Company can be an OA because if the university does not have a function to run a company, the university can still be called "university". All other attributes displayed in Figure 2 are obviously CA. Of course here we can add some OA for all the roles for the extra duties or functions.

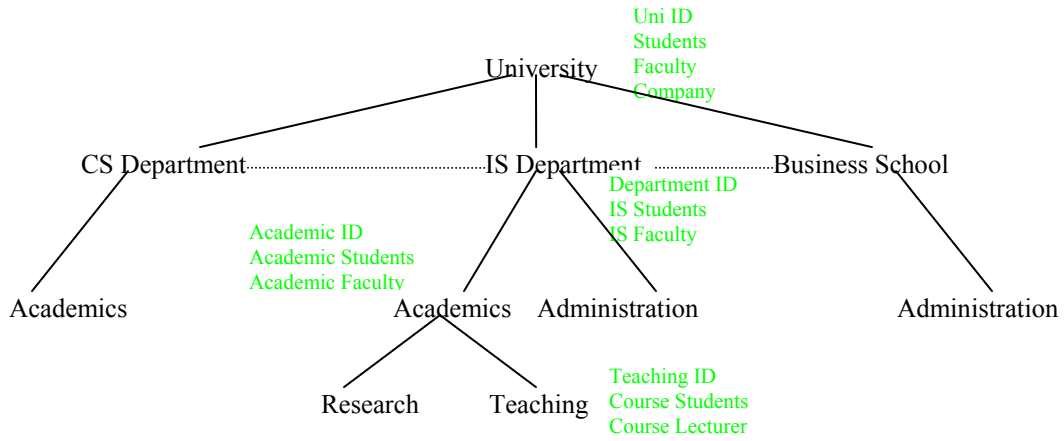

Figure 2 A part of a university hierarchy

We use a part of a college leadership hierarchy [Figure 3] to demonstrate the usage of ObA and HA. In Figure 3, there are different roles in a college, Dean, Head of Department, Research leader, Teaching leader, project researcher, course lecturer, etc. The Course lecturer at the bottom of the hierarchy has the role attributes: Report to teaching leader, Take course duties and student administration. Because the course leader is supervisor of the course lecturer, if necessary, the teaching leader can inherit the role of the course lecturer. As a concern of the teaching leader' workload or duty assignment, the teaching leader can only inherit a part of duty of the course lecturer. For an example, the teaching leader can only take the load of student administration from the course lecturer. We can set the attribute of student administration as ObA and set two other attributes, report to the head and take course duties, as HA. The same mechanism can be applied to dean and head of department, head of department and teaching leader. Through ObA and HA, we can have a partial heritance for hierarchical roles. So far to the best of our knowledge none literatures contribute to the partial heritance in RBAC research. If all the attributes of a role are ObA, any inheritance will accept all attributes of the role. If the attributes of a role are HA, any these attributes are not inheritable to its descendants. Furthermore, well-defined HA can be used to keep the privacy of a role.

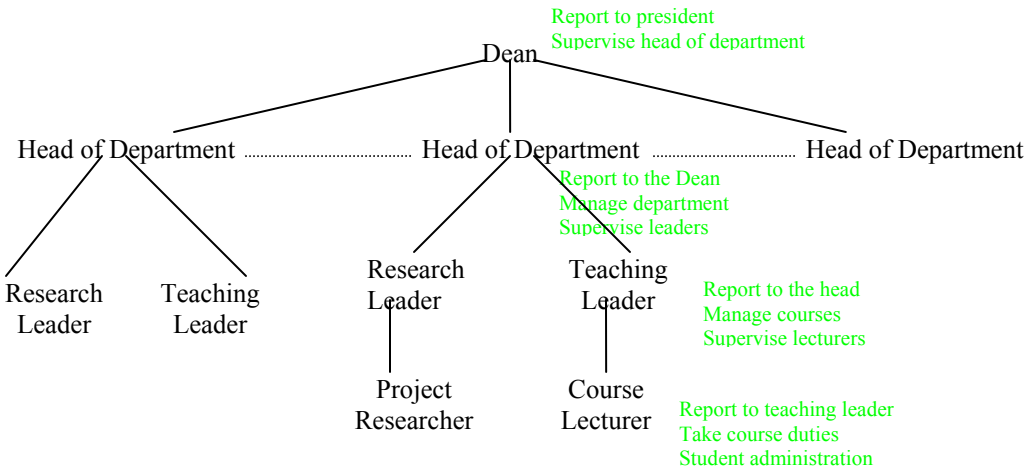

Figure 3 A part of a college leadership hierarchy

4.3 Attributes Operation

Because any role consists of its attributes, each attribute can be assigned to different types. In order to effectively use the role attributes to extend current RBAC, it is essential to have fundamental operations on the attributes. These basic operations include "add an attribute", "delete an attribute", "update an attribute", "suspend an attribute", and "change the type of an attribute".

4.3.1ADD

As a role consists of a set of attributes, any system needs a function to add an attribute into an existing role if it is needed. Now we define the operation of ADD for an attribute.

ADDattribute (Attr: r)
If Attr $\notin$ r & Attr $\in$ *ATTR*
Then r=r $\bigcup$ {Attr}

4.3.2 DELETE

With the time going, some systems need a function to remove an attribute from an existing role if it is needed. Now we define the operation of DELETE for an attribute.

DELETEattribute(Attr:r)
If Attr $\in$ *r* & Attr $\in$ *ATTR*
Then r=r\{Attr}

4.3.3 UPDATE

When a role is defined with its attributes by the system, after a while, some attributes of the role need updates from time to time. This requires a function to operate on the attributes to update any attributes which need changes. The UPDATE operation is defined as the follows.

UPDATEattribute(Attr:r)
If Attr$\in r$ & Attr$\in ATTR$ & New Attr$\neq$ Attr
Then r=r\{Attr} & Attr=New Attr
Then r=r$\cup$ {Attr}

4.3.4 SUSPEND

Some attributes of a role can be no use for a time period. After a certain time, those attributes should have previous function. For an example, an attribute is assigned to check the log of DBMS. If DMBS needs maintenance for several days, during the period, this attribute does not need work. It is not necessary to delete this attribute. Here we introduce an operation, SUSPEND. It is defined as the follows.

SUSPENDattribute(Attr:r)
If Attr$\in r$ & Attr$\in ATTR$
While Attr unavailable Then r=r\{Attr}
Else r=r$\cup$ {Attr}

4.3.5 TPYE-CHANGE

As we discussed previously, there are four different types of attributes, CA, OA, ObA, and HA. At different scenarios, the same attribute might need a different type to match the required function of a role. In order to make that happen effectively, we need an operation, TPYE-CHANGE, as the follows.

TYPE-CHANGE(Attr.type:r)
If Attr$\in r$ & Attr$\in ATTR$
If Attr.type$\neq$ requested type
Then Attr.type=requested type

5 Attributes and Object, Users, Permissions and Roles

The attributes become the extended feature of RBAC. Traditional RBAC needs a new consideration of its mechanism. The core BBAC has a categorization and definition of role, user, permission, and object. There are many literatures about the co-relationships of these four entities [more references here]. We do not address more over these traditional relationships here. Because the attributes are very important components of any role, it is necessary to redraw the co-relationships of attribute and role, attribute and user, attribute and object.

5.1 Attributes and Roles

Any role consists of its attributes. While a role is assigned and defined in a system, automatically a series of attributes are bound to the role. Any attributes of a role will be set in two different types according to their requirements.

$$r \in R$$
$$R \subseteq ATTR \times ATTR \times Attr.tpye\{T1\} \times Attr.type\{T2\}$$

$$T1 = \{CA, OA\}; T2 = \{ObA, HA\}$$

Through this extension, any roles in RBAC systems have their own attributes which can be assigned into suitable types. Through this improvement, the role has much more flexibility and operability by using the attributes.

5.2 Attributes and Users

The attributes have no direct relationships with users. All the functions of attributes have to be embedded in a role or roles.

$$Users \times R$$

A user can have one or many roles. A role can be assigned to one or many users. Users never concern the attributes of a role.

5.3 Attributes and Permissions

Traditional RBAC has a direct relationship between roles and permissions. Usually $Permissions \times Roles$ means a many-to-many mapping permission-to-role assignment. Now we have a series of attributes to represent a role. It is useful to combine the attributes directly with the permissions. When an operation on the attributes is done, the role function has some changes accordingly. If the attributes are directly combined with permissions, any operation on attributes will be reflected back to permission changes. That makes permission-to-role assignment much more dynamic and applicable.

$AP \subseteq Permissions \times Attributes$ a many-to-many mapping permission-to-attribute

In order to simplify the management operation, none of any permission or any combination of permissions is allowed to be assigned to an attribute of any role more than once. This can ensure that any different attributes of a role have a different permission or permission combination. It is obvious that an operation on attributes will automatically reflect the permission change to a role. For example, an attribute of a role is assigned to look after an FTP server. Actually the permission to operate on the FTP server is combined with this attribute. If a DELETE operation is executed on this attribute, it means the permission to operate on the FTP server is removed from the role as the attribute is deleted. In the same way, an ADD operation on a role leads to the role having an extra permission or its combination as an attribute is added in. An UPDATE on an attribute leads to update the permission associated with the updated attribute. A SUSPEND operation on an attribute leads to temporarily stop a related permission or its combination. A TYPE-CHANGE operation on an attribute affects related permission or its combination as compulsory or optional, obvious or hidden to a role. It is obvious that any operation on the attributes directly impacts on the function of a role and its associated permission or combinations.

5.4 Attributes and Objects

The attributes do not have directly relationship with objects. Any interactions between attributes and objects have to pass the bridge of a role like the following.

$$Attributes \rightarrow\rightarrow Roles \rightarrow\rightarrow Objects$$

Also objects can be directly mapped onto permissions or their combinations, users, as well as roles.

6 Attributes and ANSI RBAC Framework

The role is a fundamental component of RBAC. We have introduced the attributes for the roles and all the components of RBAC are impacted by the attributes in a different degree. It is important to discuss how the attributes impact ANSI RBAC framework. Figure 4 shows the ANSI RBAC framework.

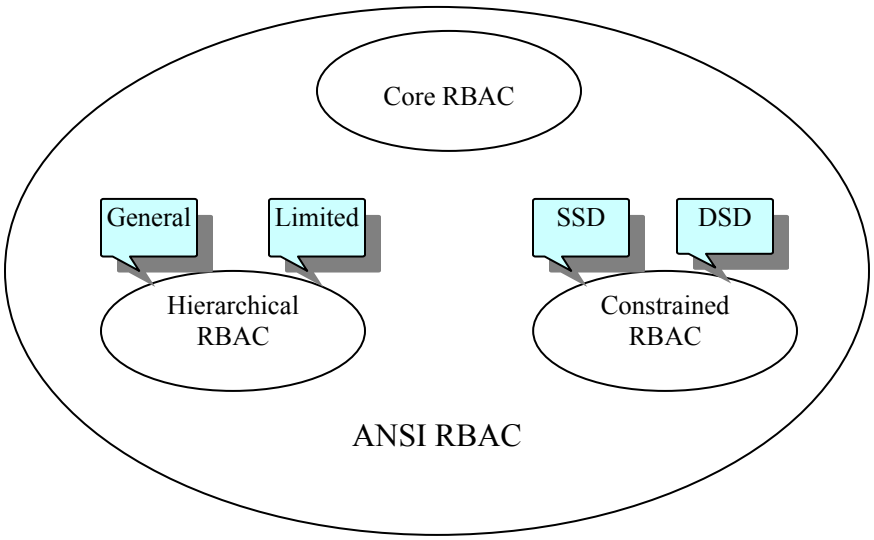

Figure 4 ANSI RBAC Framework

6.1 Attributes and Core RBAC

Table 1 shows the fundamental difference between ANSI Core RBAC and Attribute-extended core RBAC.

	ANSI Core RBAC	Attribute-extended Core RBAC
Entities	Users Roles Permissions	Users Roles Permissions *Attributes*
Relationships	*USERS* × *ROLES* *ROLES* × *PERMISSIONS*	*USERS* × *ROLES* *ROLES* × *ATTRIBUTES* *ATTRIBUTES* × *PERMISSIONS*
Operations on entities	AddUser DeleteUser	AddUser DeleteUser

	AddRole DeleteRole AddPermission DeletePermission	AddRole DeleteRole AddPermission DeletePermission *AddAttribute* *DeleteAttribute* *UpdateAttrbiute* *SuspendAttribute* *Type-ChangeAttribute*

Table 1 ANSI Core RBAC and Attribute-extended Core RBAC

6.2 Attributes and Hierarchical RBAC

Table 2 shows the differences of ANSI Hierarchical RBAC and Attribute-extended Hierarchical RBAC.

	ANSI Hierarchical RBAC	Attribute-extended Hierarchical RBAC
General Role Hierarchies and Limited Role Hierarchies	$Roles \times Roles$ is irreflexive and acycle A partial order is reflexive and transitive closure of $Roles \times Roles$	No difference as ANSI Hierarchical RBAC
User Inheritance	Any User gets a role and also get this role's parental authorities	Any User gets a role and gets only ObAs of its parents.
Permission Inheritance	A role *r* is authorized for all permissions for what *r'* is authorized where *r* inherits *r'*	Only ObAs of *r'* are inheritable. HAs of *r'* are not inheritable.
Activation Inheritance	Activating a role *r* automatically activates another role *r'* where *r* inherits *r'*	Only ObAs of *r'* will have active functions. HAs have non-function to *r.*
Role Privacy on Hierarchies	If there is an inheritance from *r'* to *r*, *r'* has no privacy to *r.*	If there is an inheritance from *r'* to *r*, *r'* has its privacy to *r* if *r'* has some HAs.

Table 2 Comparison between ANSI Hierarchical RBAC and Attribute-extended Hierarchical RBAC

6.3 Attributes and Constrained RBAC

Table 3 shows the comparison between ANSI Constrained RBAC and Attribute-extended Constrained RBAC.

	ANSI Constrained RBAC	Attribute-extended Constrained RBAC
Static Constraint	This constraint limits the roles which can be assigned to a user.	For this constraint at the role level, it is the same. But a further level about attributes needs further exploring and defining. To our best knowledge, there is no further research on this aspect. We will do more work on that later.
Dynamic Constraints	This constraint limits the roles which a user can activate in a session.	Similarly here a further level of attributes needs exploring and defining. So far there is no literature about this extension. We will do more work on that in the future

Table 3 Differences between ANSI Constrained RBAC and Attribute-extended Constrained RBAC

7 New RBAC Framework with Attributes

Figure 5 shows the relationships between the attributes and ANSI framework. The attributes have extended the functions of the role. The core RBAC, hierarchical BAAC and constrained RBAC are all impacted by the attributes. More details have been discussed in previous sections.

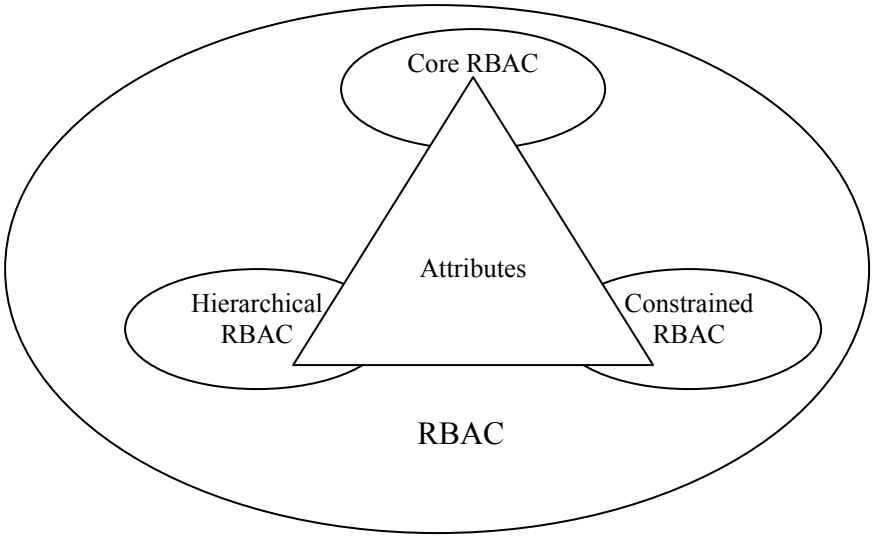

Figure 5 Attribute-extended RBAC Framework

8 Conclusion Remarks and Future Works

We have worked out a new entity, attribute, for ANSI RBAC. The attributes significantly extend the function of the roles in a system into a new level. The attributes have extended the operations, definitions and transactions in all three components of ANSI RBAC, namely core RBAC, hierarchical RBAC, and constrained RBAC. The flexibility and privacy of a role can be effectively implemented by different attribute types. This paper has a significant new idea to contribute the ANSI RBAC framework by adapting the attributes. More impact evaluation is needed to be done to the attributes in the future. More research will be carried out on the relationships and transactions between the attributes and hierarchical RBAC as well as constrained RBAC.

Acknowledgement

The paper is also partially supported by USQ's early career research grant, the investigation and evaluation of the usage and roles of firewalls to organisational Intranets.

References

ANSI, *American National Standard for Information Technology* – Role Based Access Control, ANSI INCITS 359-2004, February 2004.

Byun, J. W., Bertino, E., and Li, N. Purpose Based Access Control of Complex Data for Privacy Proctection, *In Proceedings of the Tenth ACM Symposium on Access Control Models and Technologies*, pages 102-110, June 2005, Stockholm, Sweden.

Covington, M. J., Long, W., Srinivasan, S., Dey A. K., Ahamed, M. and Abowd, G. D. *Securing Context-Aware Applications Using Environment Roles, In Proceedings of the Sixth ACM Symposium on Access Control Models and Technologies*, pages 10-20, May 2003, Chantilly, Virginia, USA.

Ferraiolo, D. F., Sandhu, R. S., Gavrila, S., Kuhn, D. R. and Chandramouli, R. Proposed NIST standard for role-based access control, *ACM Transactions on Information and Systems Security*, 4(3):224-274, August 2001.

Giuri, L. and Iglio, P. Rold Templates for Content-Based Access Control, *In Proceedings of the second ACM Workshop on Role-Based Access Control*, pages 153-159, 1997, Fairfax, Virginia, USA.

Sandhu, R. S., Coyne, E. J., Feinstein, H. L., and Youman, C. E. Role-Based Access Control Models, *IEEE Computer*, 29 (2): 38-47, February 1996.