International Journal of Engineering Applied Sciences and Technology, 2020
Vol. 5, Issue 3, ISSN No. 2455-2143, Pages 295-302
Published Online July 2020 in IJEAST (http://www.ijeast.com)

MATHEMATICAL MODELLING OF DELEGATION OF ROLES IN RBAC

Harendra Subedi[123*], Iskra Popova[2], Silvio Ranise[3]
[1].Royal Institute of Technology, KTH
[2].Department of Computer and Systems Sciences (DSV), Stockholm University
[3].Security and Trust Unit, FBK, Trento, Italy

Abstract - Role Base Access Control (RBAC) and its components have been researched in multiple levels. The research explains and exemplifies by creating mathematical models describing the different aspects of RBAC's administrative issues. However, the issues regarding formalization (Mathematical Modelling) of delegation and revocation of roles in RBAC could be relatively new area to research.

Undoubtedly, this research offers an important extension of the policy and it delivers flexibility in the user to user delegation of roles, especially in the environment where roles are organized in a hierarchy. The process allows a user with a role that is higher in the hierarchy to assign a full or part of the role to someone who is lower in the hierarchy or at the same level. Interestingly, this process consists of time springiness depending on the choice whether for a limited time or for permanently.

This paper aims at providing different type of delegation and techniques with a comprehensive mathematical Modelling of the processes. Obviously, the objective is to derive a mathematical model for delegation roles in RBAC policy, for deriving mathematical models' formal method is applied. The mathematical models developed include grant and transfer delegation with and without role hierarchy. The organization using RBAC has been considered as a case in point to illustrate and clarify the mathematical models. The mathematical models presented here can serve as a starting point for developing, implementations of delegation of roles on top of existing authorization modules based on the RBAC model.

Keywords: *Delegation; RBAC; Role; Role Hierarchy; Formal Methods*

I. INTRODUCTION

Definition of RBAC Policy

RBAC is the access control policy where the user can access the resources on the basis of its role rather than its identity. Let us consider U as the set of users, R as the set of roles, P as the set of permissions, UA as the set of user assignments, PA as the set of permission assignments, and $\geq$ as the role hierarchy. We can define UA (User Assignment) relation as the subset of $U \times R$ (User × Role), PA (Permission Assignment) relation as the subset of $R \times P$ (Role × Permission) and $\geq$ (Role hierarchy) as the subset of $R \times R$ (Role × Role) which is also a partial order. In RBAC, by joining the relations UA and PA, it is possible to derive the relation associating user(s) to permission(s). Formally from [1] [2] [3] [4] [5] [6], roles in a set R associate permissions in a set P to users in a set U by using the following two relations: $UA \subseteq U \times R$ and $PA \subseteq R \times P$. Roles are structured hierarchically so as to permit permission inheritance [7]. A role hierarchy is a partial order $\geq$ on R, where $r_1 \geq r$ means that r_1 is more senior than r_2 for $r_1, r_2 \in R$. A user u is an explicit member of role r in UA when $(u,r) \in UA$ while u is an implicit member of r in UA if there exists $r' \in R$ such that $r' \geq r$ and $(u,r') \in UA$. Given UA and PA, a user u has permission p if there exists a role $r \in R$ such that $(p, r) \in PA$ and u is a member of r. Now, we can consider the tuple $< U, R, P, PA, \geq, UA >$ as an RBAC policy, and this tuple will be extended when more complex delegation models will be considered [6] [8] [9].

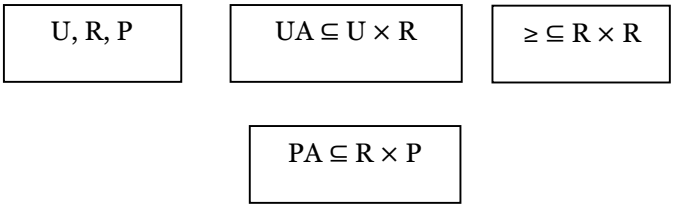

Figure 1:- Different Components of the RBAC Policy

Figure 1 shows the different components of the RBAC policy.

Properties of RBAC Policy

The number of relations defining the RBAC system for big organizations can be very large. When day-to-day tasks change within the organization, modification of too many relations might be required at the same time. This can make the management and administrating of RBAC to be cumbersome and problematic. In order to make the

International Journal of Engineering Applied Sciences and Technology, 2020
Vol. 5, Issue 3, ISSN No. 2455-2143, Pages 295-302
Published Online July 2020 in IJEAST (http://www.ijeast.com)

administration / handling of RBAC policy simple, it is desirable to allow for the user assignment relationship to be dynamic and leave the rest of the relations static. This can be done because the set of users, the set of roles, set of permissions, the permission assignment relation, and the role hierarchy of an organization does not change frequently.

If there is a huge change in the organizational structure then only these relations are likely to be changed, so it is recommended to keep them static once the security manager or the system administrator has defined them. The only relation that changes during daily operation is user assignment relation. If a new three user comes to the organization then, he/she can be assigned a role in UA. Once the new user is associated to the role, then he/she will get the permissions associated to that role which is defined in the permission assignment relation PA.

It is very unlikely for new roles to be created in an organization frequently or the role hierarchy of an organization to change very often. New roles in an organization are created, if there is a change in role hierarchy or if there is a change in the organizational structure or if there is a massive reorganization within the organization. Thus, it is reasonable to assume that the only relation, that is likely to change frequently, is UA whereas the other relations are static and they do not change frequently.

While describing the evolution of RBAC systems, one can consider the UA relation as the RBAC policy since all the other relations are static. As a consequence, questions such as if the user u has the permission p can be rephrased in terms of roles only, i.e. "Does the user u can have role r?" From now on, we consider the problem of establishing whether a user u can become a member of role r rather than if u can get permission p.

There are two types of user actions that can be performed by the user which change the state of the RBAC system: assigning a role to a user (delegation) and revoking the role from a user (revocation). If the user can perform these actions without any constraint, it would be difficult to foresee all the implications of the concurrent execution of several delegations by many users. For this reason, rules are specified in order to constrain delegation and revocation to make it possible to understand the implications of a sequence of delegations and revocations.

Example Scenario /Case

Throughout this paper, we will use the following scenario to illustrate our ideas. The presented scenario is totally imaginary and we believe that this scenario is not only simple to understand but also will cover all the possible cases of delegation and revocation of roles in RBAC.

In an organization there are lots of people working in different departments. Each department has its own functionalities. Each person working in the department is associated with certain role(s). If we consider the organizational structure of Research lab as an example, we can see that there are lots

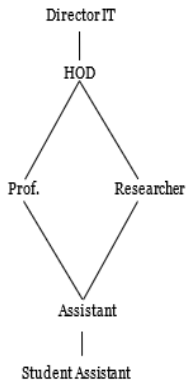

Figure 2.1:- Role Hierarchy

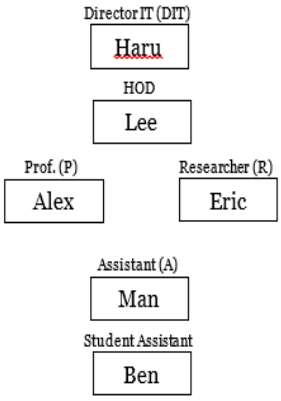

Figure 2.2:- Role Hierarchy of user with role membership

Figure 2 :- Role Hierarchy and Its user of a Digital forensic department

of departments for specific tasks, and each department of the research lab has its own role hierarchy. There might be different roles in the Research lab, such as board of director, chief executive officer, managing director, director, head of departments (HOD) etc. There is also the possibility of having various departments such as administration, IT, finance etc. It will be difficult to describe all organization structure and its role hierarchy. Therefore, we decided to pick one department called Digital forensics, in order to simplify our task of explaining the relationship between the roles in the role hierarchy. We will try to use the example of this department in the entire text that follows to explain different mathematical models. Figure 2.1 shows the role hierarchy and Figure 2.2 shows the user and its role membership in the role hierarchy of the digital forensics department. DIT (Director of IT) is the role which is considered as the highest senior role in the IT department and its associated user is Haru. Haru is accountable for entire IT department of the research lab and he is liable to control all the other departments. However, in our case we are only focusing on Digital forensics department. In this department, we have different roles, such as HOD, Prof. (P), Researcher (R), Assistant (A) and Student Assistant (SA). In the digital forensic department, Lee is associated with the role of HOD, Alex with the role Prof., Eric with the role Researcher, Man with the role Assistant and Ben with the role Student Assistant. If we only consider the role hierarchy of

International Journal of Engineering Applied Sciences and Technology, 2020
Vol. 5, Issue 3, ISSN No. 2455-2143, Pages 295-302
Published Online July 2020 in IJEAST (http://www.ijeast.com)

the digital forensic department, then we can conclude that HOD is the highest senior role and the student assistant is the lowest junior role. The role HOD inherits the role and permission of its junior role(s) [7].

Any user, who has a role, can delegate its role or a junior role to another user within or outside of role hierarchy, temporarily or permanently. For example, Lee can delegate HOD to Alex, Man, Eric etc., or Lee can delegate his HOD to another member Jen from different role hierarchy very easily.

Definition of delegation

Administrative delegation in RBAC enables administrators to delegate the role and associated permissions to another user on behalf of the owner of the role. In user to user delegation, the user associated to the particular role can delegate its role or some junior roles to another user. Sometimes the delegation of the role does not follow the role hierarchy in the $RBAC_1$ which means that, the delegation occurs within the same level between the users of two non-comparable roles. The non-comparable roles are those roles which cannot be compared with each other with respect to the role hierarchy.

In ARBAC [1] [10], there is an administrator, who takes care of delegation and revocation of roles on behalf of the users. Instead for user to user delegation, the delegating user of the role acts as an administrator for that particular delegation. In this way additional human intervention such as the effort of administrators to delegate or to revoke the role on behalf of a user is not required. However, for some form of revocation, (see Paper modelling of Revocation of Permanent Delegation) some administrative effort is needed.

A user must follow certain rules to delegate or to receive delegation. The basic rule is that a user can delegate its role or some junior roles if the user to whom he is willing to delegate satisfies certain conditions. The effect of performing a delegation is an update of the UA relationship whereby the new association between the delegated role and the user to which the delegation has been given is added to UA. Formally, we can see UA as $UA = UA_0 \cup UD$ (Formally, UD relation can also be defined as $UD \subseteq U \times R$.) where UA_0 is static (where user role assignments are given by the system) and UD is dynamic (where user role assignments are established via delegations among users) [6] [7] [8] [11] [12].

Formally, the rules to perform a delegation are specified as tuples of the relation *can_delegate* such that $can\_delegate \subseteq R \times pre$, where *pre* is the set of preconditions (to be defined below). *Can_delegate* is used to constrain the freedom of users to delegate roles in order to avoid security problems. *Pre* is the set of preconditions, which is used to identify candidate users to receive the delegation. This can be made precise in the following way: - *Pre (Precondition)* can be defined as a set of signed roles, i.e. $\pm r$ where $r \in R$. The meaning of signed roles is that the role can be represented either by $+r$ or $-r$ and is used for checking the precondion. If the signed role is $+r$ means that the user u is a member of role r in *UA* and if it is $-r$ means, that user u is a not a member of role r in *UA*. The precondition acts as the condition that has to be satisfied by the user in order to receive the delegation. Preconditions can be very complex. Example of complex preconditions can be considered as if the user u_2 is a member of the role r_1, r_2, r_3, and if the user u_2 is not a member of the role r_4, r_5, r_6 etc., and then only the user u_2 can delegate the role to another user say u_3. This kind of complex preconditions can be constructed by a conjunction of signed roles such as being an explicit or implicit member of certain roles and not being an implicit or explicit member of certain roles. The precondition described in this paper can be very sophisticated, but for reason of simplicity in our examples, we use a single signed role.

The intuition underlying $(r, pre) \in can\_delegate$ is that a user u` should be a member of role r to be able to delegate it or some junior roles of r to a user u satisfying the precondition *pre*.

Formally, it is possible to define once and for all; the notion of a user u satisfies a precondition

$pre = \{[exp_1, exp_2 \ldots, exp_n]\}$ with respect to UA_0 as follows

for each expression *exp*

- of the form $+r$, we have
  User u is a member of role r in UA_0
- of the form $-r$, we have
  User u is not a member of role r in UA_0

It is clear that "being a member of" encapsulates both the case where a role hierarchy is used and where it is not. In this way, the preconditions for $RBAC_0$ do not consider the role hierarchy whereas those for $RBAC_1$ do so.

The formalization of the effect of the execution of the delegation on the state of the RBAC system (that can be thought of the dynamic relation *UD*) depends on the type of delegation and revocation that one considers and will be the main subject of the rest of the paper.

In particular, we will provide mathematical framework (based on simple set theory) to specify the various types of delegation and revocation of

International Journal of Engineering Applied Sciences and Technology, 2020
Vol. 5, Issue 3, ISSN No. 2455-2143, Pages 295-302
Published Online July 2020 in IJEAST (http://www.ijeast.com)

roles in RBAC considered in E. Barka's PhD Paper [9].

Meaning of the above precondition is that the user *u`* who is delegating the role *r`* should be either implicit or explicit member of *r`* such that the users receiving the delegation will either have non-comparable roles to *r`* or have junior role than *r`* with respect to the role hierarchy. In [9], it is required that a role *r* that is being delegated to a user *u* should be non-comparable or senior (with respect to the role hierarchy) to the role of *u*. Thus, in the rest of this paper, we assume that this check is preformed every time a delegation is going to be performed and if the condition is violated, then the delegation is considered not to be enabled. Formally, this is expressed as follows:

> Given $(r, pre) \in can\_delegate$,
> a user *u* satisfies *pre* is defined as before with the additional condition:
> if $(u, r_u) \in UA$
> then *(neither* $r_u \geq r$ nor $r \geq r_u$*)* this mean r_u and *r* are not comparable with respect to role hierarchy
> or $(r_u \geq r$ and $r_u \neq r)$ this means that r_u is more senior that *r* with respect to the role hierarchy

II. MODELLING OF GRANT/TEMPORARY DELEGATION

In grant delegation (also known as temporary delegation), roles are delegated to other users according to some rules but the delegator maintains his membership in the delegated role. The delegator holds full responsibility for the role that has been delegated [2] [3] [13] [14]. The delegation can either be total or partial. The term total and partial in delegation defines how many permissions associated to the role have been delegated. To be precise, total delegation means that all the permissions that are associated to the role have been delegated, whereas partial delegation means that only the subset of the permissions which is associated to the role has been delegated. The receiver of the delegation is constantly monitored by the user who has delegated the role. The delegator of the role bears the full responsibility on behalf of the receiver; this means that the delegator will be responsible and accountable for the role that it has delegated even if the receiver does something wrong. Grant delegation in RBAC96 comes in two flavours: delegation within flat roles ($RBAC_0$) and delegation within role hierarchy ($RBAC_1$) [3] [9] [13] [15] [16]. The following sections deal with the different models of grant delegation in $RBAC_0$ and $RBAC_1$.

III. MODELLING OF ONE STEP GRANT DELEGATION IN FLAT ROLE ($RBAC_0$)

This subsection deals with one-step grant delegation in $RBAC_0$, i.e. the roles that have been delegated once cannot be delegated further. In other words, the delegatee cannot further delegate the role that he has received by delegation from another user. Delegation in this model is total, which means that, while delegating, the delegator either delegates all the permissions that are associated with the role or does not delegate any of them. Delegation between the users of the same role is also not allowed [5] [9] [14], since it is obviously useless. The following example can be taken into consideration from the scenario which is described in previous chapter of this paper. Let us assume that Lee is associated to HOD role in Digital forensic department and is a very busy person, and most of the time he is out for conferences and seminars. Now for two weeks, Lee is going for a conference which is to be held in Sydney, but he has few tasks pending and they are supposed to be due within three weeks. It is likely that Lee will not finish his tasks within the stipulated period of time. So, before going to Sydney, Lee decides to delegate his tasks to Maddy who is associated to a role researcher in the criminology department. In this model, there is no presence of role hierarchy, so any user having the role can delegate its role to any other user within an organization.

Formally the Modelling of grant delegation in flat RBAC ($RBAC_0$) can be defined as follows

1. Define the *UA* relation as $UA = UA_0 \cup UD_1$ where UA_0 is static and UD_1 is dynamic.
2. The tuples in *can_delegate* allows us to define a transition relation over UD_1 Modelling the effect of delegation on *UA* as follows: -
    a. If $(r, pre) \in can\_delegate$ and *u`* is a member of *r* and *u* satisfies pre (with respect to UA_0)
    b. Then $UD_1` = UD_1 \cup \{(u, r)\}$
3. We say that user *u`* delegates the role *r`* to the user *u* (who satisfies pre) and write $UD_1 \rightarrow UD_1`$ to denote the transition induced by the delegating action.

IV. MODELLING OF ONE STEP GRANT DELEGATION IN ROLE HIERARCHY ($RBAC_1$)

In this subsection we will discuss about the one step delegation in the role hierarchy or the one step delegation in $RBAC_1$. This delegation model is almost the same as the one step delegation for flat model ($RBAC_0$). The main difference is that the delegation is performed in the presence of a role hierarchy. The basic idea behind this delegation is that the user associated to the role can delegate its

International Journal of Engineering Applied Sciences and Technology, 2020
Vol. 5, Issue 3, ISSN No. 2455-2143, Pages 295-302
Published Online July 2020 in IJEAST (http://www.ijeast.com)

role or some junior role to the user who is associated with the role which is junior to the role that is being delegated or to the user who satisfies a certain precondition. This delegation takes place within the role hierarchy and the delegation can be either downwards or across [9]. The roles that have been delegated in one step delegation cannot be further delegated. The nature of the delegation can either be partial or full depending upon the fact that the role itself or some junior roles is delegated. Now, let us consider the following example to illustrate the one step delegation in role hierarchy. Alex, Prof. by the role, is going to be father for the first time; he is very excited and is thinking of taking few days off from the office so as to be with his wife for their first child. Alex has lots of work to finish, and he has also not finished his courses, and it will take seven more lectures to finish his course and it is very unlikely that he will finish the course after becoming father. So, Alex decides to delegate part of prof. role (teaching) to Eric who has the role R (Researcher). The roles of Alex and Eric are non-comparable since they are at the same level in the role hierarchy. The type of delegation that occurred in this case is one step cross delegation. In one step delegation, Eric cannot further delegate the role that he has received from Alex.

Following is the mathematical Modelling of one step grant delegation in Role Hierarchy ($RBAC_1$)

1. Define *UA* relation as $UA= UA_0 \cup UD_1$ where UA_0 is static and UD_1 is dynamic.
2. The tuples in *can_delegate* allows us to define a transition relation over UD_1 Modelling the effect of delegation on *UA* as follows: -
    a. If $(r, pre) \in can\_delegate$ and $u`$ is a member of r and u satisfies *pre* (with respect to UA_0)
    b. Then $UD_1`=UD_1 \cup \{(u, r`)\}$ for some $r`$ which is junior or equal to r with respect to role hierarchy.
3. We say that user $u`$ delegates the role $r`$ to the user u (who satisfies the pre) and we write $UD_1 \rightarrow UD_1`$ to denote the transition induced by the delegating action.

V. MODELLING OF TWO STEP GRANT DELEGATION IN ROLE HIERARCHY

This model is an extension of the one step delegation model in the role hierarchy, with more flexibility in that the role received from a delegation can be further delegated to another user, but the user receiving the second delegation cannot delegate that role further. So, this means that a role can be delegated twice within the role hierarchy but it cannot be delegated more than twice. This type of delegation can be partial or full and the delegation should take place either between two non-comparable roles or downwards in the role hierarchy. For example, let us continue the example considered for one step delegation with role hierarchy. We assume that the first delegation is as explained above. After few days, Eric realizes that he is not able to work properly due to extra work load, so he decides to further delegate his role prof. (teaching) obtained from Alex to Man who has role A and which is one step below in the role hierarchy with respect to Alex and Eric's roles. This kind of delegation is an example of the two-step delegation. In the following Modelling we can see that the first part of this model is identical to the one step delegation with the only difference in the second part of the delegation. More precisely, in place of checking in if a user satisfies a precondition with respect to *UA*, the user delegating the role does so with respect to in UD_1 (User Delegation 1) to delegate the role. After performing the delegation, the user delegating the role will make necessary update in UD_2 (User Delegation 2) by adding the new user to role assign to UD_2. Let us consider the above example once more. When Eric wants to perform the delegation of the role (teaching) to MAN which he has received from Alex, he checks the UD_1 relationship at first. Upon performing the delegation, by Eric to Man, Eric will update the UD_2 relation.

In the following you can find the mathematical Modelling of two step delegation within the Role Hierarchy.

1. We can define the *UA* relation as $UA= UA_0 \cup UD_1 \cup UD_2$ where UA_0 relation is the static part of *UA* and UD_1 and UD_2 relation are the dynamic parts.
2. The tuples in *can_delegate* allow us to define a transition relation as follows: -
    a. If $(r, pre) \in can\_delegate$ and $u`$ is a member of role r and u satisfies *pre* (with respect to UA_0)
    b. Then $UD_1`=UD_1 \cup \{(u, r`)\}$ and $UD_2`= UD_2$ for some $r`$ which is junior or equal to r with respect to the role hierarchy.
    c. If $(r, pre) \in can\_delegate$ and $u`$ is a member of role in UD_1 and user u satisfies *pre* (with respect to UD_1)
    d. Then $UD_2`=UD_2 \cup \{(u, r`)\}$ and $UD_1`= UD_1$ for some $r`$ which is junior or equal to r with respect to the role hierarchy.
3. We write $(UD_1, UD_2) \rightarrow (UD_1`, UD_2`)$ to denote the transition induced by the two-step delegation action.

Modelling of K Steps Delegation in the Role Hierarchy

International Journal of Engineering Applied Sciences and Technology, 2020
Vol. 5, Issue 3, ISSN No. 2455-2143, Pages 295-302
Published Online July 2020 in IJEAST (http://www.ijeast.com)

This model is a further generalization of the previous delegation model, which enables users to delegate the role for K steps. In the above-mentioned model, a user can delegate its associated role to another user either once or twice. In this model, we allow for K step delegation of a role where K is a natural number greater than or equal to 1. Following, you can find the Modelling of K^{th} step delegation in the role hierarchy

1. Let $K \in \mathbb{N}$ some fixed value
2. Let $UA = \bigcup_{j=0}^{K} UD_0$ where $UD_0 = UA_0$

   is the static relation and UD_1, $UD_2, \ldots, UD_K$, are the dynamic relation.
3. If $(r, pre) \in can\_delegate$
   For some $i \in \{1, 2, \ldots, K-1\}$ we have
   User $u`$ is a member of role r in UD_i
   User u satisfies pre with respect to UD_{i-1}
   Then $UD_j` = UD_j$ for $j=1, 2, \ldots, k-1$ and $j \neq i$
   $UD_i` = UD_i \cup \{(u, r`)\}$ for some $r`$ which is junior or equal to r with respect to role hierarchy.
4. We write $(UD_0, \ldots, UD_i, \ldots, UD_k) \rightarrow (UD_0`, \ldots, UD_i`, \ldots, UD_k`)$ to denote the transition induced by a k-step delegation

Modelling of Transfer/Permanent Delegation

If the delegator of a role loses his membership in the delegated role, then this kind of the delegation is considered as transfer delegation or permanent delegation. In the transfer delegation, both parties involved in delegation, that is the delegator and the receiver of the delegation should agree on it. After performing the transfer delegation, the delegator has no right and holds no responsibility with respect to the role that has been delegated. It means that, after performing transfer delegation, the delegator will lose its membership and ownership in that particular role. After receiving the delegation, the receiver will get the membership and act as an original member of that role. Transfer delegation should be total, which means that, the delegator should delegate the whole set of permissions associated to a role. Transfer delegation in RBAC comes in two flavours, they are delegation within the flat roles ($RBAC_0$) and delegation in role hierarchy ($RBAC_1$) [9] [13] [15]. Most importantly, delegation between users of the same role is not allowed and also considered as useless [9]. The following subsection deals with Modelling transfer delegation of roles in $RBAC_0$ and $RBAC_1$.

Transfer Delegation in $RBAC_0$ or $RBAC_1$

The basic idea behind transfer delegation in $RBAC_0$ or $RBAC_1$ is that any user associated with the role can transfer its role to any user provided some preconditions are satisfied. This type of permanent delegation should be total in nature i.e. delegator either delegates all of the permission associated with a role to the new member or it does not delegate any of them [9]. Let us consider the following example to illustrate the transfer delegation. Lee is associated with the role HOD in Digital forensic department and he is also associated with Advisor role in the Criminology department. For some reason Lee was not able to work properly as an Advisor in the criminology department. Lee thinks that, Sunil who has the advisor role in Digital forensic department is more suitable for the task then himself. So, he decides to transfer the responsibility of his advisor role to Sunil. Upon transferring the advisor role to Sunil, Lee will lose all of his permissions and rights on the advisory role and Sunil will become an original member of the advisor role. After transferring the role to Sunil, Lee will no more have any responsibility in the advisor role. Furthermore, in this model, a user can further transfer the received role to another user since the receiver of the transfer delegation becomes the original member of the role. A user delegating the role is responsible for updating the delegation in the *UA* relation. A user, who performs the transfer delegation will add the membership of new user and delete its membership from the role.

Following is the mathematical Modelling of Permanent delegation in $RBAC_0$ or $RBAC_1$

1. The *UA* relation is dynamic.
2. The tuples in *can_delegate* allow us to define a transition relation over *UA*
   Modelling the effect of delegation as follows: -
   a. If $(r, pre) \in can\_delegate$ and $u`$ is an explicit member of r and u satisfies pre (with respect to UA_0)
   b. Then $UA` = UA \cup \{(u, r)\} \setminus \{(u`, r)\}$
3. We say that user $u`$ has transferred the role $r$ to the user $u$ (who satisfies $pre$) and we write $UA \rightarrow UA`$ to denote the transition induced by the delegating action.

When considering $RBAC_0$, the above definition does not use any role hierarchy when checking if the user u satisfies the preconditions *pre*. This means that the user u is checked to be an explicit member of a role r when $+r$ is in the precondition pre (or not to be an explicit member of r when $-r$ is in pre). Instead, when considering $RBAC_1$, the above definition does use the role hierarchy. So, the user u is checked to be (not to be) a member (either implicit or explicit) of role r when $+r$ ($-r$, respectively) belong to pre.

VI. CONCLUSIONS

One of the most important extensions for flexibility to RBAC consists of adding a delegation and revocation mechanism for the roles. The contribution with this work is the abstract mathematical framework (based on set theory) to specify the various types of delegation and revocation considered in E. Barka's PhD paper [9]. On the basis of the framework presented by E. Barka, we have presented a mathematical model for specifying the delegation and revocation of roles in RBAC which are necessary to maintain some desired security property. We have generalized E. Barka's approach by introducing a precondition to make delegation more flexible. While modelling, we have introduced some rules known as precondition, which a user must satisfy to be identified as a candidate user to receive the delegation. Any user satisfying the precondition is eligible to receive either the grant delegation or the transfer delegation.

In grant delegation, roles are delegated to another user according to some rules, and the delegator maintains his membership in the delegated role. Grant delegation comes in two flavours i.e. with or without role hierarchy. Grant Delegation proposed in this paper for $RBAC_0$ does not account the role hierarchy and the roles that have been delegated once cannot be delegated further. Whereas, grant delegation in $RBAC_1$ account role hierarchy. We have presented three sub models for grant delegation in role hierarchy, they are: -

1. One step delegation where roles are delegated within role hierarchy for one time and the roles that have been delegated once cannot be delegated further
2. Two-step delegation where the role received from a delegation can be further delegated to another user, but the user receiving the second delegation cannot delegate that role further.
3. K step delegation is the generalization of the previous delegation model, which enables users to delegate the role for K steps.

In, transfer delegation the user associated with the role can transfer its role to any user provided that he satisfies some preconditions. In case of transfer delegation, the delegator loses his membership in the delegated role. Transfer delegation also comes in two flavours i.e. in $RBAC_0$ and $RBAC_1$. Transfer delegation in $RBAC_0$, the user is checked to be an explicit member or not to be an explicit member of the role. Instead in $RBAC_1$, the user is checked to be (not to be) a member (either implicit or explicit) of the role.

VII. DISCUSSION

On the basis of RBAC96 model which was developed by R. Sandhu, E. Barka in [9] has provided framework for two ($RBAC_0$ and $RBAC_1$) role based delegation and revocation models to illustrate some practical access control policies. But it does not cover mathematical Modelling of delegation and revocation of roles in RBAC. This paper has come up with the mathematical Modelling of grant and transfer delegation with two flavours of RBAC ($RBAC_0$ and $RBAC_1$. This mathematical Modelling for delegation and revocation of roles in RBAC96 helps an authorized user to delegate or revoke his role or some junior roles to another user in efficient and secure manner.

Similarly, E. Barka in [9] has defined can-delegate function in his framework which describes what type of role a user can delegate. But unfortunately, he has not mentioned any condition which a user must satisfy in order to receive the delegation, for this problem we have defined a precondition. Preconditions, which is used to identify candidate users to receive the delegation. It means that a receiver of a delegation must satisfy a precondition in order to receive various types of delegation role.

In 2000 G.J. Ahn and R. Sandhu in [16] has defined Role-Based Authorization Constraints Specification and also introduce an intuitive formal language for specifying role-based authorization constraints named RCL 2000 including its basic elements, syntax, and semantics. Their model is more towards authorization rather than delegation and revocation of roles.

Mathematical Modelling of ARBAC, which deals with the administration aspect of RBAC, which is well defined and fully formalized [10] [1] and unfortunately, does not deals with delegation and revocation of roles of RBAC.

VIII. REFERENCES

[1] F. Alberti, A. Armando and S. Ranise, "Efficient Symbolic Automated Analysis of Administrative Role Based Access Control Policies," in 6th ACM Symposium on Information, Computer, and Communications Security (ASIACCS), Hong Kong, 2011.

[2] R. Sandhu, E. J. Coyne, H. L. Feinstein and C. E. Youman, "Role-Based Access Control Models," IEEE Computer, vol. 29, pp. 38-47, 1996.

[3] R. Sandhu, "Rationale for the RBAC96 Family of Access Control Models," in The first ACM Workshop on Role-based access control, 1996.

[4] A. Armando and S. Ranise, "Automated Analysis of Infinite State Workflows with Access Control Policies," in 7th Int. Workshop on Security and Trust Management, Copnhagen, 2011.

[5] R. Sandhu, V. Bhamidipati and M. Q., "The ARBAC97 Model for Role-Based Administration of Roles," 1998.

[6] R. Sandhu, D. Ferraiolo and R. Kuhn, "The Nist Model for Role-Based Access Control: Towards a Unified Standard.," [Online]. Available: http://csrc.nist.gov/rbac/sandhu-ferraiolo-kuhn-00.pdf.

[7] W. Jansen, "Inheritance Properties of Role Hierarchies," [Online]. Available: http://csrc.nist.gov/groups/SNS/rbac/documents/design_implementation/pp-rbac-fin.pdf.

[8] D. F. Ferraiolo and D. R. Kuhn, "Role-Based Access Controls," in National Computer Security Conference, 1992.

[9] E. Barka, "Framework for Role-Based Delegation Models," Virginia, 2002.

[10] A. Armando and S. Ranise, "Automated Symbolic Analysis of ARBAC Policies," in 6th Int. Workshop on Security and Trust Management, Athens, 2010.

[11] D. F. Ferraiolo, J. A. Cugini and D. R. Kuhn, "Role-Based Access Control (RBAC): Features and Motivations," 11th Annual Computer Security Applications Proceedings, 1995.

[12] D. R. Kuhn, E. J. Coyne and T. R. Weil, "Adding Attributes to Role-Based Access control," 2010.

[13] E. Barka and R. Sandhu, "Role-Based Delegation Model / Hierarchical Roles (RMDM1)," in 20th Annual Computer Security Applications Conference, 2004.

[14] N. Li and M. V. Tripunitara, "Security analysis in role-based access control," in ACM Transactions on Information and System Security (TISSEC), 2006.

[15] E. Barka and R. Sandhu, "Framework for Role-Based Delegation Model," in Computer Security Applications, 2000. ACSAC '00. 16th Annual Conference , 2000.

[16] G. Ahn and R. Sandhu, "Role-Based Authorization Constraints Specification," ACM Transactions on Information and System Security, vol. 3, no. 4, p. 207–226, 2000.