Separation of Duty in Role-Based Access Control Model through Fuzzy Relations

Hassan Takabi Morteza Amini Rasool Jalili
Network Security Center
Computer Engineering Department
Sharif University of Technology
Tehran, Iran*
{takabi@ce., m_amini@ce., jalili@}sharif.edu

Abstract

As a security principle, separation of duty (SoD) is widely considered in computer security. In the role-based access control(RBAC) model, separation of duty constraints enforce conflict of interest policies. There are two main types of separation of duty policies in RBAC, Static SoD (SSoD) and Dynamic SoD (DSoD). In RBAC, Statically Mutually Exclusive Role (SMER) constraints are used to enforce Static Separation of Duty policies. Dynamic Separation of duty policies, like SSoD policies, are intended to limit the permissions that are available to a user. However, DSoD policies differ from SSoD policies by the context in which these limitations are imposed. A DSoD policy limits the availability of the permissions over a users permission space by placing constraints on the roles that can be activated within or across a users sessions. Like SMER, in RBAC Dynamically Mutually Exclusive Role (DMER) constraints are used to enforce DSoD policies. We investigated using of a fuzzy approach to address the issue in order to provide a more practical solution. In this paper, we propose a model to express the separation of duty policies in RBAC using the fuzzy set theory. The concept of trustworthiness, which is fuzzy in nature, is used to express this model. In comparison with non-fuzzy methods, our method is more pragmatic and more consistent with the real world. The expressiveness of our method is higher than the non-fuzzy ones. We show expression of some constraints in our method which cannot be expressed by non-fuzzy methods. Applicability of the method is shown through an example of the real world.

*This work is partially supported by Iran Telecommunication Research Center(ITRC) under grant number 500/8478.

1. Introduction

Separation of duty (SoD) is an important security principle used for prevention of fraud and errors. It is used to enforce conflict of interest policies, requiring that two or more different users be responsible for the completion of a task or set of related tasks. The purpose of separation of duty in RBAC is "to ensure that failures of omission or commission within an organization are caused only as a result of collusion among individuals. To minimize the likelihood of collusion, individuals of different skills or divergent interests are assigned to separate tasks required in the performance of a business function. The motivation is to ensure that fraud and major errors cannot occure without deliberate collusion of multiple users"[13]. The simplest form of the SoD principle states that, if a sensitive task is comprised of two steps, then different users should perform different steps. Generally, when a sensitive task is comprised of n steps, an SoD policy requires the cooperation of at least k (for some $k \leq n$) different users to complete the task. Consider the following example of purchasing and paying for goods. The steps to perform such a task are: (1) ordering the goods and recording the details of the order; (2) recording the arrival of the invoice and verifying that the details on the invoice match the details of the order; (3) verifying that the goods have been received, and the features of the goods match the details on the invoice; and (4) authorizing the payment to the supplier against the invoice [8]. We want to ensure that for an order that was never placed yet, no payment released, and that the received goods match those in the order and those in the invoice. If we consider a policy that requires a different user to perform each step, it may be too restrictive. It may be permissible, for instance, that the user who places the order also records the arrival of the invoice. One may require that (a) at least three users cooperation is needed to perform all four steps, and (b) two different users perform steps (1) and (4) (i.e., no single user can order goods and authorize payment for them).

0-7695-2876-7/07 $25.00 © 2007 IEEE
DOI 10.1109/IAS.2007.68

An SoD policy may be enforced either statically or dynamically. In static enforcement, Static SoD (SSoD) policies are specified. SSoD enforces constraints on the assignment of users to roles. Each SSoD policy states that no k - 1 users together have all permissions to complete a sensitive task. It seems that if an SSoD policy is satisfied, then the corresponding SoD policy is also satisfied. However, care must be taken to ensure this. Consider the example described above. Suppose that initially a user Bob has the permission to order goods. After placing an order, Bob's order permission is revoked and then Bob is assigned to have the permission to authorize payments. Now Bob can authorize a payment against the order he placed earlier. The SoD policy is violated even though Bob never has the order permission and payment permission at the same time. Such situations can be avoided by dynamic separation of duty(DSoD) policies. DSoD allows a user to be authorized for two or more roles that do not create a conflict of interest when acted independently, but produce policy concerns when activated simultaneously. Dynamic Separation of duty (DSoD) relations, like SSoD relations, are intended to limit the permissions that are available to a user. This model component defines DSoD properties that limit the availability of the permissions over a users permission space by placing constraints on the roles that can be activated within or across a users sessions.

Separation of Duty has been studied extensively in RBAC [3], [4], [5]. Ferraiolo *et al.* [13] states that "one of RBACs great advantages is that SoD policies can be implemented in a natural and efficient way". RBAC uses mutual exclusion constraints to implement SoD policies. The most common kind of mutual exclusion constraint is Statically Mutually Exclusive Roles (SMER). Generally, a SMER constraint requires that no user is a member of t or more roles in a set of m roles $\{r_1, r_2, ..., r_m\}$. SMER constraints are part of most RBAC models, including the RBAC96 models by Sandhu et al. [14] and the proposed NIST standard for RBAC [15]. Literature in RBAC also studies dynamic mutually exclusive role (DMER) constraints. With such a constraint, a user is prevented from activating mutually exclusive roles simultaneously in a session. SMER and DMER constraints are the only types of constraints included in the proposed NIST standard for RBAC [15].

This paper proposes a new paradigm for separation of duty policies in role-based access control. The paradigm is based on using the fuzzy set theory and in particular the concept of trust and trustworthiness which have the fuzzy nature. In description of the idea, we assume the reader is familiar with the basic RBAC concepts.

The rest of this paper is organized as follows. Section 2 discusses the related work. In section 3, we present a brief formal definition of the extended RBAC model. Finally, section 4 provides our method for modeling separation of duty(SoD) policies based on fuzzy set theory, followed by our conclusion.

2. Related Work

The concept of SoD has long been existed in the physical world, sometimes under the name the two-man rule in the banking industry and the military. In the information security literature the notion of SoD first appeared in Saltzer and Schroeder [1] under the name separation of privilege. Clark and Wilson [2] called attention to separation of duty as one of the major mechanisms to counter fraud and error. Separation of duty ensures that the objects in the real world are consistent with the information about these objects in the computer system. Nash and Poland [9] emphasized the difference between dynamic and static enforcement of SoD policies. In one of the earliest paper on RBAC, Ferraiolo et al. [15] used the terms Static and Dynamic SoD to refer to static and dynamic enforcement of SoD. A DSoD constraint prevents a user from simultaneously activating mutually exclusive roles in a session. However, as we now discuss, DSoD constraints do not seem to enforce SoD policies, because they do not prevent a user from activating mutually exclusive roles across multiple sessions. In RBAC, each session has only one user. Thus, a sensitive task cannot be finished in one session; several sessions are required. Consider the example discussed in Section 1. Suppose that the permission to place an order and the permission to issue payment are assigned to two different roles that are specified to be mutually exclusive in a DSoD constraint. Bob can start a session, activate the role having the order permission, create an order, end the session, start another session, activate the role having the payment permission, and authorize a payment against the order. This violates the SoD policy. Kuhn [5] discussed mutual exclusion of roles for separation of duty and proposes a safety condition: that no one should possess the privilege to execute all step of a task, thereby being able to execute the task. Simon and Zurko [11] and Gligor et al. [4] discuss various kinds of constraints and their usage in RBAC. The latter discusses also the composition of constraints. Both papers refer to their proposed constraints as SoD policies. Several constraints languages have been proposed to support SoD in RBAC [3], [6], [7]. Sandhu [10] presented a history-based mechanism for enforcing SoD policies dynamically. Li et al. [8] discussed about enforcing SoD policies and also studied problem of verification of enforcement. Moon et al. [12] proposed a symmetric RBAC model that supplements the constraints on permission assignment. The proposed model reflects the conflicts of interests between roles by presenting the constraints on permission assignment that take the separation of duties and role hierarchies into consideration.

3. Fuzzy Role Based Access Control Model

In our previous work [16], we presented a method to improve RBAC using fuzzy set theory. We defined two parameters related to the concept of trust and trustworthiness. The first parameter is user trustworthiness (UT) which means how much a user in system is reliable and how mush we trust him/her to assign a specific role or roles in RBAC. The second parameter is role's required trustworthiness (RT) which determines the amount of trust is required by a user to play the role in system. Then, we presented an algorithm to compute these two parameters using fuzzy relation equations. After computing a user trustworthiness (UT) and a role's required trustworthiness (RT), user-role assignment (UA) and role activation are performed based on the trust level of the user (UT) in comparison with the required trust level of the role (RT). In user assignment(UA) relation, user u_i can be assigned to role r_j, if and only if the user trustworthiness ($UT(u_i)$) satisfies the role's required trustworthiness ($RT(r_j)$). Analogously, in role activation, user u_i can activate role r_j, if and only if the user trustworthiness ($UT(u_i)$) satisfies the role's required trustworthiness ($RT(r_j)$).

The model definition is based on the RBAC formalism presented in [13]. The model has the following components:

- USERS: a set of users
- ROLES: a set of roles
- OBS: a set of objects
- OPS: a set of operations
- PRMS: a set of permissions
- TD: a set of TRUSTWORTHINESS DEGREEs

TD is a degree that represents the trustworthiness of a component. Here we use the trustworthiness of the user (UT), and the roles required trustworthiness (RT).

3.1. The Model Specification

Considering the following definitions and also the basic fuzzy concepts, and the model components, a formal definition of our model will be presented.

- USERS, ROLES, OPS, and OBS (users, roles, operations and objects respectively).
- $UA \subseteq USERS \times ROLES$, a many-to-many mapping usere-to-role assignment relation.
- $assigned\_users : (r : ROLES) \rightarrow 2^{USERS}$, the mapping of role *r* into a set of users. Formally: $assigned\_users(r) = \{u \in USERS \mid (u, r) \in UA\}$
- $PRMS = 2^{(OPS \times OBS)}$, the set of permissions.
- $PA \subseteq PRMS \times ROLES$, a many-to-many mapping permission-to-role assignment relation.
- $assigned\_permissions(r : ROLES) \rightarrow 2^{PRMS}$, the mapping of role *r* into a set of permissions. Formally: $assigned\_pemissions(r) = \{p \in PRMS \mid (p, r) \in PA\}$
- $UT(u : USERS) \rightarrow TD$, the mapping of user *u* into the corresponding trustworthiness degree (many-to-one mapping).
- $RT(r : ROLES) \rightarrow TD$, the mapping of role *r* into the corresponding trustworthiness degree (many-to-one mapping).
- $Op(p : PRMS) \rightarrow \{op \subseteq OPS\}$, the permission to operation mapping, which gives the set of operations associated with permission *p*.
- $Ob(p : PRMS) \rightarrow \{ob \subseteq OBS\}$, the permission to object mapping, which gives the set of objects associated with permission *p*
- $SESSIONS$ =the set of sessions.
- $session\_user(s : SESSIONS) \rightarrow USERS$, the mapping of session *s* into the corresponding user.
- $session\_roles(s : SESSIONS) \rightarrow 2^{ROLES}$, the mapping of session *s* into a set of roles. Formally: $session\_roles(s_i) \subseteq \{r \in ROLES \mid (session\_users(s_i), r) \in UA\}$
- $avail\_session\_perms(s : SESSIONS) \rightarrow 2^{PRMS}$, the permissions available to a user in a session= $\bigcup_{r \in session\_roles(r)} assigned\_permissions(r)$

4. Separation of Duty Policies through Fuzzy Relations

As our intent is modeling the real world, using a fuzzy approach to model the separation of duty policies, can provide a more practical solution. We define three types of separation of duty policies using fuzzy set theory. These types are Fuzzy Static Separation of Duty(FSSoD), Fuzzy Statically Mutually Exclusive Role(FSMER), and Fuzzy Dynamically Mutually Exclusive Role(FDMER) respectively. These policies are defined as follows:

We assume that u, r, p, td are a user, role, permission, and trustworthiness degree. us, rs, and ps are a set of users, roles, and permissions respectively . Functions $UT(u)$, and $RT(r)$ give us the trustworthiness degree of a user u and a role r. The $FuzzyUnion$ notion refers to the union operation of two fuzzy sets that defined in section 3.

FSSoD policies

A Fuzzy Static Separation of Duty(FSSoD) policy is expressed as $fssod(\{p_1, p_2, ..., p_n\}, td)$.
$FSSoD \subseteq (2^{PRMS} \times TD)$ is collection of pairs (ps, td) in Fuzzy Static Separation of Duty, with the property that only a set of users that together have enough trustworthiness degree can perform a task that requires all of these permissions in $\{p_1, p_2, ..., p_n\}$. We assume a role with all of these permissions ($r^{'}$) and then compute RT for this assumed role using $compute\_RT(r^{'})$ and name it td.

Formally:

$assigned\_users\_of\_permission(p) =$
$\{u \in USERS | \exists r : (u, r) \in UA, (p, r) \in PA\}$
$assigned\_users\_of\_permissions(ps) =$
$\bigcup_{p \in ps} assigned\_users\_of\_permission(p)$
$\forall (ps, td) \in FSSoD, \forall us \subseteq USERS$: Compare(
$FuzzyUnion_{u \in (us \cap assigned\_users\_of\_permissions(ps))}$
$(UT(u)), td) = TRUE$

In the presence of a role hierarchy Fuzzy Static Separation of Duty is redefined based on authorized users rather than assigned users as follows:

$authorized\_roles\_p(p) = \{r \in ROLES | r \geq r^{'},$
$(p, r^{'}) \in PA\}$
$authorized\_users\_of\_permission(p) = \{u \in USERS |$
$\exists r \in authorized\_roles\_p(p) : (u, r) \in UA\}$
$authorized\_users\_of\_permissions(ps) =$
$\bigcup_{p \in ps} authorized\_users\_of\_permission(p)$
$\forall (ps, T) \in FSSoD, \forall us \subseteq USERS$: Compare(
$FuzzyUnion_{u \in (us \cap authorized\_users\_of\_permissions(ps))}$
$(UT(u)), td) = TRUE$

In FSSoD policies, at the contrary of SSoD policies, there is no need to determine the exact number of users. It is sufficient to check, whether a set of users together have enough trustworthiness degree to perform the task or not. Therefor, enforcement of the least privilege principle [1] with FSSoD is easier than with SSoD.

FSMER Constraints

A Fuzzy Statically Mutually Exclusive Role(FSMER) constraint is expressed as $fsmer(\{r_1, r_2, ..., r_m\}, td)$

$FSMER \subseteq (2^{ROLES} \times TD)$ is collection of pairs (rs, td) in Fuzzy Statically Mutually Exclusive Role, where td is a trustworthiness degree, with the property that a user can acquire roles from $\{r_1, r_2, ..., r_m\}$ that their trustworthiness degree together is not greater than td.

Formally:

$assigned\_roles(u) = \{r \in ROLES | (u, r) \in UA\}$
$\forall (rs, T) \in FSMER, \forall u \subseteq USERS$: Compare(
$FuzzyUnion_{r \in (rs \cap assigned\_roles(u))}(RT(r)), td)$
$= FALSE$

In the presence of a role hierarchy Fuzzy Statically Mutually Exclusive Role is redefined based on authorized roles rather than assigned roles as follows:

$authorized\_roles(u) = \{r \in ROLES | r^{'} \geq r,$
$(u, r^{'}) \in UA\}$
$\forall (rs, T) \in FSMER, \forall u \subseteq USERS$: Compare(
$FuzzyUnion_{r \in (rs \cap authorized\_roles(u))}(RT(r)), td)$
$= FALSE$

FDMER Constraints

A Fuzzy Dynamically Mutually Exclusive Role(FDMER) constraint is expressed as $dmer(\{r_1, r_2, ..., r_n\}, td)$

$FDMER \subseteq (2^{ROLES} \times TD)$ is collection of pairs (rs, td) in Fuzzy Dynamically Mutually Exclusive Role, where td is a trustworthiness degree, with the property that in a session a user can activate roles from $\{r_1, r_2, ..., r_m\}$ that their trustworthiness degree together is not greater than td.

Formally:

$session\_user(s) : SESSIONS \rightarrow USERS$
$session\_roles(s) = \{r \in ROLES |$
$(session\_user(s), r) \in UA\}$
$\forall s \in SESSIONS, \forall (rs, T) \in FDMER,$
$\forall u \in USERS$: Compare(
$FuzzyUnion_{r \in (rs \cap session\_roles(s))}(RT(r)), td)$
$= FALSE$

In the presence of a role hierarchy Fuzzy Dynamically Mutually Exclusive Role is redefined based on session authorized roles rather than session roles as follows:

$session\_authorized\_roles(s) = \{r \in ROLES | r^{'} \geq r,$
$(session\_user(s), r^{'}) \in UA\}$
$\forall s \in SESSIONS, \forall (rs, T) \in FDMER,$
$\forall u \in USERS$: Compare(
$FuzzyUnion_{r \in (rs \cap session\_authorized\_roles(s))}$
$(RT(r)), td) = FALSE$

In FSMER/ FDMER policies, using td we can limit the access permissions of a user such that a user has not permissions greater than a specified level, whereas in SMER/ DMER policies, we can't express such constraints. Also in SMER/ DMER policies, to enforce least of privilege principle, One has to consider all possible combinations of roles that can be assigned/ activated together. The number of such combinations is likely much larger than the number

of roles, and therefore, considering all such combinations due to verify constraints is very difficult and complicated. That is obvious that, using FSMER/ FDMER to enforce least privilege principle is much easier than using SMER/ DMER, because in this case, less policies are needed to express constraints -will be shown in section 5 with an example- and also it is more powerful to express constraints.

FUSMER/ FUDMER Constraints

Another group of policies can be expressed using our method too. In FUSMER/ FUDMER we can use the user trustworthiness degree ($UT(u)$) instead of td. In FSMER/ FDMER, a user can only acquire/ activate roles from a role set that their trustworthiness degree together is not greater than td,whereas in FUSMER/ FUDMER a user can only acquire/ activate roles from a role set that their trustworthiness degree together is not greater than $UT(u)$. In fact, using FSMER/ FDMER policies, we express constraints such that each user regardless of its trustworthiness degree can only acquire/ activate roles from a role set that their trustworthiness degree together is not greater than td,whereas in FUSMER/ FUDMER the roles that a user can acquire/activate depends on its trustworthiness degree. It is clear that by using SMER/ DMER policies we can not express such constraints.

5. Application Example

In order to illustrate the applicability of the proposed model, consider the buying and paying for goods problem from section 1 as an example. The model components are defined as follows:
$USERS = \{Alice, Bob, Cathy, Dina\}$
$ROLES = \{r_1, r_2, r_3, r_4\}$
$PRMS = \{p_1, p_2, p_3, p_4, p_5, p_6, p_7\}$, such that p_1 is 'ordering the goods', p_2 is 'recording the details of the order', p_3 is 'recording the arrival of the invoice', p_4 is 'verifying that the details on the invoice match the details of the order', p_5 is 'verifying that the goods have been received', p_6 is 'verifying that the features of the goods match the details on the invoice', and p_7 is 'authorizing the payment to the supplier against the invoice'.
$assigned\_permissions(r_1) = \{p_1, p_2\}$
$assigned\_permissions(r_2) = \{p_3, p_4\}$
$assigned\_permissions(r_3) = \{p_5, p_6\}$
$assigned\_permissions(r_4) = \{p_7\}$
We have these constraints that (a) at least three users cooperation is needed to perform all four steps, and (b) no single user can order goods and authorize payment for them.
With non-fuzzy method which is explained in RBAC standard [13], these constraints are expressed as:
$ssod(\{p_1, p_2, p_3, p_4, p_5, p_6, p_7\}, 3)$
$smer(\{r_1, r_4\}, 2)$
We define Y, a finite set of values that can be assigned to trustworthiness degree, as follows: $Y = \{0, 0.2, 0.4, 0.6, 0.8, 1\}$ We compute the user trustworthiness(UT) for users and role's required trustworthiness(RT) for roles using proposed algorithm in section 4 and we have:
$UT(Alice) = [0.6, 0.6, 0.4, 0.3, 0.2, 0.1]$
$UT(Bob) = [0.5, 0.6, 0.7, 0.7, 0.7, 0.9]$
$UT(Cathy) = [0.5, 0.5, 0.6, 0.7, 0.8, 0.8]$
$UT(Dina) = [0.6, 0.5, 0.7, 0.6, 0.8, 0.9]$
$RT(r_1) = [0.6, 0.6, 0.5, 0.5, 0.4, 0.4]$
$RT(r_2) = [0.5, 0.6, 0.7, 0.7, 0.8, 0.8]$
$RT(r_3) = [0.1, 0.2, 0.4, 0.6, 0.7, 0.9]$
$RT(r_4) = [0.1, 0.1, 0.3, 0.5, 0.7, 0.9]$

Now,we expressed mentioned constraints using our proposed method as:
$fssod(\{p_1, p_2, p_3, p_4, p_5, p_6, p_7\}, td)$
We assume a role with all of these permissions $\{p_1, ..., p_7\}$ and then compute RT for this assumed role using $compute\_RT$. The computed value for RT is $td = [0.7, 0.7, 0.7, 0.7, 0.8, 0.9]$
Considering formal definition of FSSoD policies in section 6 and the value of UT and RT for users and roles and value of the td, at least three users is needed that together have enough trustworthiness degree to perform all of four steps and in fact to have all of $\{p_1, p_2, p_3, p_4, p_5, p_6, p_7\}$ permissions, and there are no two users that together have enough trustworthiness degree to perform all of these tasks.

The second constraint expressed as:
$fsmer(\{r_1, r_4\}, td)$ that td is
$td = [0.5, 0.5, 0.6, 0.6, 0.7, 0.9]$.
Considering formal definition of FSMER policies in section 6 and the value of RT for roles, as RT of $\{r_1, r_4\}$ together is greater than td, no single user can acquire these two roles.

Also we can express constraints as:
$fusmer(\{r_2, r_4\}, UT(u))$ that $UT(u)$ is a user's trustworthiness degree.
Considering formal definition of FUSMER policies in section 6 and the value of UT and RT for users and roles, as RT of $\{r_2, r_4\}$ together is greater than UT of each of users, no single user can acquire these two roles.

Assume that, we want to express a constraint that a user that acquire r_4, can not acquire r_1 or r_2, but there is no constraint to acquire r_1 and r_2 together. We can express this constraint using $fsmer(\{r_1, r_2, r_4\}, td)$. As the $RT(r_4)$ is relatively big, a user that acquire r_4 can not acquire any other role. The $RT(r_1)$ and $RT(r_2)$ are relatively small, thus a user can acquire both of them. If we want to use SMER policies due to express this constrain, we should use $smer(\{r_1, r_4\}, 2)$, $smer(\{r_2, r_4\}, 2)$ policies.

As described above, we can express such constraint us-

ing one FSMER policy, whereas if we use SMER policies to express this constraint, at least two policies are needed.

6. Conclusion

In this paper, a new paradigm for separation of duty policies in role-based access control model were introduced. The proposed paradigm uses fuzzy set theory to model separation of duty policies in the RBAC model. We defined the FSSoD, FSMER/ FDMER, and FUSMER/ FUDMER policies based on the concept of trust and trustworthiness, which are fuzzy in nature, are used. In comparison to non-fuzzy methods in expressing separation of duty policies, our method is more pragmatic, and more consistent with the real world. It is shown that enforcement of least privilege principle with FSSoD is easier than with SSoD. In comparison to SMER/ DMER as using FSMER/ FDMER needs less policies to express constraints. In addition, the method facilitates expression of constraints which can not be expressed using non-fuzzy methods.

References

[1] J. H. Saltzer and M. D. Schroeder, *The protection of information in computer systems*, In Proceedings of the IEEE, 63(9), pp. 1278-1308, 1975.

[2] D. D. Clark and D. R. Wilson, *A comparision of commercial and military computer security policies*, In Proceedings of the IEEE Symposium on Security and Privacy, pp. 184-194, 1987.

[3] J. Crampton, *Specifying and enforcing constraints in role-based access control*, In Proceedings of the Eighth ACM Symposium on Access Control Models and Technologies (SACMAT 2003), pp. 43-50, Como, Italy, 2003.

[4] V. D. Gligor, S. I. Gavrila, and D. F. Ferraiolo, *On the formal definition of separation-of-duty policies and their composition*, In Proceedings of IEEE Symposium on Research in Security and Privacy, pp. 172-183, 1998.

[5] D. R. Kuhn, *Mutual exclusion of roles as a means of implementing separation of duty in role-based access control systems*, In Proceedings of the Second ACM Workshop on Role-Based Access Control (RBAC97), pp. 23-30, 1997.

[6] G. J. Ahn and R. Sandhu, *Role-based authorization constraints specification*, ACM Transactions on Information and System Security, 3(4):207226, 2000.

[7] G. J. Ahn and R. Sandhu, *The RSL99 Language for Role-Based Separation of Duty Constraints*, In Proceedings of the 4th ACM Workshop on Role-Based Access Control, Fairfax, Virginia, pp. 43-54, 1999.

[8] N. Li, Z. Bizri, and M. V. Tripunitara, *On mutually-exclusive roles and separation of duty*, In Proceedings of the 11th ACM conference on Computer and communications security, pp. 42-51 ,Washington DC, USA ,2004.

[9] M. J. Nash and K. R. Poland, *Some conundrums concerning separation of duty*, In Proceedings of the IEEE Symposium on Research in Security and Privacy, pp. 201-209, 1990.

[10] R. S. Sandhu, *Transaction control expressions for separation of duties*, In Proceedings of the Fourth Annual Computer Security Applications Conference (ACSAC88), 1988.

[11] T. T. Simon and M. E. Zurko, *Separation of duty in role-based environments*, In Proceedings of the 10th Computer Security Foundations Workshop, pp. 183-194. IEEE Computer Society Press, 1997.

[12] C. J. Moon, D. H. Park, S. J. Park, and D. K. Baik, *Symmetric RBAC model that takes the separation of duty and role hierarchies into consideration*, Computers and Security, Vol. 23, No. 2, pp. 126-136, 2004.

[13] ANSI. American National Standard for Information Technology- *Role Based Access Control*, ANSI INCITS 359-2004, 2004.

[14] R. Sandhu, E. Coyne, H. Feinstein, and C. Youman, *Role Based Access Control Models*, IEEE Computer, Vol. 29, No. 2, pp. 38-47, 1996.

[15] D. F. Ferraiolo, R. Sandhu, S. I. Gavrila, D.R. Kuhn, and R. Chandramouli, *Proposed NIST Standard for role based access control*, ACM Transactions on Information and system securiy, Vol. 4, No. 3, pp. 224-274, 2001.

[16] H. Takabi, M. Amini, and R. Jalili, *Trust-Based User-Role Assignment in Role-Based Access Control*, In Proceedings of the ACS/IEEE International Conference on Computer Systems and Applications (AICCSA 2007), 2007.