A Formal Structure of Separation of Duty and Trust in Modelling Delegation Policy

Ogundele O.S., Adewale O.S., Alese B.K. and Falaki S.O.

Department of Computer Science,
Federal University of Technology, Akure, Nigeria.
oloruntoba78@yahoo.com

Abstract

There are considerable number of approaches to policy specification both for security management and policy driven network management purposes as reported in [20]. *This specification sort security policies into two basic types: authorization and obligation policies. Most of the researches in security policies specification over the years focus on authorization policy modelling. In this paper, we report our approach in the design and Modelling of obligation Policy as delegation in information security by considering separation of duty and trust as pre-requisite conditions for delegation. The formal structures of the Delegation models developed was adapted from the Mathematical structures of Separation of duty (both Static and Dynamic SoD) in RBAC environment as described in* [8] and [16]. *Three factors of Properties, Experiences and Recommendation as described in* [22] *were used for the Trust Modelling. Future works proposed include the development of a formal model for revocation after delegation and integration of appropriate authorization policy with the model.*

1. Introduction

Separation of Duty (SoD) is widely recognized as a fundamental principle in computer security [5], [19]. In its simplest form, the principle states that a sensitive task should be performed by two or more different users acting in cooperation. The concept of SoD has long existed before the information age; it has been widely used in, for example, the banking industry and the military, sometimes under the name "the two-man rule" [16]. More generally, an SoD policy requires the cooperation of at least *n* different users to complete the task. SoD has been identified as a high-level mechanism that is "at the heart of fraud and error control" [5, 8]. An SoD policy is a high-level policy in the sense that it does not restrict which users are allowed to carry out the individual steps in a sensitive task, but rather states an overall requirement that must be satisfied by any set of users that together complete a task [16].

[8] viewed SoD as an application-design principle based on three well-understood design and implementation steps given as *integrity property definition, application design and user assignment to application partitions.* Users of different skills or interests are assigned to operate in different application partitions. These assignments may last for limited periods of time, and may change dynamically. Therefore, users' collusion to perpetrate fraud will be checked or at least controlled to the minimal.

According to [10] and [21] and as stated in [8] despite the importance of SoD as a security principle and its well-understood application in business, industry, and government; few computer systems have supported SoD as a security policy to date.

The lack of wide-spread support was attributed by [8] to three separate reasons. First, SoD is an inherently *application-oriented* policy and, thus, has been perceived to yield limited payoff for operating systems and networks. Secondly, when the SoD principle is interpreted within different applications, it may yield many different SoD policies [21] and recurrent administrative costs. Third, most SoD policies proposed to date have been only informally defined and, therefore, subject to ambiguous or incomplete specifications, and limited assurance.

Therefore, it is believe that the incorporation of SoD into Role Based Access Control Model (RBAC) to support Role Delegation will address this lack of widespread support as reported in [8]. Since a system must not only support authorization, but must also enable active entities to securely delegate their roles without breaching security policies.

2. Trust

[22] define trust as a relationship between a truster and a trustee and it is dependent on a given task. The truster's trust for a trustee, with respect to a given task, depends on several factors namely, *properties, experiences* and *recommendation.*

Properties are verifiable characteristics of the trustee. Experiences correspond to the past work

experience of the trustee and it reflect past interactions that the truster had with the trustee. Recommendations are the information that the truster obtains from third parties whom the truster trust about the capabilities of the trustee. How to quantify these factors and assess the trustworthiness of an entity before designating him as the delegate has also been described in [22]. Trust modelling has been reported in [11] and [13] where it was claimed that trust is a relationship between two entities on a specific statement and is represented using degrees of *belief* b, *disbelief* d and *uncertainty* U.
In [11] and [13], *subjective logic* was utilized to define *recommendation* and *consensus* formulae in order to take into account multiple subjective views on the same statement. In [12], a new component *base rate* a, was added where a [0,1]. In [14], how to specify trust networks consisting of multiple paths between the trusted parties and provide a practical method for analyzing and deriving measures of trust in such environments were shown.
[4] proposed a Trust based Access Control (TrustBAC) model where the assignment of users to roles depended on their trust value -1 to 1. While in [15]; a trust model where the notion of trust contexts was formalized was designed.

3. Structure of Separation of Duty Policy In RBAC For Delegation Policy Modelling.

In [8], security policies were viewed as composition of security properties. The notion of dependency among policy properties were illustrated by them by considering three types of properties: access-attribute (AT), access-authorization (AA) and access-management (AM).
Access attributes include subject and object attributes (e.g., user, group, role, location identifiers, secrecy and integrity levels, time-of-access intervals), and AT properties typically establish invariant relationships among attributes (e.g., lattices of secrecy and integrity levels, user-group membership invariants, inheritance of role permissions). AA properties determine whether a subject's current access attributes satisfy the conditions for accessing an object given the current object attributes.
Formally, [8] described a simple structure of SoD in RBAC as a security policy *P* as follows:

$\mathcal{P} = \mathrm{P} \wedge Admin(P)$
Where $\mathrm{P} = \mathrm{AT} \wedge \mathrm{AA} \wedge \mathrm{AM}$.
The property P itself may have other properties in addition to $Admin(P)$; for example, application – oriented policies such as SoD, also include property$Compat(\mathrm{P}, App)$. They specify SoD policy as incremental conjunctions of properties to RBAC policies. That is;
$SoD - \mathcal{P} = SoD - \mathrm{P} \wedge Admin(SoD - P) \wedge Compat(SoD - \mathrm{P}, App) \wedge \boldsymbol{RBAC - \mathcal{P}}$Where
$\boldsymbol{RBAC - \mathcal{P}} = \mathrm{RBAC - P} \wedge Admin(\mathrm{RBAC - P})$, and both
$SoD - \mathcal{P}$ and $\mathrm{RBAC - P}$ are conjunctions of AT, AM, and AA properties.

[8] further defined the types, functions and properties of Role Based Access Control (RBAC) system that are necessary to define SoD properties. They considered RBAC system to be defined by a state machine model. They denote set of system states by *STATES*, the set of subjects by *SUBJECTS*, the set of users by *USERS*, the set of operations by *OPERATIONS*, and the set of object by *OBJECTS*. A RBAC system is characterized by the fact that a user's membership to a "role" and by the roles' permissions to perform operations on those objects. Hence, a *role* is a collection of operations on object sets.
The class of roles, *ROLES*, is a subset of $2^{OPERATIONSX}\, 2^{OBJECTS}$.
The function
$auth\colon STATESXROLESXOBJECTS \longrightarrow 2^{OPERATIONS}$ Defines the operations allowed to each role in each state of the system:

$$\forall\, s \in STATES, \forall op \in OPERATIONS, \forall r \in ROLES, \forall obj \in OBJECTS, op \in auth(s, r, obj) \Leftrightarrow \exists objset \subseteq OBJECTS\colon obj \in objset \wedge (op, objset) \in r$$

The function $role\_members\colon ROLES \longrightarrow 2^{USERS}$ defines the users *assigned* to a given role.
The function $subject\_user\colon SUBJECTS \longrightarrow USERS$ returns the user associated with the subject.
The function $subject\_roles\colon STATEXSUBJECTS \longrightarrow 2^{USERS}$ returns the roles assumed by a user in a given state while executing a given subject. These roles must have been assigned to the subject's user.
The function $current\_role\_set\colon STATESXUSERS \longrightarrow 2^{USERS}$ is defined as follows:
$\forall s \in STATES, \forall u \in USERS,$

$$current\_role\_set(s, u) = \bigcup_{\substack{S \in SUBJECT \\ subject\_user(S) = u}} subject\_roles(S)$$

If $r \in current\_role\_set(s, u)$,then we say that role *r* is *enabled* or *active* for the user *u* in state *s*

4. Access Obligation Formalization

As an extension and a refinement to the above as well as the three properties (AT, AM and AA), we introduced access – obligation (AO) as the fourth requirement which is a property that defines obligation policy that we shall modelled as delegation requirement. AO properties determine whether a subject's current role based on attributes and role hierarchy satisfy the conditions for delegating that role to another subject in accessing

the same object given the current object attributes without breaching security policy.
Therefore, we extend the model described in [8] to incorporate delegation (obligation policy) by modelling Access Obligation which is formalized as follows;

$$\mathcal{P} = \mathrm{P} \wedge \mathrm{AO} \wedge Admin(P)$$

To formalize Access – Obligation (AO), consider an Access Control system that is characterized by the fact that a subject's right to access object are defined by subject's attributes, membership to "role" and roles' permissions to perform operations on the object. Hence, a *role* is a collection of secure operations on object sets.
Therefore;
$\forall\, Sbj_i, Sbj_k \in SBJ, \quad \forall\, op \in OP, \quad \forall\, rol \in ROL,$
$\forall\, obj \in OBJ,$
Where Sbj, Obj, op, rol are Subject, Object, Operation and Roles respectively.
Then, Access Obligation is a function modelled as follows,
$AO\colon op \in can\_delegate(Sbj_i, Sbj_k, rol, obj) \Leftrightarrow$
$\exists\, Obj \subseteq OBJ\colon rol \in$
$ROL \wedge (op, obj, role\_permission) \in rol$

The equations above therefore define the function; Access-Obligation (Obligation Policy) as an extension to SoD structure defined in [8].
Varieties of SoD properties and their relationship in secure RBAC systems and their composition based on property conjunctions have been defined. In [21], Static and Dynamic SoD properties were distinguished. [6] defined Static SoD (SSoD) by the rule that "each user must be permitted to use only certain transaction". Their work was formalized by [8] in RBAC environment as follows.
Let App be an application and $RoleSet$ its assigned roles in a secure RBAC system. $\sigma \in \sum_0$ satisfies the $SSoD$ property with respect to App if any two distinct roles in $RoleSet$ do not have common members. Such roles are said to be restricted. Formally,
$\sigma \in SSoD(RoleSet, App) \Leftrightarrow$
$\quad \forall\, r_1, r_2 \in RoleSet, r_1 \neq r_2 \Rightarrow$
$role\_member(r_1) \cap role\_member(r_2) = \emptyset$

We developed a stronger version of this model below to incorporate delegation policy by adding the delegation requirements – (AO: can_delegate) – modelled above that the target object of the two roles be disjoint. Delegation in Static Separation of Duty (SSoD) rules state that delegation of role cannot take place between subjects (even if they have the same or share similar attributes) and belong to different role hierarchies; if the subjects have distinct roles, do not have common members and are not authorized to perform operations on the same object with an application. For instance, Two Accountants (SUBJECTS) that are Professional (the same ATTRIBUTE) on employment in a bank must occupy two distinct positions; which could be an auditor (ROLE1) or an accountant (ROLE2) and they will be performing different functions (OBJECTS).
Let $App = [OBJ, role\_member, can\_delegate]$ and
$\forall\, Sbj_i, Sbj_k \in SBJ, \forall\, Rol_i, Rol_k \in ROL,$
$\forall\, Sbj\_atr_i, Sbj\_atr_k \in SBJ\_ATR,$
$\forall\, Rol\_hier_i, Rol\_hier_k \in ROL\_HIER$
if
$[Sbj\_atr_i(sbj_i) = Sbj\_atr_k(sbj_k)] \in SBJ\_ATR,$
and
$[Rol\_hier_i(Rol_i) \neq Rol\_hier_k(Rol_k)]$
$\qquad \in ROL\_HIER,$
Then
$role\_member(Rol_i) \cap role\_member(Rol_k)$
$= \emptyset$

$$\wedge\, Sbj_i(Rol_i) \bigcap_{Sbj\_atr_i, Sbj\_atr_k \in SBJ\_ATR} Sbj_k(Rol_k)$$

$= \emptyset$
Therefore, Delegation with Static SoD is formalized as the function stated as follows;

$SSoD(Dlg)\colon |\{obj \in$
$OBJ | can\_delegate(Sbj_i, Rol_i, obj_i) \cap App \neq$
$\emptyset\} \overset{\cap}{_{Rol_i, Rol_k \in ROL}} \{obj \in$
$OBJ | can\_delegate(Sbj_k, Rol_k, obj_k) \cap App \neq$
$\emptyset\} = \emptyset$
In [17], object based dynamic SoD was introduced as a more flexible and realistic alternative to the Static SoD. However, their informal definitions [17] [21], does not specify precisely which objects, operations and roles are subjected to the Dynamic SoD (DSoD) condition.
Delegation in DSoD rule state that delegation of role may not take place between subjects (even if they have the same or share similar attributes) and belong to same role hierarchies; if the subjects have distinct roles, may have common members and but are not authorized to perform operations on the same object within an application at the same time. For instance, A professor (SUBJECT) that is presenting a student for oral examination (OBJECT) as thesis supervisor (CURRENT_ROLE(i)) cannot stand as both the internal examiner (CURRENT_ROLE(i)) as well as external examiner (CURRENT_ROLE(k)) even though he is qualified to occupy both position by virtue of his qualifications (ATTRIBUTES). The same professor may later occupy an external examiner's position (CURRENT_ROLE(k)) in another forum where he will not be internal examiner (CURRENT_ROLE(i)) but all acting on the same OBJECT (which is examination).
Let $App = [OBJ, role\_member, can\_delegate]$ and
$\forall\, Sbj_i, Sbj_k \in SBJ,$

$\forall\ Rol_i, Rol_k\ \in Current\_Rol,$
$\forall\ Sbj\_atr_i, Sbj\_atr_k\ \in SBJ\_ATR,$
$\forall\ Rol\_hier_i, Rol\_hier_k\ \in ROL\_HIER$
if
$[Sbj\_atr_i(sbj_i) = Sbj\_atr_k(sbj_k)] \in SBJ\_ATR,$
and
$[Rol\_hier_i(Rol_i) = Rol\_hier_k(Rol_k)] \in ROL\_HIER,$
Then
$role\_member(Rol_i) \cap role\_member(Rol_k) \neq \emptyset\ \wedge$

$Sbj_i, (Rol_i) \underset{Sbj_atr_i, Sbj_atr_k \in SBJ_ATR}{\cap} Sbj_k(Rol_k) \neq \emptyset$Therefore, Delegation with Dynamic SoD is the function modelled as follows;

$$DSoD(Dlg): |\{obj \in OBJ | can\_delegate\ (Sbj_i, role\_member(Current_{Rol_i}), obj) \cap App \neq \emptyset\} \underset{Rol_i, Rol_k\ \in\ ROL}{\cap} \{obj \in OBJ | can\_delegate(Sbj_k, role\_member\ (Current\_Rol_k), obj) \cap App \neq \emptyset\} = \emptyset$$

5. Modelling Trust in Delegation Policy

Having considered various research works that have been done in delegation and separation of duties, and developing models that refined and formalize the basis on which a delegator selects a delegatee to ensure secure delegation of roles from one subject to another, and without breaching the information security policy; we model trust as an extension of the already designed delegation model based on SoD as another pre-requisite condition for delegation.

The work of [22] was refined from the tasks chain to roles and subject attributes chain and is formalized as follows for each of the three criteria defined.

5.1 Formalizing "*Properties*" In Trust Modelling For Delegation Policy.

Basically, properties depend on the attributes of the subjects which are the requirements for a subject to delegate his role(s) to another subject. For example, the role of performing surgery requires the subject to be a certified surgeon. In a more details format, the surgeon has to be a specialist in a particular aspect of surgery (for instance, ENT, gynaecologist, Neuro Surgery e.t.c). Therefore, an ENT cannot delegate his role to a gynaecologist based on the difference in the area of specialization (attributes) notwithstanding that both of them are surgeons. Specific examples can be qualifications and areas of specialization of the subjects which can be quantified or weighted to evaluate the attributes value.

We formally defined structure of properties based on trust as follows:

Let the attributes needed for Sbj_i to delegate his role Rol_i to Sbj_k be defined as follows:
$Sbj\_Atr_i = \{Sbj\_Atr_{i1}, Sbj\_Atr_{i2}, \dots, \dots, Sbj\_Atr_{in}\}$

Then if
$Sbj\_Atr_k \neq \{\emptyset\}$
and
$Sbj\_Atr_i \cap Sbj\_Atr_k \neq \{\emptyset\} = \{P\}$
Then
$Sbj_i \overrightarrow{can\ delegate}\ Sbj_k : (Rol_i)$
else
$Sbj_i \overrightarrow{cannot\ delegate}\ Sbj_k : (Rol_i)$

To assign weight to the attributes value(s), the intersecting elements will be quantified and weighted values will be assigned to each element. The computation of the overall sum of the weights give the relative trust based on a set standard of the security framework.

To formalized the above,

Let the weight assigned to each element of the attributes set be defined as follows:
$Sbj\_Atr\_Wgt_i = \{Sbj\_Atr\_Wgt_{i1}, Sbj\_Atr\_Wgt_{i2}, \dots, \dots, Sbj\_Atr\_Wgt_{in}\}$

Then from equation above,
$Sbj_i \overrightarrow{can\ delegate}\ Sbj_k : (Rol_i)$ if the structure stated as follows is computed to be true and cannot delegate otherwise.

$$Sbj\_Atr\_Wgt_{ik} = \sum_{i=1}^{n} Sbj\_Atr\_Wgt_i \geq (Standard\ Value\ Specify\ for\ Rol_i\ )$$

where $1 \leq i \leq n$.

5.2 Formalizing "*Experience*" In Trust Modelling For Delegation Policy.

Experience constitutes an important factor in delegation. A delegator is more likely to choose a candidate as a delegate if the delegate has prior experience of occupying that role. [22] identified two factors that contribute to experience. One factor is when the task was performed, and the second factor is how well the tasks were performed.

The two factors above were refined to reflect subjects and roles attributes. That is, the period the subject occupies the role and his performance during his occupation of the role. For example, it will be better and more reasonable for an Head of department to delegate his role to a candidate who has occupies the position of an head of department before for the longest period of years and who is adjudged to have performed creditably. Performance values can take a value of 0 (non performer) or 1 (Satisfactorily Performance).

Formally, experience based on trust can be formalized as follows;

Let the performance period and performance values required of role Rol_i be defined as follows respectively:
$Sbj\_Pfm\_Per_i = \{Sbj\_Pfm\_Per_{i1}, Sbj\_Pfm\_Per_{i2}, \dots, Sbj\_pfm\_Per_{in}\}$
And
$Sbj\_Pfm\_Val_i = \{1\}$

Then, the expression below defined the Weight required for the occupation of role Rol_i of Sbj_i;

$$Sbj\_Exp\_Wgt_i = \sum_{i=1}^{n}(Sbj\_Pfm\_Per_i) + Sbj\_Pfm\_Val_i$$

where $1 \leq i \leq n$.

If

$Sbj\_Pfm\_Per_k \neq \{\emptyset\}$

And

$$\sum_{k=1}^{n}(Sbj\_Pfm\_Per_k) + Sbj\_Pfm\_Val_k \geq \sum_{i=1}^{n}(Sbj\_Pfm\_Per_i) + Sbj\_Pfm\_Val_i$$

where $1 \leq k \leq n$. and $1 \leq i \leq n$.

Then

$Sbj_i \overrightarrow{can\ delegate}\ Sbj_k : (Rol_i)$

else

$Sbj_i \overrightarrow{cannot\ delegate}\ Sbj_k : (Rol_i)$

5.3 Formalizing "*Recommendation*" In Trust Modelling For Delegation Policy.

Recommendations are the information that the truster obtains from reputable sources about the trustee. Recommendations in this context are provided from two other subjects whom the truster trust about the capabilities of the trustee.

Formally, recommendation based on trust can be formalized as follows:

Let β be the minimum total values from two recommenders Sbj_z and Sbj_j required for Sbj_i to delegate his role Rol_i to Sbj_k

Given the recommended values of Sbj_k from Sbj_z and Sbj_j as

$Sbj\_Rec\_Val_Z = \{Sbj\_Rec\_Val_{z1}, Sbj\_Rec\_Val_{z2}, \dots, \dots, Sbj\_Rec\_Val_{zk}\}$

And

$Sbj\_Rec\_Val_j = \{Sbj\_Rec\_Val_{j1}, Sbj\_Rec\_Val_{j2}, \dots, \dots, Sbj\_Rec\_Val_{jk}\}$

respectively,

if

$$\sum_{z=1}^{zk}(Sbj\_Rec\_Val_{zk}) + \sum_{j=1}^{jk}(Sbj\_Rec\_Val_{jk}) \geq \beta$$

Then

$Sbj_i \overrightarrow{can\ delegate}\ Sbj_k : (Rol_i)$

else

$Sbj_i \overrightarrow{cannot\ delegate}\ Sbj_k : (Rol_i)$

From the above, the two major requirements have been considered from roles, objects and subjects' attributes as pre-requisite condition for roles delegation in an open system for ease of security administration. The schematic architecture of the model is shown diagrammatically below.

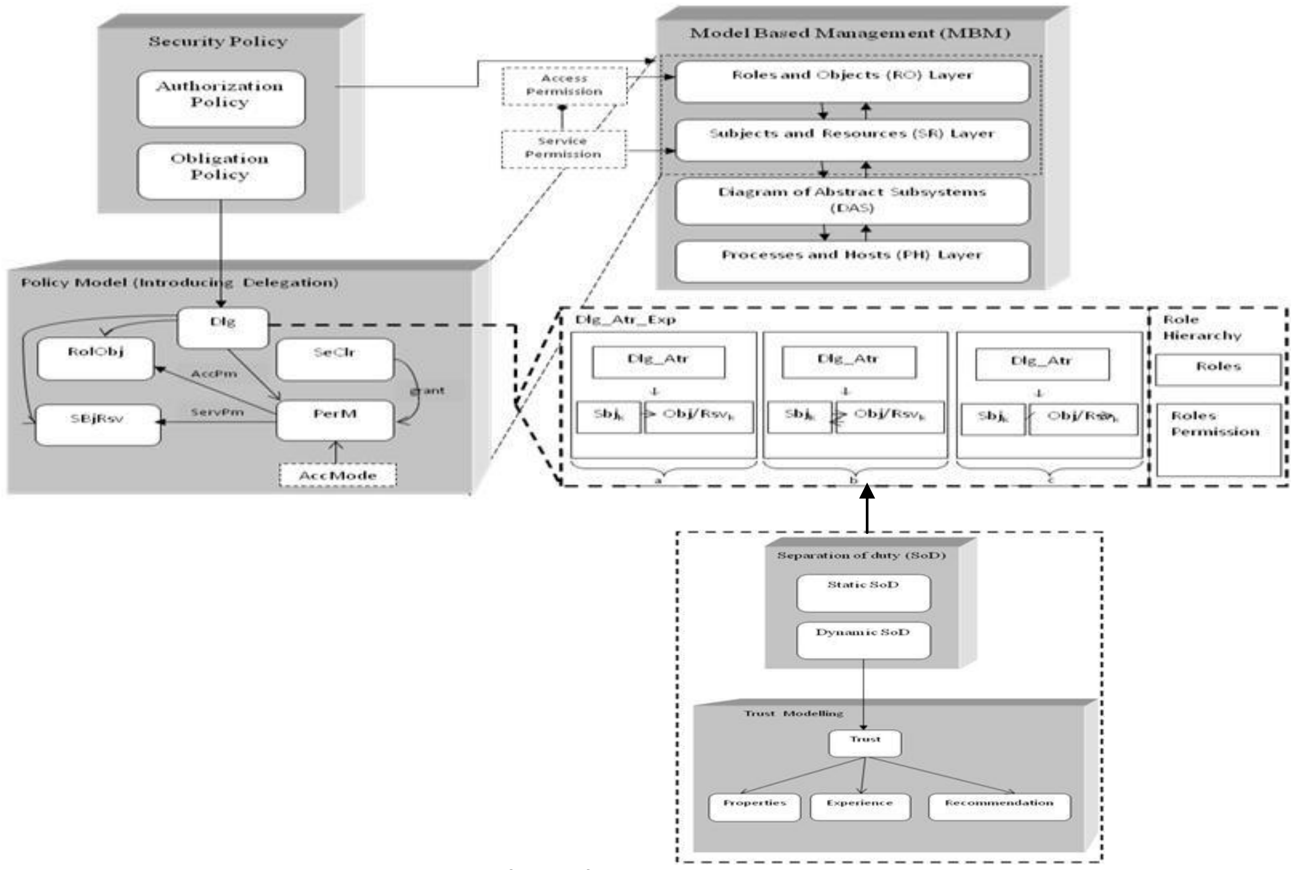

Figure 1.0: - Schematic Architecture of the Information Security Model Designed [18]

DESIGN OF EXPERIMENT, RESULTS AND DISCUSSION

A typical University departmental organization was selected for the experimental design where the following major roles were identified and coded as follows; Head of department (HOD), Examination officer (EO), Undergraduate Lecturing (ULEC), Postgraduate Lecturing (PLEC), Examiners (EXAM), Post Graduate Programme Coordinator (PGPC), Post Graduate Diploma Coordinator (PGDC), Project Coordinator (PRJC), SIWES Coordinator (SIWC), Engineers (ENGR), Technologists (TECH), administrative staff (ADM), and Students (STD), PhD Supervisors (PHSV). Detail of the experiment was reported in [18].

The experimental results obtained from the simulation runs are as presented in the tables below:

Table 1.0: Typical Experimental Result from Delegation Based on Separation of Duty

Simulation Runs	**No of Successful Delegation**	**No of Unsuccessful Delegation**	**Total No of Process Initiated**	**% of Success**
SR1.	1	14	15	6.67
SR2.	3	12	15	20
SR3.	1	14	15	6.67
SR4.	1	14	15	6.67
SR5.	0	15	15	0
Total	**6**	**69**	**75**	**% Average = 8**

Table 2.0: Typical Experimental Result from Delegation Based on Trust

Simulation Runs	**No of Successful Delegation**	**No of Unsuccessful Delegation**	**Total No of Process Initiated**	**% of Success**
SR1.	3	12	15	20
SR2.	5	10	15	33.33
SR3.	1	14	15	6.67
SR4.	3	12	15	20
SR5.	5	10	15	33.33
Total	**17**	**58**	**75**	**% Average = 22.67**

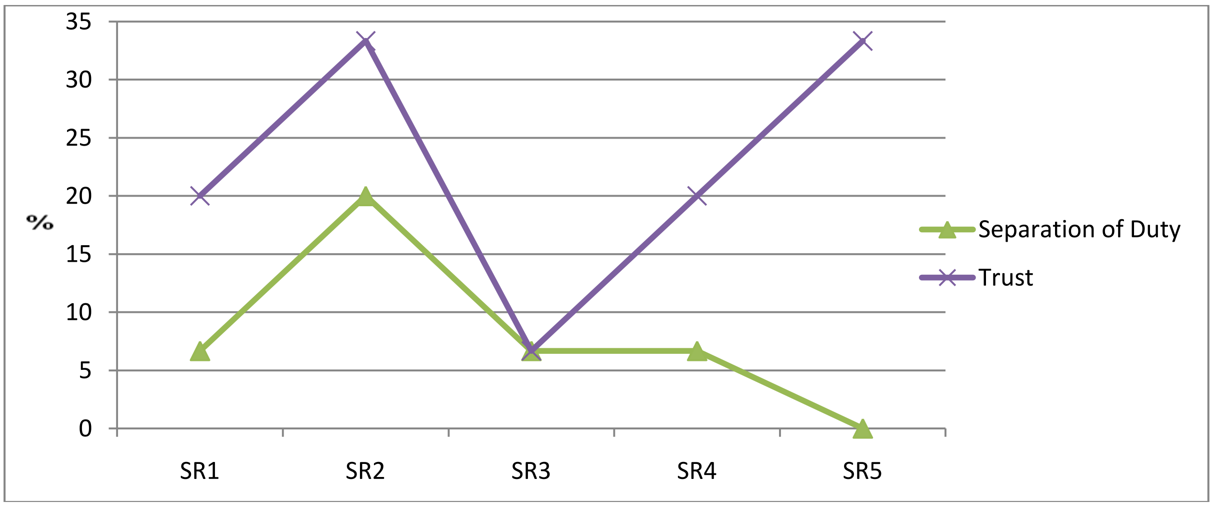

Figure 2.0: Percentage Success from Simulation Results for the delegation Models based SoD and Trust.

Results from the tables above show that percentage average of successful delegation based on separation of duty and trust from our experimental simulation runs were found to be 8% and 22.67% respectively.

Separation of duty especially when both static and dynamic SoD principles are combined was found to play a relatively significant role in ensuring secure delegation processes based on the experimental result. Therefore, logically, the results as shown in Table 1.0 revealed that it may be significantly effective in protecting system resources especially where confidentiality and integrity of the resources are to be ensured since only roles are separated. It generated a fairly low percentage of successes with the last returning a zero. The implication is the fact that, the same role that has been separated from a subject can be re assigned again to the subject from whom it was previously separated under another circumstances thus, it will expose the system vulnerabilities making it more secure.

Trust models on the other hand also record a fairly low percentage average of success (22.67%). This may or may not be objective in assigning roles to subjects in the quest to access objects. Although, experience of subject in occupying a role may be significant in delegation, but the fact that the recommenders may be subjective in giving recommendations about the subjects may also make delegation based on trust to return fairly low percentage.

6. Conclusion

Delegation gives temporary privilege to one or more users so that critical tasks can be completed without breaching security policy. Previous works [1, 2, 3, 22, 23, 24] showed that delegation is a complex problem to solve and is generally modelled separately from other administration requirements. The reason is that previous models were generally based on the RBAC (Role- Based Access Control) model which usually may not be secure enough to deal with all delegation requirements.

We proposed two requirements as a pre-requisite condition for role delegation. These are Separation of Duty (SoD), and Trust. These requirements were used for designing the mathematical structures for the delegation model of the obligation policy. It is of the authors' opinion that delegation of roles through decentralised administration and the specification of separation of duty and trust should bring about a stricter access control in information system security design based on the simulation results using formulated hypothetical data.

Future works will include the formalization and the integration of appropriate models for various revocations methodologies after successful delegation, determination, analysis and resolution of policy(ies) conflicts and integration of appropriate authorization policy into the model to further refine and validate the delegation policy.

7. References

[1] Barka E. and Sandhu R, "A Role-based Delegation Model and Some Extensions", *In proceedings of 16th Annual Computer Security Application Conference*, pp. 168-176, 2000.

[2] Barka E. and Sandhu R, "Role-Based Delegation Model/ Hierarchical Roles (RBDM1)", *In proceedings of the 20th Annual Computer Security Applications Conference (ACSAC'04)*, pp. 396-404, 2004

[3] Crampton J. and Khambhammettu H., "Delegation in Role-Based Access Control", *International Journal of Information Security*, Vol. 7, No. 2, pp. 123-136, 2008.

[4] Chakraborty S. and Ray I., "Integrating Trust Relationships into the RBAC model for access control in open Systems", *In proceeding of the 11th ACM Symposium on Access Control Models and Technologies*, Pages 49-58, 2006.

[5] Clark D. D. and Wilson D. R., "A Comparison of Commercial and Military Computer Security Policies", *IEEE Symposium on Security and Privacy*, Oakland, California, pp. 184-194. 1987

[6] Clark D. D., and D. R. Wilson, "Evolution of a Model for Computer Integrity," in *Report of the Invitational Workshop on Data Integrity*, Z.G. Ruthberg and W.T. Polk (eds.), NIST Special Publication 500-168, Appendix A, 1989.

[7] Ferriaolo David, Cugini Janet, and Kuhn Richard., "Role-based access control (RBAC): Features and motivations". *In Proceedings of 11th Annual Computer Security Application Conference*, pages 241-48, New Orleans, LA, December 11-15 1995.

[8] Gligor V. D., Gavrila S. I., Ferraiolo D., "On the Formal Definition of Separation-of-Duty Policies and their Composition", *IEEE Symposium on Security and Privacy*, 3-6 May 1998, Oakland, California.

[9] Gligor, V. D., S. I. Gavrila, and J. Cugini, "The RBAC Security Policy Model", http://cspa09.ncsl.nist.gov/disk2/rbac/docs/model.ps. 1999.

[10] Hummel, A. A., K. Deinhart, S. Lorenz, V. D. Gligor, "Role-Based Security Administration", *Sicherheit in Informationsystemen* (K. Bauknecht, D. Karagiannis, and S. Teufel (eds.)), vdf Hochschulverlag, ETH Zurich, pp. 69-92, March 1996.

[11] Josang A., "An Algebra for Assessing trust in Certification Chains", *In Proceedings of the Network and Distributed Systems Security Symposium*, Australian. 1999.

[12] Josang A. and Bhuiyan T., "Optimal Trust Network Analysis with Subjective Logic", *In Proceedings of the Second International Conference on Emerging Security Information, Systems and Technologies*. 2008.

[13] Josang A., "Artificial Reasoning with Subjective Logic", *In Proceedings of the 2nd Australian Workshop on CommonSense Reasoning*. 1997.

[14] Josang A, Gray E, and Kinateder M, "Simplification and Analysis of Transitive Trust Networks", *Web Intelligence and Agents Systems*, 4 (2): 139 – 161, 2006.

[15] Ray I., Ray I, and Chakraborty S. (2009), “An Interoperable Context sensitive Model of Trust” *Journal of Intelligent Information Systems*, 32(1): 75 – 104. 2006

[16] Li Ninghui and Wang Qihua, “Beyond Separation of Duty: An Algebra for Specifying High level Security Policies*”, CCS’06 (ACM),* October 30–November 3, Alexandria, Virginia, USA. 2006.

[17] Nash M. J. and K. R. Poland, *“*Some Conundrums Concerning Separation of Duty” *Proceeding of IEEE Symposium on Security and Privacy*, Oakland, California, pp. 201-207. 1990.

[18] Ogundele O.S., “Design of a Multilevel Access Control Models based on Attributes, Separation of duty and Trust*”. PhD Thesis.* Federal University of Technology, Akure, Nigeria, 2011.

[19] Saltzer J.H. and Schroeder M.D., “The Protection of Information in Computer Systems”, *Proceedings of the IEEE 63(9)*: 1278 – 1308, 1975

o, Italy, 2003.

[20] Sloman M. and Lupu E. C., “Security and management policy specification*”. IEEE Network, Special Issue on Policy-Based Networking*, 16(2):10–19, March/April 2002.

[21] Simon R. T., and Zurko M. E., “Separation of Duty in Role-Based Environments,” *Proc. of Computer Security Foundations Workshop X*, Rockport, Massachusetts, June 1997

[22] Toahchoodee M., Xie X., and Ray I., “Towards Trustworthy Delegation in Role Based Access Control Model”, *In Proceedings of ISC 2009 Conference,* Pg 379 – 394, Pisa, Italy, 2009

[23] Ye Chunxiao and Wu Zhongfu and Fu Yunqing, “An Attribute-Based Delegation Model and Its Extension”, *Journal of Research and Practice in Information Technology* Vol. 38, No. 1, 2006.

[24] Zhang, X.W., Oh, S. And Sandhu, R. S., *“*PBDM: A flexible delegation model in RBAC*”. Proceedings of the 8th ACM Symposium on Access Control Models and Technologies (SACMAT’03)*. Com

Received: July 2011. Accepted: Aug. 2011.