Proc. Fourth European Symposium on Research in Computer Security, Rome, Italy, September 25-27, 1996

Role Hierarchies and Constraints for Lattice-Based Access Controls

Ravi Sandhu[1]
George Mason University & SETA Corporation[2]

Abstract Role-based access control (RBAC) is a promising alternative to traditional discretionary and mandatory access controls. In RBAC permissions are associated with roles, and users are made members of appropriate roles thereby acquiring the roles' permissions. In this paper we formally show that lattice-based mandatory access controls can be enforced by appropriate configuration of RBAC components. Our constructions demonstrate that role hierarchies and constraints are required to effectively achieve this result. We show that variations of the lattice-based $\star$-property, such as write-up (liberal $\star$-property) and no-write-up (strict $\star$-property), can be easily accommodated in RBAC. Our results attest to the flexibility of RBAC and its ability to accommodate different policies by suitable configuration of role hierarchies and constraints.

1 INTRODUCTION

Role-based access control (RBAC) has recently received considerable attention as a promising alternative to traditional discretionary and mandatory access controls (see, for example, [FK92, SCY96, SCFY96]). In RBAC permissions are associated with roles, and users are made members of appropriate roles thereby acquiring the roles' permissions. This greatly simplifies management of permissions. Roles are created for the various job functions

[1]This research is partly supported by contract 50-DKNB-5-00188 from the National Institute of Standards and Technology at SETA Corporation, and grant CCR-9503560 from the National Science Foundation at George Mason University.

[2]All correspondence should be addressed to Ravi Sandhu, ISSE Department, MS 4A4, George Mason University, Fairfax, VA 22030, USA. Email: sandhu@isse.gmu.edu, voice: +1 703 993 1659, fax: +1 703 993 1638, URL: http://www.isse.gmu.edu/faculty/sandhu.

in an organization and users are assigned roles based on their responsibilities and qualifications. Users can be easily reassigned from one role to another. Roles can be granted new permissions as new applications and systems are incorporated, and permissions can be revoked from roles as needed.

An important characteristic of RBAC is that by itself it is policy neutral. RBAC is a means for articulating policy rather than embodying a particular security policy (such as one-directional information flow in a lattice). The policy enforced in a particular system is the net result of the precise configuration and interactions of various RBAC components as directed by the system owner. Moreover, the access control policy can evolve incrementally over the system life cycle, and in large systems it is almost certain to do so. The ability to modify policy to meet the changing needs of an organization is an important benefit of RBAC.

Classic lattice-based access control (LBAC) models [San93] on the other hand are specifically constructed to incorporate the policy of one-directional information flow in a lattice.[1] There is nonetheless strong similarity between the concept of a security label and a role. In particular, the same user cleared to say Secret can on different occasions login to a system at Secret and Unclassified levels. In a sense the user determines what role (Secret or Unclassified) should be activated in a particular session.

This leads us naturally to ask whether or not LBAC can be simulated using RBAC. If RBAC is policy neutral and has adequate generality it should indeed be able to do so. Particularly, because the notion of a role and the level of a login session are so similar. This question is theoretically significant because a positive answer would establish that LBAC is just one instance of RBAC thereby relating two distinct access control models that have been developed with different motivations. A positive answer is also practically significant, because it implies that the same Trusted Computing Base can be configured to enforce RBAC in general and LBAC in particular. This addresses the long held desire of multi-level security practitioners that technology which meets needs of the larger commercial marketplace be applicable to LBAC. The classical approach to fulfilling this desire has been to argue that LBAC has applications in the commercial sector. So far this argument has not been terribly productive. RBAC, on the other hand, is specifically motivated by needs of the commercial sector. Its customization to LBAC might be a more productive approach to dual-use technology.

[1] This one-directional information flow can be applied for confidentiality, integrity, confidentiality and integrity together, or for aggregation policies such as Chinese Walls [San93].

In this paper we answer this question positively by demonstrating that several variations of LBAC can be easily accommodated in RBAC by configuring a few RBAC components.[2] We use the family of RBAC models recently developed by Sandhu et al [SCFY96] for this purpose. Our constructions show that the concepts of role hierarchies and constraints are critical to achieving this result. Changes in the role hierarchy and constraints lead to different variations of LBAC. A simulation of LBAC in RBAC has been earlier given by Nyanchama and Osborn [NO96], however, they do not exploit role hierarchies and constraints and cannot handle variations so easily as our constructions of this paper.

The rest of this paper is organized as follows. We review the family of RBAC models due to Sandhu et al [SCFY96] in section 2. This is followed by a quick review of LBAC in section 3. Our simulation of several LBAC variations in RBAC is described in section 4. Section 5 gives our conclusions.

2 RBAC MODELS

A general RBAC model was recently defined by Sandhu et al [SCFY96]. It is summarized in Figure 1.[3] The model is based on three sets of entities called users (U), roles (R), and permissions (P). Intuitively, a user is a human being or an autonomous agent, a role is a job function or job title within the organization with some associated semantics regarding the authority and responsibility conferred on a member of the role, and a permission is an approval of a particular mode of access to one or more objects in the system.

The *user assignment* (UA) and *permission assignment* (PA) relations of Figure 1 are both many-to-many relations. A user can be a member of many roles, and a role can have many users. Similarly, a role can have many

[2] It should be noted that RBAC will only prevent overt flows of information. This is true of any access control model, including LBAC. Information flow contrary to the one-directional requirement in a lattice by means of so-called covert channels is outside the purview of access control per se. Neither LBAC nor RBAC addresses the covert channel issue directly. Techniques used to deal with covert channels in LBAC can be used for the same purpose in RBAC.

[3] Figure 1 shows the $RBAC_3$ model which is the most general among the family of models described in [SCFY96]. The administrative model of [SCFY96] is not relevant here. For our purpose we assume a single security officer is the only one who can configure various components of RBAC.

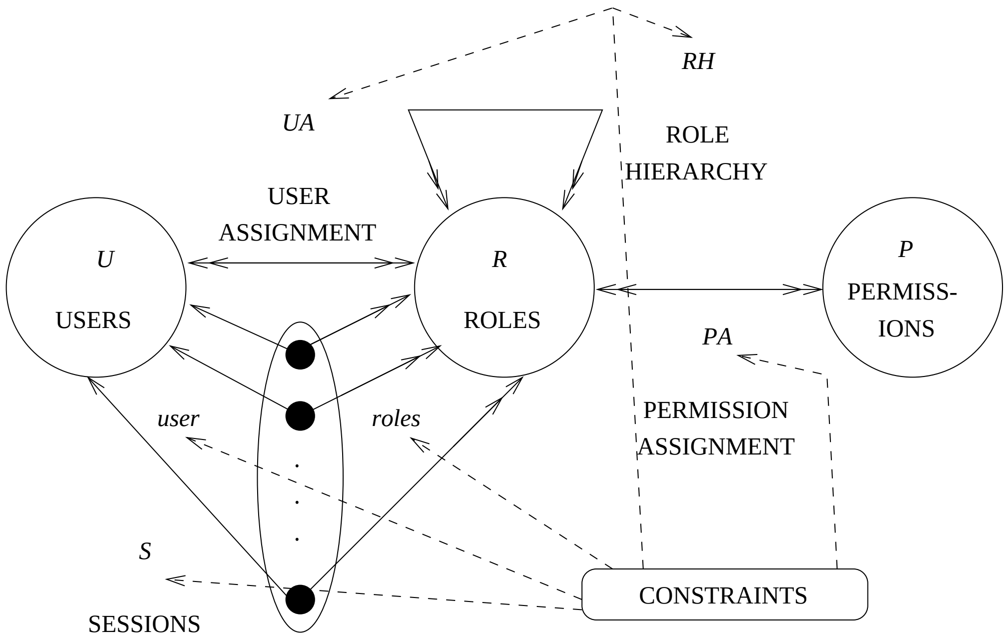


Figure 1: The RBAC Model

permissions, and the same permission can be assigned to many roles. There is a partially ordered *role hierarchy* RH, also written as $\geq$, where $x \geq y$ signifies that role x inherits the permissions assigned to role y. Inheritance along the role hierarchy is transitive and multiple inheritance is allowed in partial orders.

Figure 1 shows a set of sessions S. Each session relates one user to possibly many roles. Intuitively, a user establishes a session during which the user activates some subset of roles that he or she is a member of (directly or indirectly by means of the role hierarchy). The double-headed arrow from a session to R indicates that multiple roles can be simultaneously activated. The permissions available to the user are the union of permissions from all roles activated in that session. Each session is associated with a single user, as indicated by the single-headed arrow from the session to U. This association remains constant for the life of a session. A user may have multiple sessions open at the same time, each in a different window on the workstation screen for instance. Each session may have a different combination of active roles. The concept of a session equates to the traditional notion of a *subject* in access control. A subject (or session) is a unit of access control, and a user may have multiple subjects (or sessions) with different permissions active at the same time.

Finally, Figure 1 shows a collection of *constraints*. Constraints can apply to any of the preceding components. An example of constraints is mutually disjoint roles, such as purchasing manager and accounts payable manager, where the same user is not permitted to be a member of both roles.

The following definition formalizes the above discussion.

Definition 1 (RBAC Model) *The RBAC model has the following components:*

- U, R, P, *and* S, *sets of users, roles, permissions and sessions respectively,*
- $PA \subseteq P \times R$, *a many-to-many permission (to role) assignment relation,*
- $UA \subseteq U \times R$, *a many-to-many user (to role) assignment relation,*
- $RH \subseteq R \times R$, *a partially ordered role hierarchy (written as* $\geq$ *in infix notation),*

- *user* : $S \to U$, *a function mapping each session* s_i *to the single user* $user(s_i)$ *(constant for the session's lifetime),*

- *roles* : $S \to 2^R$ *a function mapping each session* s_i *to a set of roles* $roles(s_i) \subseteq \{r \mid (\exists r' \geq r)[(user(s_i), r') \in UA]\}$ *(which can change with time) so that session* s_i *has the permissions* $\cup_{r \in roles(s_i)} \{p \mid (\exists r'' \leq r)[(p, r'') \in PA]\}$, *and*

- *a collection of* constraints *that determine whether or not values of various components of the RBAC model are acceptable (only acceptable values will be permitted).* 2

3 LBAC MODELS

Lattice based access control (LBAC) is concerned with enforcing one directional information flow in a lattice of security labels. LBAC is also known as mandatory access control (MAC) or multilevel security.[4] Depending upon the nature of the lattice the one-directional information flow enforced by LBAC can be applied for confidentiality, integrity, confidentiality and integrity together, or for aggregation policies such as Chinese Walls [San93]. There are also variations of LBAC where the one-directional information flow is partly relaxed to achieve selective downgrading of information or for integrity applications [Bel87, Lee88, Sch88].

The mandatory access control policy is expressed in terms of security labels attached to subjects and objects. A label on an object is called a *security classification*, while a label on a user is called a *security clearance*. It is important to understand that a Secret user may run the same program, such as a text editor, as a Secret subject or as an Unclassified subject. Even though both subjects run the same program on behalf of the same user, they obtain different privileges due to their security labels. It is usually assumed that the security labels on subjects and objects, once assigned, cannot be changed (except by the security officer). This last assumption, that security labels do not change, is known as *tranquility*.[5] The security labels form a

[4]LBAC is typically applied in addition to classical discretionary access controls (DAC) [SS94] but for our purpose we will focus only on the MAC component. DAC can be accommodated in RBAC as an independent access control policy just as it is done in LBAC.

[5]Non-tranquil LBAC can also be simulated in RBAC but is outside the scope of this paper.

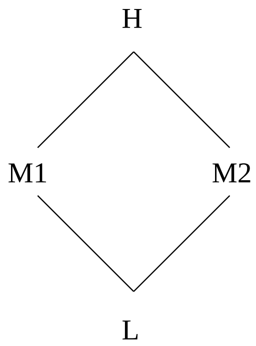


Figure 2: A Partially Ordered Lattice

lattice structure as defined below.

Definition 2 (Security Lattice) *There is a finite lattice of security labels $\mathcal{SC}$ with a partially ordered dominance relation $\geq$ and a least upper bound operator.*[6] 2

An example of a security lattice is shown in Figure 2. Information is only permitted to flow upward in the lattice. In this example, H and L respectively denote high and low, and M1 and M2 are two incomparable labels intermediate to H and L. This is a typical confidentiality lattice where information can flow from low to high but not vice versa.

The specific mandatory access rules usually specified for a lattice are as follows, where λ signifies the security label of the indicated subject or object.

Definition 3 (Simple Security) *Subject s can read object o only if $\lambda(s) \geq \lambda(o)$.* 2

Definition 4 (Liberal $\star$-property) *Subject s can write object o only if $\lambda(s) \leq \lambda(o)$.* 2

The $\star$-property is pronounced as the star-property.

[6] The least upper bound operator is not relevant to our constructions which apply to partially ordered security labels in general.

For integrity reasons sometimes a stricter form of the $\star$-property is stipulated. The liberal $\star$-property allows a low subject to write a high object. This means that high data may be maliciously destroyed or damaged by low subjects. To avoid this possibility we can employ the strict $\star$-property given below.

Definition 5 (Strict $\star$-property) *Subject s can write object o only if $\lambda(s) = \lambda(o)$.* 2

The liberal $\star$-property is also referred to as write-up and the strict $\star$-property as non-write-up or write-equal.

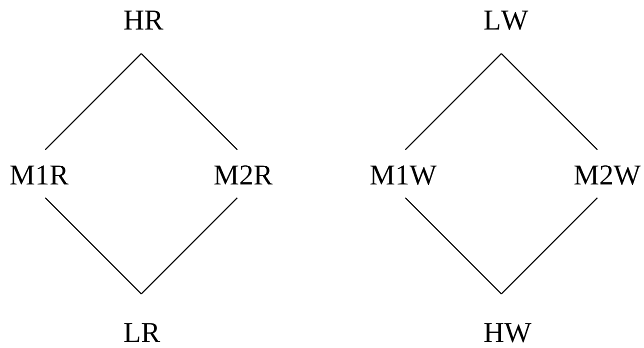
In variations of LBAC the simple-security property is usually left unchanged as we will do in all our examples. Variations of the $\star$-property in LBAC whereby the one-directional information flow is partly relaxed to achieve selective downgrading of information or for integrity applications [Bel87, Lee88, Sch88] will be considered later.

4 CONFIGURING RBAC FOR LBAC

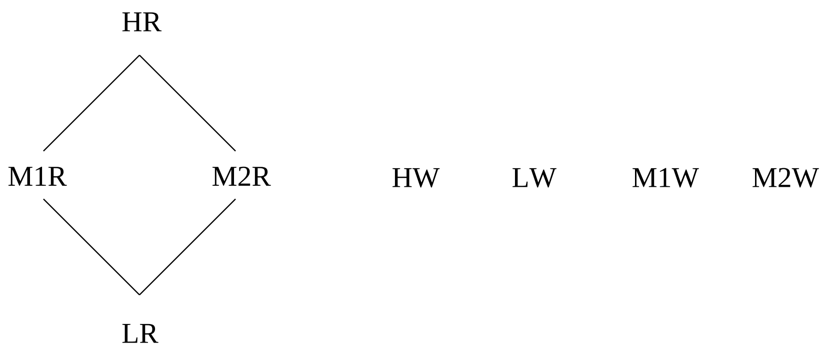
We now show how different variations of LBAC can be simulated in RBAC. It turns out that we can achieve this by suitably changing the role hierarchy and defining appropriate constraints. This suggests that role hierarchies and constraints are central to defining policy in RBAC.

4.1 A Basic Lattice

We begin by considering the example lattice of Figure 2 with the liberal $\star$-property. Subjects with labels higher up in the lattice have more power with respect to read operations but have less power with respect to write operations. Thus this lattice has a dual character. In role hierarchies subjects (sessions) with roles higher in the hierarchy always have more power than those with roles lower in the hierarchy. To accommodate the dual character of a lattice for LBAC we will use two dual hierarchies in RBAC, one for read and one for write. These two role hierarchies for the lattice of Figure 2 are shown in Figure 3(a). Each lattice label x is modeled as two roles xR and xW for read and write at label x respectively. The relationship among the four read roles and the four write roles is respectively shown on the left



(a) Liberal ⋆-Property



(b) Strict ⋆-Property

Figure 3: Role Hierarchies for the Lattice of Figure 2

and right hand sides of Figure 3(a). The duality between the left and right lattices is obvious from the diagrams.

To complete the construction we need to enforce appropriate constraints to reflect the labels on subjects in LBAC. Each user in LBAC has a unique security clearance. This is enforced by requiring that each user in RBAC is assigned to exactly two matching roles xR and xW. An LBAC user can login at any label dominated by the user's clearance. This requirement is captured in RBAC by requiring that each session has exactly two matching roles yR and yW. The condition that x $\geq$ y, that is the user's clearance dominates the label of any login session established by the user, is not explicitly required because it is directly imposed by the RBAC model anyway.

LBAC is enforced in terms of read and write operations. In RBAC this means our permissions are read and writes on individual objects written as (o,r) and (o,w) respectively. An LBAC object has a single sensitivity label associated with it. This is expressed in RBAC by requiring that each pair of permissions (o,r) and (o,w) be assigned to exactly one matching pair of xR and xW roles respectively. By assigning permissions (o,r) and (o,w) to roles xR and xW respectively, we are implicitly setting the sensitivity label of object o to x.

The above construction is formalized below.

Example 1 (Liberal $\star$-Property)

- *R = {HR, M1R, M2R, LR, HW, M1W, M2W, LW}*
- *RH as shown in Figure 3(a)*
- *P = {(o,r), (o,w) | o is an object in the system}*
- *Constraint on UA: Each user is assigned to exactly two roles xR and LW*
- *Constraint on sessions: Each session has exactly two roles yR and yW*
- *Constraints on PA:*
  - *(o,r) is assigned to xR iff (o,w) is assigned to xW*
  - *(o,r) is assigned to exactly one role xR* 2

The set of permissions P remains the same in all our examples so we will omit its explicit definition in subsequent examples.

Variations in LBAC can be accommodated by modifying this basic construction in different ways. In particular, the strict $\star$-property retains the hierarchy on read roles but treats write roles as incomparable to each other as shown in Figure 3(b).

Example 2 (Strict $\star$-Property) *Identical to example 1 except RH is as shown in Figure 3(b).* 2

Now the permission (o,w) is no longer inherited by other roles as is the case in example 1.

4.2 Lattice with Trusted Write Range

Next we consider a version of LBAC in which subjects are given more power than allowed by the simple security and $\star$-properties [Bel87]. The basic idea is to allow subjects to violate the $\star$-property in a controlled manner. This is achieved by associating a pair of security labels λ_r and λ_w with each subject (objects still have a single security label). The simple security property is applied with respect to λ_r and the liberal $\star$-property with respect to λ_w. In the LBAC model of [Bel87] it is required that λ_r should dominate λ_w. With this constraint the subject can read and write in the range of labels between λ_r and λ_w which is called the *trusted range.* If λ_r and λ_w are equal the model reduces to the usual LBAC model with the trusted range being a single label.

The preceding discussion is remarkably close to our RBAC constructions. The two labels λ_r and λ_w correspond directly to the two roles xR and yW we have introduced earlier. The dominance required between λ_r and λ_w is trivially recast as a dominance constraint between x and y. This leads to the following example.

Example 3 (Liberal $\star$-Property with Trusted Range) *Identical to example 1 except*

- *Constraint on UA: Each user is assigned to exactly two roles xR and yW such that $x \geq y$ in the original lattice*
- *Constraint on sessions: Each session has exactly two roles xR and yW such that $x \geq y$ in the original lattice* 2

Lee [Lee88] and Schockley [Sch88] have argued that the Clark-Wilson integrity model [CW87] can be supported using LBAC. Their models are similar to the above except that no dominance relation is required between x and y. Thus the write range may be completely disjoint with the read range of a subject. This is easily expressed in RBAC as follows.

Example 4 (Liberal $\star$-Property with Independent Write Range) *Identical to example 3 except $x \geq y$ is not required in the constraint on UA and the constraint on sessions.* 2

A variation of the above is to use the strict $\star$-property as follows.

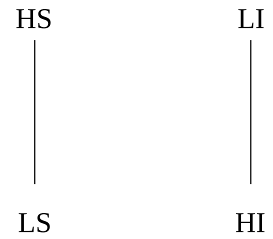
Example 5 (Strict $\star$-Property with Designated Write) *Identical to example 2 except*

- *Constraint on UA: Each user is assigned to exactly two roles xR and yW*
- *Constraint on sessions: Each session has exactly two roles xR and yW* 2

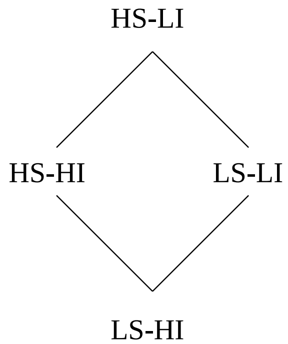
Example 5 can also be directly obtained from example 4 by requiring the strict $\star$-property instead of the liberal $\star$-property. Example 5 can accommodate Clark-Wilson transformation procedures as outlined by Lee and Schockley. (Lee and Schockley actually use the liberal $\star$-property in their construction, but their lattices are such that the construction is more directly expressed by example 5.)

4.3 Independent Confidentiality and Integrity Roles

Next we turn our attention to integrity lattices and their interaction with confidentiality lattices. LBAC was first formulated for confidentiality purposes. It was subsequently observed that if high integrity is at the top of the lattice and low integrity at the bottom then information flow should be downward rather than upward (as in confidentiality lattices). In [San93] it is argued that it is simpler to fix the direction of information flow and put high integrity at the bottom and low integrity at the top in integrity lattices. Because the confidentiality models were developed earlier we might as well stay with lattices in which information flow is always upwards.



(a) Two Independent Lattices



(b) One Composite Lattice

Figure 4: Confidentiality and Integrity Lattices

Figure 3(a) shows two independent lattices. The one on the left has HS (high secrecy) on the top and LS (low secrecy) on the bottom. The one on the right has LI (low integrity) on the top and HI (high integrity) on the bottom. In both lattices information flow is upward. The two lattices can be combined into the single composite lattice shown in Figure 3(b).[7]

One complication in combining confidentiality and integrity lattices (or multiple lattices in general) is that these lattices may be using different versions of the $\star$-property. We have discussed earlier that the strict $\star$-property is often used in confidentiality lattices due to integrity considerations. In integrity lattices there is no similar need to use the strict $\star$-property, and one would expect to see the liberal $\star$-property instead.

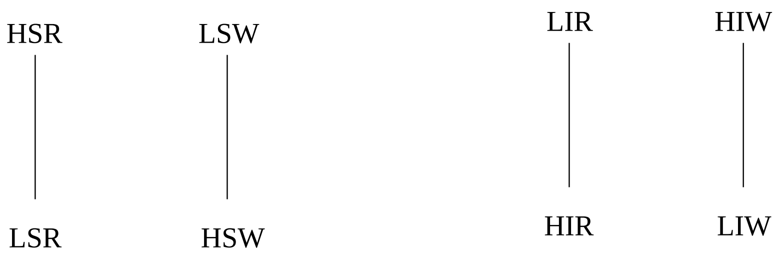
In order to accommodate different versions of the $\star$-property for the two lattices we could keep two distinct lattices as shown in Figure 3(a). We know how to recast each lattice in RBAC with liberal or strict $\star$-properties as appropriate. Three of these combinations[8] are shown in Figure 5 and described formally below.

Example 6 (Liberal Confidentiality and Liberal Integrity $\star$-Property)

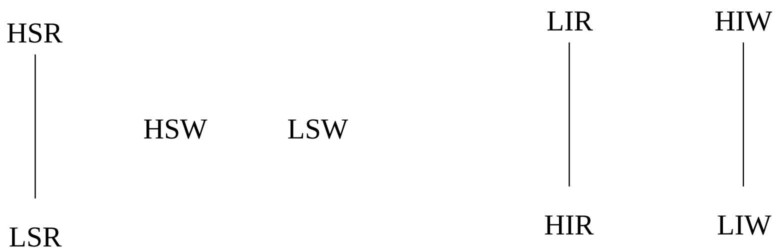
- *R = {HSR, LSR, LSW, HSW, LIR, HIR, HIW, LIW}*
- *RH as shown in Figure 5(a)*
- *Constraint on UA: Each user is assigned to exactly two pairs of roles xSR, xSW and yIR, yIW*
- *Constraint on sessions: Each session has exactly two pairs of roles xSR, xSW and yIR, yIW*
- *Constraints on PA:*
  - *(o,r) is assigned to xSR iff (o,w) is assigned to xSW*
  - *(o,r) is assigned to exactly one role xSR*
  - *(o,r) is assigned to yIR iff (o,w) is assigned to yIW*
  - *(o,r) is assigned to exactly one role yIR* 2

[7] It is always possible to mathematically combine multiple lattices into a single lattice.

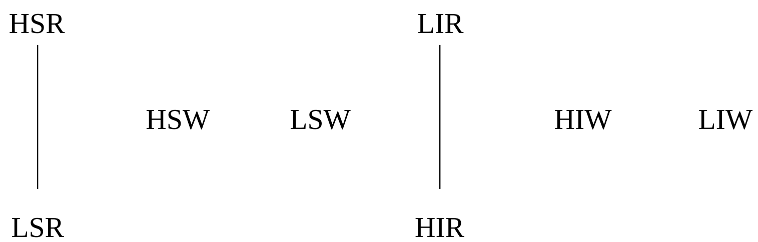
[8] The fourth combination of liberal confidentiality and strict integrity could be easily constructed but is rather unlikely to be used in practice so is omitted.



(a) Liberal Confidentiality and Liberal Integrity

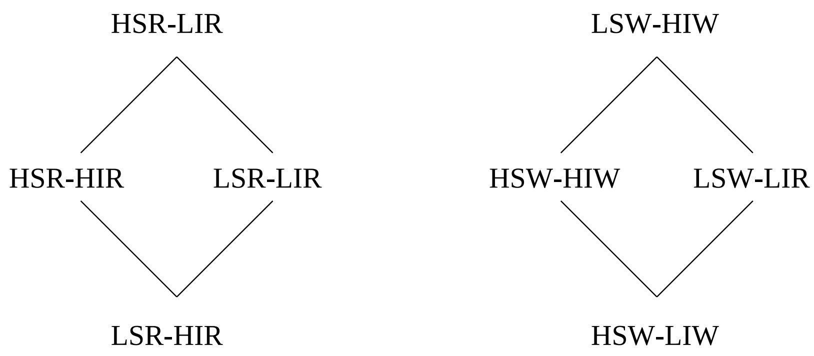


(b) Strict Confidentiality and Liberal Integrity

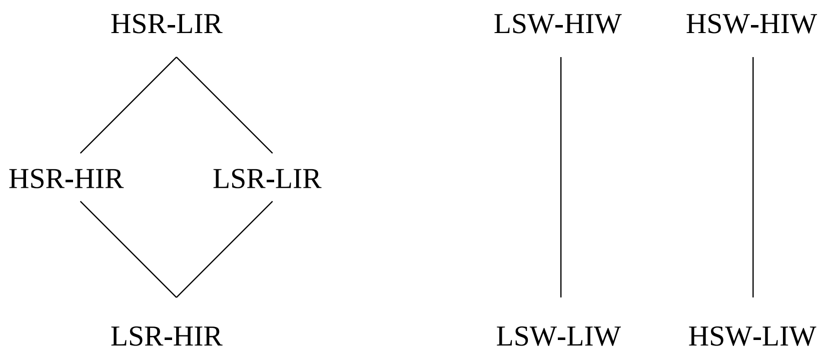


(c) Strict Confidentiality and Strict Integrity

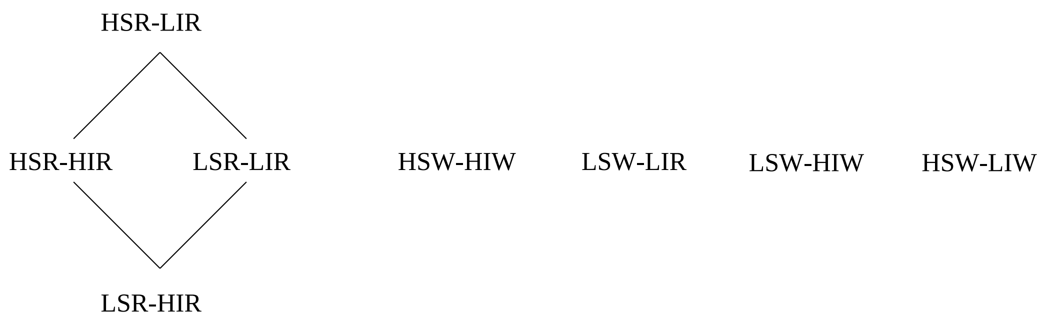
Figure 5: Independent Confidentiality and Integrity Roles



(a) Liberal Confidentiality and Liberal Integrity



(b) Strict Confidentiality and Liberal Integrity



(c) Strict Confidentiality and Strict Integrity

Figure 6: Composite Confidentiality and Integrity Roles

Example 7 (Strict Confidentiality and Liberal Integrity $\star$-Property) *Identical to example 6 except that RH is as shown in Figure 5(b).* 2

Example 8 (Strict Confidentiality and Strict Integrity $\star$-Property) *Identical to example 6 except that RH is as shown in Figure 5(c).* 2

4.4 Composite Confidentiality and Integrity Roles

The preceding constructions require each user and session to have a pair of roles for each lattice. We now show how the same results can be achieved by a single pair of roles. Consider the composite lattice of Figure 3(b). Since the simple security property does not change we have a similar role hierarchy for the read roles shown on the left hand side of the three role hierarchies of Figures 6(a), (b) and (c). In each case the hierarchy for the write roles needs to be adjusted as shown on the right hand side of each of these figures. The constructions are formally described below.

Example 9 (Liberal Confidentiality and Liberal Integrity $\star$-Property)

- *R = {HSR-LIR, HSR-HIR, LSR-LIR, LSR-HIR, HSW-LIW, HSW-HIW, LSW-LIW, LSW-HIW}*
- *RH as shown in Figure 6(a)*
- *Constraint on UA: Each user is assigned to exactly two roles xSR-yIR and xSW-yIW*
- *Constraint on sessions: Each session has exactly two roles uSR-vIR and uSW-vIW*
- *Constraints on PA:*
  - *(o,r) is assigned to xSR-yIR iff (o,w) is assigned to xSW-yIW*
  - *(o,r) is assigned to exactly one role xSR-yIR* 2

Example 10 (Strict Confidentiality and Liberal Integrity $\star$-Property) *Identical to example 9 except that RH is as shown in Figure 6(b).* 2

Example 11 (Strict Confidentiality and Strict Integrity $\star$-Property) *Identical to example 9 except that RH is as shown in Figure 6(c).* 2

The constructions indicate how a single pair of roles can accommodate lattices with different variations of the $\star$-property. The construction can clearly be generalized to more than two lattices.

5 CONCLUSION

In this paper we have shown how different variations of lattice based access controls (LBAC) can be simulated in role-based access control (RBAC), specifically using the models developed by Sandhu et al [SCFY96]. RBAC is itself policy neutral but can be easily configured to specify a variety of policies as we have shown. The main components of RBAC that need to be adjusted for different LBAC variations are the role hierarchy and constraints. This attests to the flexibility and power of RBAC.

A practical consequence of our results is that it might be better to develop systems that support general RBAC and specialize these to LBAC. RBAC has much broader applicability than LBAC, especially in the commercial sector. LBAC can be realized as a particular instance of RBAC. This approach provides the added benefit of greater flexibility for LBAC, for which we have seen there are a number of variations of practical interest. In LBAC systems these variations so far require the rules to be adjusted in the implementation. RBAC provides for adjustment by configuration of role hierarchies and constraints instead.

References

[Bel87] D.E. Bell. Secure computer systems: A network interpretation. In *Third Annual Computer Security Application Conference*, pages 32–39, 1987.

[CW87] D.D. Clark and D.R. Wilson. A comparison of commercial and military computer security policies. In *Proceedings IEEE Computer Society Symposium on Security and Privacy*, pages 184–194, Oakland, CA, May 1987.

[FK92] David Ferraiolo and Richard Kuhn. Role-based access controls. In *15th NIST-NCSC National Computer Security Conference*, pages 554–563, Baltimore, MD, October 13-16 1992.

[Lee88] T.M.P. Lee. Using mandatory integrity to enforce "commercial" security. In *Proceedings IEEE Computer Society Symposium on Security and Privacy*, pages 140–146, Oakland, CA, May 1988.

[NO96] Matunda Nyanchama and Sylvia Osborn. Modeling mandatory access control in role-based security systems. In *Database Security VIII: Status and Prospects*. To appear, 1996.

[San93] Ravi S. Sandhu. Lattice-based access control models. *IEEE Computer*, 26(11):9–19, November 1993.

[SCFY96] Ravi S. Sandhu, Edward J. Coyne, Hal L. Feinstein, and Charles E. Youman. Role-based access control models. *IEEE Computer*, 29(2):38–47, February 1996.

[Sch88] W.R. Schockley. Implementing the clark/wilson integrity policy using current technology. In *NIST-NCSC National Computer Security Conference*, pages 29–37, 1988.

[SCY96] Ravi Sandhu, Ed Coyne, and Charles Youman, editors. *Proceedings of the 1st ACM Workshop on Role-Based Access Control.* ACM, 1996.

[SS94] Ravi S. Sandhu and Pierangela Samarati. Access control: Principles and practice. *IEEE Communications*, 32(9):40–48, 1994.