CERIAS Tech Report 2005-97

AN ANALYSIS OF EXPRESSIVENESS AND DESIGN ISSUES FOR THE GENERALIZED TEMPORAL ROLE-BASED ACCESS CONTROL MODEL

by James B.D. Joshi , Elisa Bertino,and Arif Ghafoor

Center for Education and Research in
Information Assurance and Security,
Purdue University, West Lafayette, IN 47907-2086

An Analysis of Expressiveness and Design Issues for the Generalized Temporal Role-Based Access Control Model

James B.D. Joshi, *Member*, *IEEE*, Elisa Bertino, *Fellow*, *IEEE*, and Arif Ghafoor, *Fellow*, *IEEE*

Abstract—The Generalized Temporal Role-Based Access Control (GTRBAC) model provides a comprehensive set of temporal constraint expressions which can facilitate the specification of fine-grained time-based access control policies. However, the issue of the expressiveness and usability of this model has not been previously investigated. In this paper, we present an analysis of the expressiveness of the constructs provided by this model and illustrate that its constraints-set is not minimal. We show that there is a subset of GTRBAC constraints that is sufficient to express all the access constraints that can be expressed using the full set. We also illustrate that a nonminimal GTRBAC constraint set can provide better flexibility and lower complexity of constraint representation. Based on our analysis, a set of design guidelines for the development of GTRBAC-based security administration is presented.

Index Terms—Role-based access control, temporal constraint, expressiveness analysis, minimality.

1 Introduction

With the development of Internet-based services and applications proceeding at unprecedented speed, prudent security mechanisms are needed if the full potential of the Internet is to be exploited [3], [13]. Central to a security system is the access control mechanism. Robust access control techniques are needed for alleviating security concerns and raising the trust level for Internet-based services and applications [3]. Role-based access control (RBAC) approaches have emerged as an attractive solution for implementing security mechanisms for organizations with diverse security requirements [10], [15], [21], [26]. Such approaches can provide a viable alternate to the traditional discretionary and mandatory access control (DAC and MAC) techniques [13], [15], [22], [25], [27]. Several valuable features associated with RBAC models, such as their policy neutrality, support for enforcing the principle of least privilege, and efficiency [12], [15], [26], make them an ideal candidate for developing broad range of access control policies. In particular, for heterogeneous multidomain environments, such as the Internet [3], [11], [13], [15], [23], [28], RBAC models provide an elegant framework for secure interoperation.

Context is an important aspect in any access control management. In particular, time plays a key role in managing time-sensitive accesses. A leading example is the workflow management systems, where tasks generally have critical requirements in terms of their invocation/completion deadlines. For such applications, time-based access control techniques are desirable [2], [4], [5], [6], [17]. To address this issue, a Generalized Temporal RBAC (GTRBAC) model has been proposed in [17]. The model allows specification of a broad range of temporal constraints, an important feature that orthogonally applies to all aspects of an RBAC system, such as enabling and activation of roles, user-role, and role-permission assignments. In particular, the GTRBAC model distinguishes between role *enabling* and role *activation*. A role is *enabled* if a user is allowed to acquire its assigned permissions. An *enabled* role becomes *active* when a user acquires its permissions during a session. This is in contrast to a *disabled* role, which cannot be activated by any user. Therefore, constraints on enabling/disabling of roles specify when roles can/cannot be assumed by users. Implementation of an XML-based framework to support GTRBAC policy specification and enforcement has been reported in [7], [8], [14].

The GTRBAC model also includes the following three types of hierarchies [16], [17]: *inheritance-only* hierarchy (*I*-hierarchy) that only allows permission-inheritance semantics, *activation-only* hierarchy (*A*-hierarchy) that only allows role activation semantics, and *inheritance-activation* hierarchy (*IA*-hierarchy) that allows both permission-inheritance and role-activation semantics. In the presence of timing constraints on various entities, the separation of the *permission-inheritance* and the *role-activation* semantics associated with the three types of hierarchies provide a basis for capturing various time-based inheritance semantics of these hierarchies. These temporal hierarchies can further be divided into *restricted* and *unrestricted* types [16], [17]. Separation of inheritance and activation semantics is necessary for capturing lattice-based policies using RBAC [16], [24].

- *J.B.D. Joshi is with the Department of Information Sciences and Telecommunications, University of Pittsburgh, Pittsburgh, PA 15260. E-mail: jjoshi@mail.sis.pitt.edu.*
- *E. Bertino is with the Department of Computer Science, Purdue University, West Lafayette, IN 47907. E-mail: bertino@cerias.purdue.edu.*
- *A. Ghafoor is with the School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN 47907. E-mail: ghafoor@purdue.edu.*

Manuscript received 22 Apr. 2004; revised 8 Feb. 2005; accepted 23 Feb. 2005; published online 3 June 2005.
For information on obtaining reprints of this article, please send e-mail to: tdsc@computer.org, and reference IEEECS Log Number TDSC-0059-0404.

An open issue for any model with a rich constraint language is its expressiveness and minimality. The latter indicates whether or not the set of constraints specified by the model is minimal. Minimality is a crucial criterion for determining whether a nonminimal model provides any practical benefits over the minimal model. Nevertheless, a nonminimal model may provide better flexibility and more benefits in terms of complexity and usability. Given the large variety of RBAC constraint languages recently proposed, their expressive power and minimality are becoming important issues [1], [5], [9], [17], [18], [19]. Furthermore, there is a growing emphasis on developing RBAC models that use a generic set of context and content based attributes to capture complex access control requirements [9], [17], [18], [19]. In this paper, we present an analytical framework for addressing the issue of expressiveness and minimality of constraint languages for RBAC. In particular, we cast our analysis in the framework of the GTRBAC model since this model has a rich constraint language. It can be noticed that the set of constraints for the GTRBAC model is not minimal [17]. One of the contributions of this paper is to prove the existence of a minimal model that has a subset of constraint types defined in the original GTRBAC model and has the same expressive power as the original model possesses. We also illustrate how various sets of different constraint types can be used to generate a family of GTRBAC models having the same expressive power. However, an important issue, as mentioned earlier, is to determine whether or not having a nonminimal set of constraints in the GTRBAC model is of any benefit. In particular, we show that a nonminimal GTRBAC model offers several advantages in terms of complexity of specification and usability. Usability of the model is informally expressed as manageability and convenience in the specification of the access control policy. It also refers to maintaining a clear semantics among the constraints. For example, as we show in this paper, timing constraints on user-role assignments may be replaced by temporal constraints on role enabling to enforce the same policy. However, doing so may alter the original semantics. In this example, for instance, the original meaning of "a user is scheduled to assume a particular role in a given interval of time" is changed to the new semantics which says "the role is enabled in the given interval of time." Through detailed analysis, we provide a set of design guidelines for constructing better representation of GTRBAC policies. The results presented in this paper can lead to the development of efficient tools that can facilitate effective administration of the GTRBAC-based policies.

The approach presented in this paper for analyzing the expressiveness and minimality of the GTRBAC constraint model has a much broader significance as this approach can be used to evaluate any nonminimal specification model in terms of its complexity and flexibility. Flexible context-based access control models are critical for emerging Internet-based applications and have already been introduced in numerous COTS products, such as Oracle, which provides some functionality to handle contextual information in access control decision. The proposed analysis can be used to derive policy design guidelines for developing semantically clear and less complex policies. To the best of our knowledge, no previous work has addressed the issue of minimality of an RBAC model and has analyzed its complexity and usability.

The paper is organized as follows: In Section 2, we briefly overview the GTRBAC model as well as various temporal role hierarchies. In Section 3, we analyze the expressiveness of the GTRBAC model and present the minimality results that lead to a set of constraint design guidelines. We present related work and conclusions in Sections 4 and 5, respectively.

2 GTRBAC Model and Temporal Role Hierarchy

2.1 GTRBAC Model

The GTRBAC model supports a separate notion of role enabling and role activation and provides constraint and event expressions associated with them [17]. An enabled role indicates that a valid user can activate it, whereas an activated role indicates that at least one user has activated the role. The model allows specification of the following set of constraints:

1. *Temporal constraints on role enabling/disabling*. These constraints allow specification of intervals and durations in which a role is enabled. When a duration constraint is specified, the enabling/disabling of a role is initiated by a constraint enabling event that results either from the firing of a trigger or through an administrator initiated runtime event.
2. *Temporal constraints on user-role and role-permission assignments*. These constraints allow the specification of intervals and durations in which a user or permission is assigned to a role.
3. *Activation constraints*. These constraints allow the specification of restrictions on the activation of a role. These include, for example, specifying the total duration for which a user may activate a role or the number of concurrent activations of the role at a particular time.
4. *Runtime events*. A set of runtime events allows the administrator to dynamically initiate the GTRBAC events, or enable the duration or activation constraints. Another set of runtime events allows users to request the activation or deactivation of a role.
5. *Constraint enabling expressions*. The GTRBAC model includes events that enable or disable the aforementioned duration and role activation constraints.
6. *Triggers*. The GTRBAC triggers allow expressing dependencies among events.

Table 1 summarizes different types of constraints and expressions of the GTRBAC model. The periodic expression used in the constraint expressions is of the form (I, P) [20], where P is an *expression* denoting an infinite set of periodic time instants, and $I = [\texttt{begin}, \texttt{end}]$ is a time interval denoting the lower and upper bounds that are imposed on instants in P [20]. The function $Sol(I, P)$ is used to

TABLE 1
GTRBAC Constraint Expressions

Constraint categories	Constraints		Expression	Set/Type
Periodicity Constraint	User-role assignment		$(I, P, pr$:assign$_U$/deassign$_U$ r to $u)$	C_{Urp}
	Role enabling		$(I, P, pr$:enable/disable $r)$	C_{Rp}
	Role-permission assignment		$(I, P, pr$:assign$_P$/deassign$_P$ p to $r)$	C_{PRp}
Duration Constraints	User-role assignment		$([(I, P)\mid D], D_U, pr$:assign$_U$/deassign$_U$ r to $u)$	C_{Urd}
	Role enabling		$([(I, P)\mid D], D_R, pr$:enable/disable $r)$	C_{Rd}
	Role-permission assignment		$([(I, P)\mid D], D_P, pr$:assign$_P$/deassign$_P$ p to $r)$	C_{PRd}
Duration Constraints on Role Activation	Total active role duration	Per-role	$([(I, P)\mid D], D_{active}, [D_{default}], pr$:active$_{R_total}$ $r)$	C^a_{dr}
		Per-user-role	$([(I, P)\mid D], D_{uactive}, u, pr$:active$_{UR_total}$ $r)$	C^a_{dur}
	Max role duration per activation	Per-role	$([(I, P)\mid D], D_{max}, pr$:active$_{R_max}$ $r)$	C^a_{mr}
		Per-user-role	$([(I, P)\mid D], D_{umax}, u, pr$:active$_{UR_max}$ $r)$	C^a_{mur}
Cardinality Constraint on Role Activation	Total no. of activations	Per-role	$([(I, P)\mid D], N_{active}, [N_{default}], pr$:active$_{R_n}$ $r)$	C^a_{nr}
		Per-user-role	$([(I, P)\mid D], N_{uactive}, u, pr$:active$_{UR_n}$ $r)$	C^a_{nur}
	Max. no. of concurrent activations	Per-role	$([(I, P)\mid D], N_{max}, [N_{default}], pr$:active$_{R_con}$ $r)$	C^a_{nmr}
		Per-user-role	$([(I, P)\mid D], N_{umax}, u, pr$:active$_{UR_con}$ $r)$	C^a_{nmur}
Trigger	$E_1, \ldots, E_n, C_1, \ldots, C_k \rightarrow pr{:}E$ after Δt			C_{tr}
Constraint Enabling	pr:enable/disable c where $c \in \{(D, D_x, pr{:}E), (C), (D, C)\}$			C_c
Run-time Requests	*Users' activation request*		(s: (de) activate r for u after Δt))	C_u
	Administrator's run-time request		(pr:assign$_U$/de-assign$_U$ r to u after Δt)	C_{admin}
			(pr:enable/disable r after Δt)	C_{admin}
			(pr:assign$_P$/de-assign$_P$ p to r after Δt)	C_{admin}
			(pr:enable/disable c after Δt)	C_{admin}

denote all the time instants in (I, P). In this paper, we also use function $ESol(I, P)$ to represent the set of endpoints of the intervals in (I, P), i.e., if (I, P) represent the set of intervals $\{(t_{s_1}, t_{e_1}), (t_{s_2}, t_{e_2}), \ldots, (t_{s_n}, t_{e_n})\}$, then

$$ESol(I, P) = \{t_{s_1}, t_{e_1}, t_{s_2}, t_{e_2}, \ldots, t_{s_n}, t_{e_n}\}.$$

D expresses the duration specified for a constraint. In the duration and role activation constraint expressions, D_x and N_x indicate the constrained durations and cardinalities. If the subscript x starts with u, then it is a *per-user-role* constraint; otherwise, it is a *per-role* constraint. For instance, D_{active} indicates the duration for which a specified role remains active, whereas $D_{uactive}$ indicates how long a user may activate the specified role. The following example illustrates the specification of a GTRBAC policy. For more details, we refer the readers to [17].

Example 1. Consider the GTRBAC policy depicted in Table 2 for a hospital enterprise. Constraint 1a specifies that the roles DayDoctor and NightDoctor are to be enabled in *DayTime* and *NightTime*. In constraint 1b, *Adams* is assigned to role DayDoctor on *Mondays*, *Wednesdays*,

TABLE 2
Example GTRBAC Access Policy for a Medical Information System

1	a.	(*DayTime*, enable DayDoctor), (*NightTime*, enable NightDoctor)
	b.	((M, W, F), assign$_U$ *Adams* to DayDoctor), ((T, Th, S, Su), assign$_U$ *Bill* to DayDoctor), ((M, W, F), assign$_U$ *Alice* to NightDoctor), ((T, Th, S, Su), assign$_U$ *Ben* to NightDoctor)
	c.	([10am, 3pm], assign$_U$ *Carol* to DayDoctor)
2	a.	(assign$_U$ *Ami* to NurseInTraining), (assign$_U$ *Elizabeth* to DayNurse)
	b.	$c1$ = (6 *hours*, 2 *hours*, enable NurseInTraining)
3	a.	(enable DayNurse $\rightarrow$ enable $c1$)
	b.	(activate DayNurse for *Elizabeth* $\rightarrow$ enable NurseInTraining after 10 *min*)
	c.	(enable NightDoctor $\rightarrow$ enable NightNurse after 10 *min*) (disable NightDoctor $\rightarrow$ disable NightNurse after 10 *min*)
	d.	(enable DayDoctor $\rightarrow$ enable DayNurse after 10 *min*), (disable DayDoctor $\rightarrow$ disable DayNurse after 10 *min*)
4	a.	(10, active$_{R_n}$ DayNurse)
	b.	(5, active$_{R_n}$ NightNurse)
	c.	(2 *hours*, active$_{R_total}$ NurseInTraining)

TABLE 3
Various Status Predicates

Predicate	Meaning
enabled(r, t)	*Role r is enabled at time t*
u_assigned(u, r, t)	*User u is assigned to role r at time t*
p_assigned(p, r, t)	*Permission p is assigned to role r at time t*
can_activate (u, r, t)	*User u can activate role r at time t*
can_acquire (u, p, t)	*User u can acquire permission p at time t*
can_be_acquired(p, r, t)	*Permission p can be acquired through role r at time t*
active(u, r, s, t)	*Role r is active in user u's session s at time t*
acquires(u, p, s, t)	*User u' acquires permission p in session s at time t*
Axioms: *For all* $r \in$ Roles, $u \in$ Users, $p \in$ Permissions, $s \in$ Sessions, *and time instant* $t \geq 0$, *the following implications hold*:	
1	*assigned*(p, r, t)→ *can_be_acquired*(p, r, t)
2	*assigned*(u, r, t) → *can_activate* (u, r, t)
3	*can_activate* (u, r, t) ∧ *can_be_acquired*(p, r, t) → *can_acquire* (u, p,t)
4	*active*(u, r, s, t) ∧ *can_be_acquired*(p, r, t) → *acquires*(u, p, s, t)

and *Fridays*, whereas *Bill* is assigned to it on *Tuesdays*, *Thursdays*, *Saturdays*, and *Sundays*. The assignment in entry 1c indicates that *Carol* can assume the DayDoctor role every day between 10am and 3pm. In assignment 2a, users *Ami* and *Elizabeth* are assigned roles NurseInTraining and DayNurse, respectively, without any periodicity or duration constraints. In other words, their assignments are always valid. Constraint 2b specifies a duration constraint of two *hours* on the enabling time of the NurseInTraining role. However, this constraint is only valid for six *hours* after constraint $c1$ is enabled. Consequently, once the NurseInTraining role is enabled, *Ami* will be able to activate the NurseInTraining role for at the most two hours.

Trigger 3a indicates that constraint $c1$ is enabled once the DayNurse is enabled. As a result, the NurseInTraining role can be enabled within the next six *hours*. Trigger 3b indicates that 10 *min* after *Elizabeth* activates the DayNurse role, the NurseInTraining role is enabled for a period of two *hours*. As a result, a nurse in-training can then have access to the system only if *Elizabeth* has an active session in the system. In other words, once the roles are activated, *Elizabeth* acts as a training supervisor for a nurse in training. It is possible that *Elizabeth* activates the DayNurse role several times within six hours after the DayNurse role is enabled. The activation constraint 4c limits the total activation time associated with the NurseInTraining role to two *hours*. Set 4 shows additional activation constraints. For example, constraint 4a indicates that there can be, at the most, 10 users activating the DayDoctor role at a given time, whereas constraint 4b indicates that there can be, at the most, five users activating the NightNurse role at a time.

2.2 Temporal Role Hierarchies

In this section, we briefly overview the temporal hierarchies in the GTRBAC model [17]. Table 3 shows the predicate notations used in defining the semantics of these hierarchies. Predicates $enabled(r,t)$, $assigned(u,r,t)$, and $assigned(p,r,t)$ refer to the status of roles, user-role, and role-permission assignments at time t. Predicate $can\_activate(u,r,t)$ indicates that user u can activate role r at time t, implying that user u is implicitly or explicitly assigned to role r. $active(u,r,s,t)$ indicates that role r is active in user u's session s at time t, whereas $acquires(u,p,s,t)$ implies that u acquires permission p at time t in session s. The axioms in Table 3 capture the key relationships among these predicates and identify the permission-acquisitions and role-activations allowed in the GTRBAC model. Axiom 1 states that if a permission is assigned to a role, then it *can be acquired* through that role. Axiom 2 states that all users assigned to a role *can activate* that role. Axiom 3 states that if a user u *can activate* a role r, then all the permissions that *can be acquired* through r *can be acquired* by u. Similarly, axiom 4 states that if there is a user session in which a user u has activated a role r, then u *acquires* all the permissions that *can be acquired* through r. We note that axioms 1 and 2 indicate that permission-acquisition and role-activation semantics are governed by explicit user-role and role-permission assignments.

Semantically, a role hierarchy extends the scope of the permission-acquisition and role-activation semantics beyond the explicit assignments through the hierarchical relations defined among roles. Within the GTRBAC framework, the following three hierarchy types are identified: *permission-inheritance-only* hierarchy (*I*-hierarchy), *role-activation-only* hierarchy (*A*-hierarchy), and the combined

TABLE 4
Role Hierarchies in GTRBAC

Category	Short form	Notation	The condition c holds
Unrestricted hierarchies	*I-hierarchy*	$(x \geq_t y)$	$\forall p, (x \geq_t y) \wedge can\_be\_acquired(p, y, t) \rightarrow can\_be\_acquired(p, x, t)$
	A-hierarchy	$(x \succcurlyeq_t y)$	$\forall u, (x \succcurlyeq_t y) \wedge can\_activate(u, x, t) \rightarrow can\_activate(u, y, t)$
	IA-hierarchy	$(x \succsim_t y)$	$(x \succsim_t y) \leftrightarrow (x \geq_t y) \wedge (x \succcurlyeq_t y)$
Enabling time restricted hierarchies	*Weakly Restricted*		
	I_w-*hierarchy*	$(x \geq_{w,t} y)$	$\forall p, (x \geq_{w,t} y) \wedge enabled(x, t) \wedge can\_be\_acquired(p, y, t) \rightarrow can\_be\_acquired(p, x, t)$
	A_w-*hierarchy*	$(x \succcurlyeq_{w,t} y)$	$\forall u, (x \succcurlyeq_{w,t} y) \wedge enabled(y, t) \wedge can\_activate(u, x, t) \rightarrow can\_activate(u, y, t)$
	IA_w-*hierarchy*	$(x \succsim_{w,t} y)$	$(x \succsim_{w,t} y) \leftrightarrow (x \geq_{w,t} y) \wedge (x \succcurlyeq_{w,t} y)$
	Strongly Restricted		
	I_s-*hierarchy*	$(x \geq_{s,t} y)$	$\forall p, (x \geq_{s,t} y) \wedge enabled(x, t) \wedge enabled(y, t) \wedge can\_be\_acquired(p, y, t) \rightarrow can\_be\_acquired(p, x, t)$
	A_s-*hierarchy*	$(x \succcurlyeq_{s,t} y)$	$\forall u, (x \succcurlyeq_{s,t} y) \wedge enabled(x, t) \wedge enabled(y, t) \wedge can\_activate(u, x, t) \rightarrow can\_activate(u, y, t)$
	IA_s-*hierarchy*	$(x \succsim_{s,t} y)$	$(x \succsim_{s,t} y) \leftrightarrow (x \geq_{s,t} y) \wedge (x \succcurlyeq_{s,t} y)$
Consistency Property: *Let* $<f_1>, <f_2> \in \{\geq_t, \succcurlyeq_t, \succsim_t\}$. *Let x and y be distinct roles such that* $(x<f_1>y)$; *then the condition* $\neg(y<f_2>x)$ *must hold.*			

inheritance-activation hierarchy (*IA*-hierarchy) [16], [17]. Each of these hierarchies may be of *restricted* or *unrestricted* type. A *restricted* hierarchy may further be categorized as *weakly* or *strongly* restricted. In Table 4, the semantics of each hierarchy type is defined by its corresponding condition *c*. The condition c for the *unrestricted I*-hierarchy, $(x \geq_t y)$, implies that if $(x \geq_t y)$ holds, then the permissions that can be acquired through role x include all the permissions that *can be acquired through* role y. In other words, permissions of the junior roles are inherited by the senior role. Similarly, the condition *c* corresponding to the *unrestricted A*-hierarchy implies that if user u *can activate* role x, and $(x \succeq_t y)$ is defined, then user u *can* also *activate* role y even if he is not explicitly assigned to y. Moreover, it also implies that user u cannot acquire y's permissions by merely activating x. In other words, permission-inheritance is not allowed in an *unrestricted A*-hierarchy. The *IA*-hierarchy is the most general form of hierarchy and includes both permission-inheritance and role-activation semantics.

Restricted hierarchies capture the semantic relation between the enabling times of the hierarchically related roles. The *weakly restricted* hierarchies allow inheritance or activation semantics in the nonoverlapping enabling intervals of the hierarchically related roles, whereas the *strongly restricted* hierarchies allow inheritance and activation semantics only in the overlapping enabling intervals. According to the condition for the *weakly restricted I*-hierarchy, if $(x \geq_{w,t} y)$ is defined, then only role x needs to be enabled at time t for the inheritance semantics to apply. Role y may or may not be enabled at that time. However, for the *strongly restricted I*-hierarchy, if $(x \geq_{s,t} y)$ is defined, both x and y need to be enabled at time t for the inheritance semantics to apply. The *restricted A* and *IA*-hierarchies are defined similarly.

3 Expressiveness of GTRBAC and Design Considerations

As shown in Section 2, the GTRBAC model allows specification of a large set of time-based constraints. A pertinent question is whether such an exhaustive set of temporal constraints is desirable at all or if there is a minimal set of constraint types that has the same expressive power as the set containing all the constraint types of the GTRBAC model [17]. In this section, we show formally that the set of GTRBAC constraint types is not minimal. By introducing the notion of *activity-equivalence* or *a-equivalence*, we show that there exists a minimal set of constraint types that have an expressive power equivalent to the set of all the GTRBAC constraint types. However, we show through an analysis that even though such a minimal set exists, the nonminimal set of GTRBAC constraints provides better alternatives for representing access constraints. In particular, such alternatives allow user convenience and lower complexity of representation. Furthermore, the large set of constraints in the GTRBAC model provides flexibility and allows appropriate choice of a semantically clear specification, enhancing the usability of the model.

Algorithm TransformPR

Input : Cf_{in}; **Output :** Cf_{out}

1. $Cf_{out} = \{T', \texttt{Users}, \texttt{Roles}', \texttt{Permissions}, RH'\} = Cf_{in} = \{T, \texttt{Users}, \texttt{Roles}, \texttt{Permissions}, RH\}$;
2. **FOR** *each* $c = (X, pr{:}\texttt{assign}_{\texttt{P}}/\texttt{deassign}_{\texttt{P}}\ p\ \texttt{to}\ r) \in T$, *where* $X = \{(I, P), ([(I, P)], D_x], D)\}$ **DO**
3. Create a unique role r_i;
4. Replace all occurrences of $\{X, pr{:}\texttt{assign}_{\texttt{P}}/\texttt{deassign}_{\texttt{P}}\ p\ \texttt{to}\ r\}$ by $\{X, pr{:}\texttt{enable}_{\texttt{P}}/\texttt{disable}_{\texttt{P}}\ r_i\}$ in T'
5. Add default assignment "$\texttt{assign}_{\texttt{P}}/\texttt{deassign}_{\texttt{P}}\ p\ \texttt{to}\ r_i$" to T'
6. **FOR** *each trigger* $tr \in T'$, where $tr =$ "$E_1, \ldots, E_n, C_1, \ldots, C_k \rightarrow pr{:}E_{n+1}\ \texttt{after}\ \Delta t$" **DO**
7. Replace tr by $tr' =$ "$E'_1, \ldots, E'_n, C'_1, \ldots, C'_k \rightarrow pr{:}E'_{n+1}\ \texttt{after}\ \Delta t$", *such that*, $(i = 1$ to $n+1$, $j = 1$ to $k)$
8. **IF** ($E_i ==$ "$\texttt{assign}_{\texttt{P}}/\texttt{deassign}_{\texttt{P}}\ p\ \texttt{to}\ r$") **THEN** $E'_i =$ "$\texttt{enable}_{\texttt{P}}/\texttt{disable}_{\texttt{P}}\ r_i$";
9. **ELSE** $E'_i = E_i$;
10. **IF** ($C_j ==$ "$\texttt{assigned/deassigned}\ p\ \texttt{to}\ r$") **THEN** $C'_j =$ "$\texttt{enabled/disabled}\ r_i$";
12. **ELSE** $C'_j = C_j$;
13. **ENDFOR**
14. $\texttt{Roles}' = \texttt{Roles}' \cup \{r_i\}$;
15. **FOR** *each role* $r_j \in \texttt{Roles}$ *such that* $\{r \geq r_j\}$ **DO**
16. $RH' = RH' \cup \{r_i \geq_s r_j\}$; $RH' = RH' - \{r \geq_s r_j\}$
17. **ENDFOR**
18. $RH' = RH' \cup \{r \geq_s r_i\}$; // *Strongly restricted* ***I-hierarchy***
19. **ENDFOR**

Fig. 1. Algorithm TransformPR.

3.1 Minimality of GTRBAC

Given a GTRBAC system, we call the set containing all its constraints *Temporal Constraint and Activation Base* (*TCAB*). A *TCAB* can be represented as a set

$$T = (C_{URp}, C_{Rp}, C_{PRp}, C_{URd}, C_{Rd}, C_{PRd}, C^a_{dr}, C^a_{dur}, C^a_{mr}, C^a_{mur}, C^a_{nr}, C^a_{nur}, C^a_{nmr}, C^a_{nmur}, C_{tr}, C_c),$$

where each component in this set is a constraint type as depicted in the last column of Table 1. Here, we use a constraint type name to refer to the set containing the constraints of that type as well; for example, C_{URp} also refers to the set containing the periodicity constraints on user-role assignments. In the following discussion, we use a shorter version, such as $T = (C_{Rp}, C_{URp})$, when only C_{Rp} and C_{URp} are the nonempty sets of constraints. The behavior of a GTRBAC system depends on T, the set of users Users, the set of roles Roles, the set of permissions Permissions, and the role hierarchy RH. Therefore, we use the tuple $(T, \texttt{Users}, \texttt{Roles}, \texttt{Permissions}, RH)$ to indicate a GTRBAC configuration. We use the notation $(u \overset{Cf}{\underset{t}{\Rightarrow}} p)$ to read "u *acquires permission* p *at time* t *under configuration* Cf." Next, we define the notion of *a-equivalence* between two GTRBAC configurations.

Definition 1 (Activity-equivalence or a-equivalence). *Given a GTRBAC system with two configurations* $Cf_1 = (T_1, \texttt{Users}, \texttt{Roles}_1, \texttt{Permissions}, RH_1)$ *and*

$$Cf_2 = (T_2, \texttt{Users}, \texttt{Roles}_2, \texttt{Permissions}, RH_2),$$

the configurations Cf_1 *and* Cf_2 *are said to be a-equivalent (written as* $Cf_1 \approx Cf_2$*) if, for all pairs* (u, p) *such that* $u \in \texttt{Users}$, $p \in \texttt{Permissions}$, *the following condition holds:*

$$\left(u \overset{Cf_1}{\underset{t}{\Rightarrow}} p\right) \leftrightarrow \left(u \overset{Cf_2}{\underset{t}{\Rightarrow}} p\right).$$

Furthermore, if $Cf_1 \approx Cf_x$ *and* $Cf_x \approx Cf_2$, *then* $Cf_1 \approx Cf_2$ *(transitivity property).*

The *a-equivalence* between two configurations of a GTRBAC system indicates that a user can perform the same set of accesses under the two configurations. Hence, by replacing configuration Cf_1 by Cf_2, we do not change the accesses that are allowed for each individual user. Note that *a-equivalence* does not necessarily imply policy equivalence as we consider the same set of users and permissions. Policy equivalence would mean that, at all times, the two configurations follow the same rules. Our goal here is to show different configurations of roles and constraints, allowing the same set of users to acquire the same set of permissions, and then analyze the complexities of these configurations.

Next, we illustrate that the set of GTRBAC constraint types is not minimal, i.e., some constraint types can be removed without reducing the expressive power of the GTRBAC system. Using the *a-equivalence* relation over a set of GTRBAC configurations, we will show that there is a minimal representation that uses only periodicity and duration constraints on roles and the *per-role* activation constraints. We still need the *default* assignments, represented by C_d, that assign users or permissions to roles without specifying any temporal restriction.

We now present algorithms TransformPR and TransformUR shown in Figs. 1 and 2 that can be used to generate *a-equivalent* configurations for a given input configuration. Algorithm TransformPR produces an *a-equivalent* configuration of a given GTRBAC configuration, with all the temporal constraints on role-permission assignments replaced by those on role enabling. Similarly, algorithm TransformUR produces a new configuration *a-equivalent* to the input configuration Cf_{in} with all the user-role assignments and *per-user-role* activation constraints replaced by those on role enabling and per-role activation, respectively.

The following two lemmas formally prove that the transformation carried out by each algorithm correctly outputs *a-equivalent* configurations. The proofs of these lemmas and the other formal statements presented later

Algorithm TransformUR

Input : Cf_{in}; **Output :** Cf_{out}

1. $Cf_{out} = Cf_{in}$ (i.e., {T', Users, Roles', Permissions, RH'} = {T, Users, Roles, Permissions, RH}); $S = \emptyset$;
2. **FOR** *each* $c = (X, pr{:}\texttt{assign}_U/\texttt{deassign}_U\ u\ \texttt{to}\ r) \in T$, *where* $X = \{(I, P), ([(I, P)], D_x], D)\}$ **DO**
3. Create a unique role r_i; $S = S \cup (u, r, r_i)$ // function $getSu_i(S, u, r)$ used in line 24 returns r_i
4. Replace all occurrences of $\{X, pr{:}\texttt{assign}_U/\texttt{deassign}_U\ u\ \texttt{to}\ r\}$ by $\{X, pr{:}\texttt{enable/disable}\ r_i\}$ in T'
5. Add default assignment "assign/deassign u to r_i to T' "
6. **FOR** *each trigger* $tr \in T'$, *where* $tr =$ "$E_1, \ldots, E_n, C_1, \ldots, C_k \rightarrow pr{:}E_{n+1}$ after Δt" **DO**
7. Replace tr by tr' where tr' == "$E'_1, \ldots, E'_n, C'_1, \ldots, C'_k \rightarrow pr{:}E'_{n+1}$ after Δt" *such that*
8. **IF** (E_i==" $\texttt{assign}_U/\texttt{deassign}_U\ u\ \texttt{to}\ r$") **THEN** E'_i := "$\texttt{enable}_U/\texttt{disable}_U\ r_i$" ;
9. **ELSE** $E'_i := E_i$;
10. **IF** (C_j==" $\texttt{assigned/deassigned}_U\ u\ \texttt{to}\ r$") **THEN** C'_j := "enabled/disabled r_i" ;
12. **ELSE** $C'_j := C_j$;
13. **ENDFOR**
14. Roles' = Roles' $\cup \{r_i\}$;
15. **FOR** *each role* $r_j \in$ Roles *such that* $\{r_j \geqslant r\}$ **DO**
16. $RH' = RH' \cup \{r_j \geqslant_s r_i\}$; $RH' = RH' - \{r_j \geqslant_s r\}$; // *Strongly restricted A-hierarchy*
17. **ENDFOR**
18. $RH' = RH' \cup \{r_i \geqslant_s r\}$;
19. **ENDFOR**
20. // Handle all the per-role-activation constraints
21. **FOR** *each pair* (u, r) *such that there is an activation constraint* $(X, Y_u, u, \texttt{active}_{UY}\ r) \in T'$
22. *where* $X \in \{(I, P), D\}$, $Y_u \in \{D_{uactive}, D_{umax}, N_{uactive}, D_{umax}\}$ *and*
23. $\texttt{active}_{UY} = \{\texttt{active}_{UR\_total}, \texttt{active}_{UR\_max}, \texttt{active}_{UR\_n}, \texttt{active}_{UR\_con}\}$ **DO**
24. **IF** (r_i:=$getSu_i(S, u, r)$ == NIL) **THEN** Create a unique role r_i, // $getSu_i(S, u, r)$ == NIL means that
25. **FOR** *each* $c = (X, Y_u, u, \texttt{active}_{UY}\ r) \in T'$ **DO** // *there was no u, r assignment in* line 2
26. Replace c in T' by c' *where* $c' = (X, Y_u, \texttt{active}_{UY}\ r_i)$; //*Note that old c will not be in T'*
27. Replace all occurrences of "enable c" by "enable c' "
28 **ENDFOR**
29. **IF** (r_i *was created new in* Line 24) **THEN**
30. Role' = Role' $\cup \{r_i\}$;
31. **FOR** *each role* $r_j \in$ Roles *such that* $\{r_j \gtrsim r\}$**DO**
32. $RH' = RH' \cup \{r_j \gtrsim_s r_i\}$; $RH' = RH' - \{r_j \gtrsim_s r\}$; // *Strongly restricted I****A-hierarchy***
33. **ENDFOR**
34. $RH' = RH' \cup \{r_i \geqslant r\}$;
35. Replace per-role activation constraint by $(0, \texttt{active}_{R\_n}\ r)$ in T'
36. **ENDFOR**

Fig. 2. Algorithm TransformUR.

in the paper are provided in the Appendix (which is available as a free pdf at www.computer.org/portal/pages/transactions/tdsc/content/archives.html).

Lemma 1 (Correctness of TransformPR). *Given an input configuration Cf_{in}, algorithm* TransformPR *produces Cf_{out} such that there are no temporal role-permission assignments in Cf_{out}, and $Cf_{in} \approx Cf_{out}$.*

Lemma 2 (Correctness of TransformUR). *Given an input configuration Cf_{in}, algorithm* TransformUR *produces Cf_{out} such that there are no temporal user-role assignments and per-user-role activation constraints in Cf_{out}, and $Cf_{in} \approx Cf_{out}$.*

These lemmas indicate that, by replacing the temporal constraints on assignments by those on roles and per-user-role constraints by per-role constraints, we get a reduced model that has the same expressiveness. We use the following notion of *minimal constraint set* (MCS) to show the fact that there is an *a-equivalent* configuration with the minimum number of constraint types.

Definition 2 (Minimal Constraint Set). *Let MCS(T) be the set of constraint types in TCAB T, and $CS = \{Cf_1, Cf_2, \ldots Cf_n\}$ be an a-equivalent set of configurations for some n, such that*

$$Cf_i = (T_i, \texttt{Users}, \texttt{Roles}_i, \texttt{Permissions}, RH_i)$$

for $i = 1, 2, \ldots, n$. We say that $MCS(T_i)$ is the minimal constraint set (MCS) of CS if there exists no other configuration $Cf_j = (T_j, \texttt{Users}, \texttt{Roles}_j, \texttt{Permissions}, RH_j)$ such that $i \notin j$ and $MCS(T_j) \subset MCS(T_i)$.

The definition implies that a *minimal constraint* set is the one that has the least number of temporal constraint types. Note that the definition also implies that the role set and hierarchy structures may be different in different configurations. We now proceed to present the minimality result for GTRBAC system, which is expressed by the following theorem.

Theorem 1 (Minimality of GTRBAC). *Let Cf_1 be a GTRBAC configuration such that*

$$\{C_d, C_{Rp}, C_{Rd}, C_r^a, C_{tr}, C_c\} \subset MCS(T_1).$$

There exists a GTRBAC configuration Cf_2 such that:

1. $Cf_1 \approx Cf_2$,
2. $MCS(T_2) = \{C_d, C_{Rp}, C_{Rd}, C_r^a, C_{tr}, C_c\}$, ($C_r^a$ *indicate all per-role constraint types), and*

TABLE 5
Relations on Periodic Expression

Relation	If the following condition(s) is (are) satisfied
1. PE_1 contained *in* PE_2 ($PE_1 \subset PE_2$)	$Sol(I_1, P_1) \subset Sol(I_2, P_2)$
2. PE_1 & PE_2 *are* equivalent ($PE_1 \quad PE_2$)	$Sol(I_1, P_1) = Sol(I_2, P_2)$
3. PE_1 *and* PE_2 overlap ($PE_1 \otimes PE_2$)	• $Sol(I_1, P_1) \cap Sol(I_2, P_2) \quad \varnothing$ & • $\exists\, t,\, t \in (Sol(I_1, P_1) - Sol(I_2, P_2))$, & • $\exists\, t,\, t \in (Sol(I_2, P_2) - Sol(I_1, P_1))$
4. PE_1 *and* PE_2 *are* disjoint ($PE_1 \Diamond PE_2$),	$\forall t, (t \in Sol(I_1, P_1) \wedge t \in Sol(I_2, P_2)) \rightarrow t \in (ESol(I_1, P_1) \cap ESol(I_2, P_2))$ *where* $ESol(I, P)$ *is the set of end points of intervals of* (I, P) (See Section 2.1)

3. $MCS(T_2)$ *is a minimal constraint set over* $\{Cf_1\} \cup \{Cf \mid Cf_1 \approx Cf\}$.

Theorem 1 claims that the original set of GTRBAC constraints is not minimal because a set of default assignments, periodicity, and duration constraints on role enabling (C_{Rp}, C_{Rd}), *per-role* activation constraints (C_r^a), triggers (C_{tr}), and constraint enabling expression (C_c) can be used to represent any access policy that can also be represented by the full set of the GTRBAC constraints. It can be noticed from the transformation algorithms that replacing temporal constraints on assignments by temporal constraints on roles, in general, increases the number of roles and the complexity of a role hierarchy. The reason is that algorithms `TransformPR` and `TransformUR` create a new role while replacing each temporal assignment. This may not seem intuitive and efficient as it appears that there will be as many new roles generated as there are temporal assignments being replaced. A more intuitive and practical approach would be to create the least number of roles with enabling intervals that are nonoverlapping. For example, if there is a Doctor role and each of the n users are assigned to this role for either day time or night time (or both), then, instead of creating n new roles, we can simply create the temporal roles DayDoctor and NightDoctor and assign the users to either of the two roles. To create such temporally nonoverlapping roles, we must first divide n periodic expressions into a temporally nonoverlapping set of periodic expressions such as *Daytime* and *Nighttime*. We next provide formal definitions and algorithms needed to generate this set by creating disjoint periodicity expressions from a set of periodicity expressions.

Note that, our minimal model is the one that has temporal constraints on various role enablings. It may be possible that we can construct another minimal model with temporal constraints on the assignments (user-role and/or role-permission) instead of role enablings. As roles are the central entity of an RBAC model, we focus on the minimal model shown above. Furthermore, the constraints on role-activation cannot have any equivalent representation using user-role or role-permission assignments as they are refered to as runtime constraints. Hence, there would still be some temporal constraints on roles even if we eliminate temporal constraints on role enablings.

3.2 Operations on Periodicity Expressions

In this section, we first introduce the formal notions of *containment*, *equivalence*, *overlapping*, and *disjunction* operations between a pair of periodic expressions. Note that an arbitrary set of intervals can be represented by a periodic expression. This is possible because each such expression, in the worst case, can be formulated as a periodic expression that lists every starting point and the smallest calendar as duration [20].

Definition 3 (Relations on periodic expressions). *Let* $PE_1 = (I_1, P_1)$ *and* $PE_2 = (I_2, P_2)$ *be two periodic expressions, then the relations between them can be as shown in Table 5.*

Fig. 3 shows examples of these relations. Note that the fourth part of the definition implies that if the only endpoints of intervals of two periodic expressions are common, they are considered disjoint.

We can extend these pair wise relations to define relations among a set of periodic expressions. A set of periodic expressions is said to be *disjoint* if the periodic expressions are pair wise disjoint. Similarly, a set of periodic expressions are *equivalent* if all the periodic expressions are equivalent to one another. Ideally, we want to compute a disjoint set of periodic expressions that is minimal so that these can be associated with roles to make

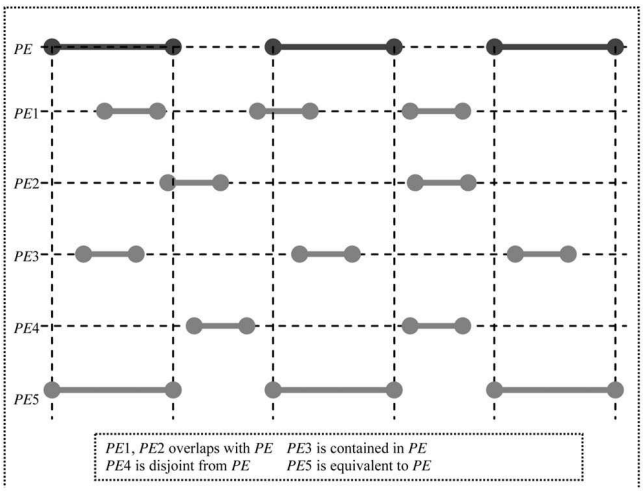

Fig. 3. Temporal relations between a pair of periodic expressions.

Example 2: To simplify notation, we assume the *Daytime* = (9am-9pm) for each expression. For example, if PE = {Sun}, we mean the interval (9am, 9pm) or daytime of a Sunday. Let PE_A = {Sun, Mon, Tue, Wed, Thu, Fri}, PE_B = {Sun, Tue}, PE_C = {Sun, Tue, Thu, Fri}, PE_D = {Sun, Mon, Tue, Wed, Sat}, PE_E = {Thu, Fri}. The following steps illustrate the computation of *MDS* using algorithm `computeMDS`.

1. $MDS_{\{PE_A, PE_B\}}$ = $\{PE'_1, PE'_2\}$ = {{Sun, Tue}, {Mon, Wed, Thu, Fri}} (as $PE_B \subset PE_A$)
2. $MDS_{\{PE_A, PE_B, PE_C\}}$ = MDS of $\{PE'_1, PE'_2, PE_C\}$ = MDS of {{Sun, Tue}, {Mon, Wed, Thu, Fri}, {Sun, Tue, Thu, Fr}}. Here,

   MDS of $(PE'_1, PE_C\}$ = $\{PE'_{x1}$ ={Sun, Tues}, PE'_{y1} = {Thu, Fr}}, (as $PE'_1 \subset PE_C$)

   MDS of $\{PE'_2, PE_C\}$= $\{PE'_{x2}$ ={Thu, Fri}, PE'_{y2} ={Sun, Tues}, PE'_{z2}= {Mon, Wed}}(as $PE'_2 \otimes PE_C$)

   Note: $PE'_{x1} \cup PE'_{x2} \cup PE'_{z2}$ = {Sun, Mon, Tues, Wed, Thu, Fri}.

   Therefore, $PE''_4 = PE_C - (PE'_{x1} \cup PE'_{x2} \cup PE'_{z2}) = \emptyset$

   Hence, $MDS_{\{PE_A, PE_B, PE_C\}}$ =$\{PE''_1, PE''_2, PE''_3\}$= {{Sun, Tues}, {Thu, Fri}, {Mon, Wed}}
3. $MDS_{\{PE_A, PE_B, PE_C, PE_D\}}$ = MDS of $\{PE''_1, PE''_2, PE''_3, PE_D\}$= MDS of {{Sun, Tues}, {Thu, Fri}, {Mon, Wed}, {Sun, Mon, Tues, Wed, Sat}}. Here,
   - MDS of $\{PE''_1, PE_D\}$ = $\{PE'_{x3}$ ={Sun, Tues}, PE'_{y3} = {Mon, Wed, Sat}} (as $PE''_1 \subset PE_D$),
   - MDS of $\{PE''_2, PE_D\}$ =$\{PE'_{x4}$= {Thu, Fri}, PE'_{y4} ={Sun, Mon, Tues, Wed, Sat}} (as $PE''_2 \Diamond PE_D$),
   - MDS of $\{PE''_3, PE_D\}$= $\{PE'_{x5}$={Mon, Wed}, PE'_{y5} ={Sun, Tues, Sat}}(as $PE''_3 \subset PE_D$),
   - Note: $PE'_{x3} \cup PE'_{x4} \cup PE'_{x5}$ = {Sun, Mon, Tues, Wed, Thu, Fri},
   - $PE'''_4 = PE_D - (PE'_{x3} \cup PE'_{x4} \cup PE'_{x5})$ = {Sat};

Therefore, $MDS_{\{PE_A, PE_B, PE_C, PE_D\}}$ =$\{PE'''_1, PE'''_2, PE'''_3, PE'''_4\}$={{Sun, Tues},{Thu,Fri},{Mon,Wed}, {Sat}}

$MDS_{\{PE_A, PE_B, PE_C, PE_D, PE_E\}}$ = MDS of $\{PE''_1, PE''_2, PE''_3, PE'''_4, PE_E\}$

= MDS of {{Sun, Tues}, {Thu, Fri}, {Mon, Wed}, {Sat}, {Thu, Fri}}

Since $PE_E = PE''_2$, $MDS_{\{PE_A, PE_B, PE_C, PE_D, PE_E\}} = MDS_{\{PE_A, PE_B, PE_C, PE_D\}}$

= $\{PE'''_1, PE'''_2, PE'''_3, PE'''_4\}$ = {{Sun, Tues}, {Thu, Fri}, {Mon, Wed}, {Sat}}

Fig. 4. Example 2.

them temporally distinct. The next definition expresses the notion of *minimal disjoint set* (MDS) over a set of periodic expressions.

Definition 4 (Minimal Disjoint Set). *Let* $PE = \{PE_1, PE_2, \ldots, PE_n\}$ *be a set of arbitrary periodic expressions. The* minimal disjoint set (MDS) over PE *is the least set of disjoint periodic expressions,* MDS_{PE}*, defined as:* $MDS_{PE} = \min_m\{PE'_i \mid 1 \le i \le m\}$ *such that the following conditions hold:*

1. *For all* $1 \le i, j \le m$, $i \ne j$, $(PE'_i \diamond PE'_j)$.
2. $Sol(PE'_1) \cup Sol(PE'_2) \cup \ldots \cup Sol(PE'_m) = Sol(PE_1) \cup Sol(PE_2) \cup \ldots \cup Sol(PE_n)$.
3. *For all* $1 \le i \le m$, $1 \le j \le n$, *either* $(PE'_i \subset PE_j)$ *or* $(PE'_i \diamond PE_j)$.

In this definition, the first and second conditions indicate that the MDS contains a disjoint set of periodic expressions containing all the time instants that are exactly contained in all the original set of periodic expressions PE_is. The third condition ensures that each PE'_i can be either contained in or disjoint from each PE_j. Example 2 (Fig. 4) illustrates the notion of MDS.

Associated with an MDS, we define a *minimal subset* (MS) *of a periodic expression over* the MDS as follows and is illustrated in Example 3 (Fig. 5).

Definition 5 (Minimal subset (MS) for a periodic expression over a MDS). *Let* $MDS_{PE} = \min_m\{PE'_i \mid 1 \le i \le m\}$ *be a minimal disjoint set over*

$$PE = \{PE_1, PE_2, \ldots, PE_n\}$$

for some n*; The* minimal subset (MS) for a periodic expression $PE_j \in PE$ over the MDS_{PE} *is the set* $MS_{PE_j}(MDS_{PE}) = \{PE'_{\pi 1}, PE'_{\pi 2}, \ldots, PE'_{\pi k}\} \subseteq MDS_{PE}$, $1 \le k \le m$ *such that,*

- $\min_k\{\pi k \mid 1 \le i \le k, \pi i \in \{1, 2, \ldots, m\}\}$, *and*
- *for each* $t \in Sol(PE_x)$, *there is exactly one* $y \in \{\pi 1, \pi 2, \ldots, \pi k\}$ *such that* $(t \in Sol(PE'_y))$.

It can be noted that the MS of a periodic expression PE_x of PE is the minimal subset of MDS_{PE} that collectively

Example 3. Consider the computation of the MDS in example 2.

$MDS_{\{PE_A,PE_B,PE_C,PE_D,PE_E\}} = \{PE'''_1, PE'''_2, PE'''_3, PE'''_4\}$

$= \{\{\text{Sun, Tues}\}, \{\text{Thu, Fri}\}, \{\text{Mon, Wed}\}, \{\text{Sat}\}\}$

Hence, we see that,

1. $MS_{PE_A}(MDS_{\{PE_A,PE_B,PE_C,PE_D,PE_E\}}) = \{PE'''_1, PE'''_2, PE'''_3\}$.
2. $MS_{PE_B}(MDS_{\{PE_A,PE_B,PE_C,PE_D,PE_E\}}) = \{PE'''_1\}$
3. $MS_{PE_C}(MDS_{\{PE_A,PE_B,PE_C,PE_D,PE_E\}}) = \{PE'''_1, PE'''_2\}$.
4. $MS_{PE_D}(MDS_{\{PE_A,PE_B,PE_C,PE_D,PE_E\}}) = \{PE''''_1, PE''''_3, PE''''_4\}$.
5. $MS_{PE_E}(MDS_{\{PE_A,PE_B,PE_C,PE_D,PE_E\}}) = \{PE''''_2\}$.

Fig. 5. Example 3.

contains all the time instants of PE_x. Next, we illustrate some formal properties related to the computation of MDS and MS. We use ${}^iMDS_{PE}$ to represent MDS of the first i periodic expressions of PE, i.e., $\{PE_1, PE_2, \ldots, PE_i\}$.

Lemma 3 (MDS for two expressions). *Let $PE = \{PE_1, PE_2\}$ be a set with a pair of nonequivalent and nondisjoint periodic expressions; then the following conditions hold:*

1. *If $(PE_i \subset PE_j)$, then, for $(i,j) \in \{(1,2),(2,1)\}$, there exist periodic expressions PE_x, PE_y such that $MDS_{PE} = \{PE_x, PE_y\}$. Furthermore, $PE_x = PE_i$ and $Sol(PE_y) = Sol(PE_j) - Sol(PE_i)$.*
2. *If $(PE_i \otimes PE_j)$, then, for $(i,j) \in \{(1,2),(2,1)\}$, there exist periodic expressions PE_x, PE_y, PE_z such that $MDS_{PE} = \{PE_x, PE_y, PE_z\}$. Furthermore, $PE_x = PE_i \cap PE_j$, $PE_y = PE_j - PE_x$ and $PE_z = PE_i - PE_x$.*

As each periodicity expression generates a set of time instants, the result follows immediately.

Fig. 6 depicts the algorithms for generating the set MDS of periodic expressions. Algorithm `PairMDS` computes MDS for a pair of periodic expressions. We note that, when two expressions are equivalent, the MDS contains a single periodic expression. Similarly, when the expressions are disjoint, the MDS contains both the periodic expressions. Algorithm `computeMDS` repeatedly calls `PairMDS` and recursively builds the MDS by first finding the MDSs of smaller sizes. The following results show the correctness of the algorithm:

Lemma 4 (MDS for n periodic expressions). *Given a nonequivalent and nondisjoint set of periodic expressions $PE = \{PE_1, PE_2, \ldots, PE_n\}$, there exist periodic expressions $PE'_1, PE'_2, \ldots, PE'_m$ such that*

$$MDS_{PE} = \{PE'_1, PE'_2, \ldots, PE'_m\}.$$

Theorem 2 (MDS using `computeMDS`). *Given an arbitrary set of periodic expressions $PE = \{PE_1, PE_2, \ldots, PE_n\}$, there exists a set $\{PE'_1, PE'_2, \ldots, PE'_m\}$ such that*

1. *$MDS_{PE} = \{PE'_1, PE'_2, \ldots, PE'_m\}$ and*
2. *for PE as input, algorithm `computeMDS` produces MDS_{PE}.*

Corollary 1 (Bounds for size of MDS). *Given a set of periodic expressions $PE = \{PE_1, PE_2, \ldots, PE_n\}$, the algorithm `computeMDS` produces $MDS_{PE} = \{PE'_1, PE'_2, \ldots, PE'_m\}$ such that if $s_n = |MDS_{PE}|$, then $1 \le s_n \le (2^n - 1)$.*

Corollary 2 (Bounds for size of MS). *Given a set of periodic expressions $PE = \{PE_1, PE_2, \ldots, PE_n\}$ and $MDS_{PE} = \{PE'_1, PE'_2, \ldots, PE'_m\}$ produced by algorithm `computeMDS`, if $p_n = |MS_{PE_1}| + |MS_{PE_2}| + \ldots + |MS_{PE_n}|$, then $n \le p_n \le n2^{n-1}$.*

Examples 2 and 3 presented earlier illustrate the computation of MDS by algorithms `computeMDS` and `pairMDS`. We next present an algorithm, shown in Fig. 7, for generatring an *a-equivalent* configuration for a given GTRBAC system by removing the temporal constraints on user-role assignments and computing the MS and MDS of the periodic expressions. Theorem 3 establishes its correctness.

Theorem 3 (Correctness of `TransformMDS`). *Given an input configuration Cf_{in} with only periodicity constraints on user-role assignments, algorithm `TransformMDS` produces a configuration Cf_{out} such that the following holds:*

1. *$Cf_{in} \approx Cf_{out}$ and*
2. *Cf_{out} has no temporal constraint on user-role assignments.*

Note, in algorithm `TransformMDS`, we consider only the periodicity constraints on *user-role* assignment. If we allow the presence of *per-role* constraints also, the algorithm can be easily extended to handle it by introducing *per-role* constraints on the newly created roles.

3.3 System Complexity and Design Considerations

The complexity of a GTRBAC system may have different dimensions. Foremost among them is the number of roles. Typically, an unmanageable number of roles in a system are undesirable. Another concern is the number of temporal constraints. In this case, we deal with the complexity incurred by a hierarchy. In addition, we have the *default*

Algorithm `PairMDS`

Input: PE_1, PE_2

Output: MDS of PE_1, PE_2

```
1  IF (PE1 = PE2) THEN RETURN{PE1};
2  IF (PE1 ◊ PE2) THEN RETURN {PE1, PE2};
3  IF (PE1 ⊂ PE2) THEN  // as per Lemma 4.2.1(a)
4        PEx = PE1;
5        PEy = PE2 - PEx;
6        RETURN {PEx, PEy};
7  IF (PE2 ⊆ PE1) THEN  // as per Lemma 4.2.1(a)
8        PEx = PE2;
9        PEy = PE1 - PEx;
10       RETURN {PEy, PEx};
11 IF (PE1 ⊗ PE2) THEN // as per Lemma 4.2.1(b)
12       PEx = PE1 ⋂ PE2;
13       PEy = PE2 - PEx;
14       PEz = PE1 - PEx;
15       RETURN {PEx, PEy, PEz}
16 END
```

Algorithm `ComputeMDS`

Input: PE_1, PE_2, …, PE_n

Output: MDS of PE_1, PE_2, …, PE_n

```
1  // Assume that PE = {PE1, PE2, …, PEn}
2  S = ∅; MDS = ∅;
3  IF | PE | = 1 THEN RETURN PE;
4  IF | PE | = 2 THEN RETURN PairMDS(PE1, PE2);
5  IF | PE | > 2 THEN
6        MDS = ComputeMDS(PE1, PE2, …, PEn-1);
7        Let MDS computed be (PE'1, PE'2, …, PE'm1);
8        FOR i = 1 to m1 DO
9              PairMDS = PairMDS(PE'i, PEn);
10             IF |PairMDS| = 1 THEN
11                return MDS;
12             IF |PairMDS| = 2 THEN
13                Let PairMDS computed be (PE'x, PE'y);
14                S = S ∪ {PE'x};
15             ELSEIF |PairMDS| = 3 THEN
16                Let PairMDS be (PE'x, PE'y, PE'z);
17                S = S ∪{PE'x, PE'z};
18       ENDFOR
19       Let S computed be (PE''1, PE''2, …, PE''m2);
20       PE''m2+1= PEn - (PE''1 ∪ PE'' ∪ …∪ PE''m2 );
21       IF (PE''m2+1=∅) THEN
22             MDS = (PE''1, PE''2, …, PE''m2,PE''m2+1);
23       ELSE
24             MDS = (PE''1, PE''2, …, PE''m2);
25       RETURN MDS
26 END
```

Fig. 6. Algorithms `PairMDS` and `ComputerMDS`.

Algorithm `TransformMDS`

Input: Cf_{in}

Output : Cf_{out}

```
1. Cfout = {T', Users, Roles', Permissions, RH'}
       = Cfin = {T, Users, Roles, Permissions, RH};
2. FOR each r ∈ Roles DO
3.    Let PE = {PE1, PE2…, PEn} and U = {u1, u2…, un}
          be such that (PEi, assign_U r to ui) ∈T';
4.    Compute MDS of PE; Let the computed
          MDS = {PE'1, PE'2…, PE'n};
5.    FOR i = 1 to n DO
6.       Compute MS_PEi for PEi
7.    ENDFOR
8.    FOR each PE'i ∈ MDS DO
9.       Create a unique role ri;
10.      FOR all uk ∈ U such that PE'i ∈ MS_PEk DO
11.         Add default assignment (assign_U ri to uk) in T'.
12.         Add constraint (PE'i, enable ri ) in T'.
13.         Remove constraint (PEi, assign_U r to ui) from T';
14.            Roles' = Roles' ∪ {ri};
15.         RH' = RH' ∪ {ri ≥s r}; // Strongly restricted A-hierarchy
16.      ENDFOR
17.   ENDFOR
18. ENDFOR
```

Fig. 7. Algorithm `TransformMDS`.

assignments, where only membership needs to be checked to determine whether a particular user is assigned to a role or not. Thus, temporal assignments introduce additional complexity compared to an RBAC system without temporal constraints because such assignments involve, besides checking for membership, ensuring the temporal validity of a membership. To simplify our discussion on the complexity issues, we first develop a family hierarchy of GTRBAC models that have equivalent expressive power, based on the results presented in the previous section, and then investigate the potential benefits of the models at a higher level of this hierarchy over those at the lower level.

For the analysis of the complexity of policy specification, we use the notation for the complexity parameters shown in Table 6. The *minimality* result in the previous section indicates that the minimal GTRBAC model is the one that includes the following temporal constraints: *per-role* activation constraint, *periodicity* and *duration* constraints for role-enabling/disabling, constraint enabling, and triggers. Fig. 8

TABLE 6
Complexity Parameters and Notation Used

Complexity parameter		**Notation Description**
Role	R	*n.R indicates n roles (**Note**: we write 1.R simply as R)*
Default (simple) assignment	S	*n.S indicates n default assignments*
Enabling time constraints on role	T_r	*n.T_r indicates n periodicity/duration constraints on (n) roles*
Temporal constraints on assignments	T_{ur} , T_{rp}	*n.T_{ur} (n.T_{rp}) indicates n periodicity/duration constraints on (n) user- role (role-permissions) assignment*
Activation time constraints on roles	A_{ur}, A_r	*n.A_{ur} (n.A_r) indicates n per-user-role (per-role) activation time constraint*
Hierarchy	H	*n.H indicates n hierarchical relations*

Level	Model	Constraint Set
2	$GTRBAC_2$	$T = T_{1,A} \cup T_{1,U} \cup T_{1,P}$
1	$GTRBAC_{1,P}$	$T_{1,P} = T_0 \cup \{C_{PRp}, C_{PRd}\}$
	$GTRBAC_{1,U}$	$T_{1,U} = T_0 \cup \{C_{Urp}, C_{Urd}\}$
	$GTRBAC_{1,A}$	$T_{1,A} = T_0 \cup \{C^a_{dur}, C^a_{mur}, C^a_{nur}, C^a_{nmur}\}$
0	$GTRBAC_0$ *Minimal*	$T_0 = \{C_d, C_{Rp}, C_{Rd}, C^a_r, C_{tr}, C_c\}$

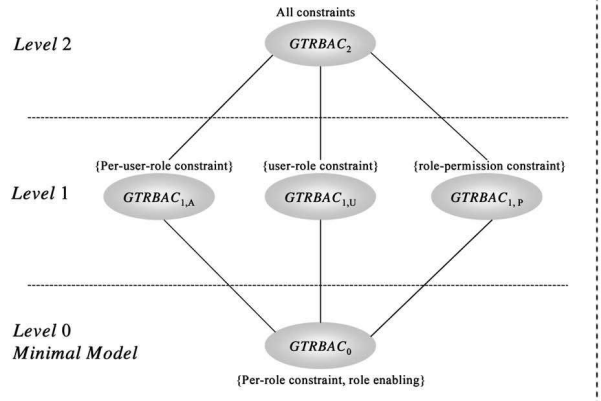

Fig. 8. A family of GTRBAC models.

shows the *minimal model* as $GTRBAC_0$ at level 0 of the family hierarchy. At level 1, we have three different models, each of which introduces a new type of constraint to the constraint set of $GTRBAC_0$. $GTRBAC_{1,A}$ represents the model having all the temporal constraints of $GTRBAC_0$ plus the *per-user-role* activation constraints. Similarly, $GTRBAC_{1,U}$ represents the model having all the temporal constraints of $GTRBAC_0$ plus the *user-role* assignment constraints, whereas $GTRBAC_{1,P}$ represents the model having all the temporal constraints of $GTRBAC_0$ plus the *role-permission* assignment constraints. At level 2, we have the $GTRBAC_2$ model that contains all the temporal constraints. Note that we can have other models between Level 1 and Level 2 representing combinations of the different pairs of models from Level 1. For our analysis, we adopt this simpler hierarchy. The results in the previous section indicate that all the models of the different levels are *a-equivalent*.

Next, we illustrate through analysis that it is advantageous to use a model at a higher level in terms of *user-convenience*, *clarity of semantics*, and *complexity of representation*. Our analysis is focused on the advantages and disadvantages of using a Level 1 model as compared to using the *minimal model* present at Level 0.

3.3.1 Constraints on Role Enabling and Assignments

As shown in Section 3.1, all temporal constraints on *user-role* and *role-permission* assignments can be replaced by the temporal constraints on roles using the algorithms in Figs. 1 and 2. However, such a transformation may result in a large number of roles and/or produce complex access control structures. In this section, we evaluate various design alternatives for choosing constraints on role enablings and assignments. This evaluation is based on comparing the complexity of representation using a Level 1 model against those of various representations using the *minimal model* for expressing the same set of access requirements.

It can be noted, in `TransformUR` of Fig. 2, the transformation involving the replacement of temporal constraints on the *user-role* assignments by the temporal constraints on the roles is similar to the transformation involving the replacement of the temporal constraints on the *role-permission* assignments by the temporal constraints on the roles in `TransformUR` (see Fig. 1), except for the difference as a result of adding the new hierarchy relations. That is, in the first case, the newly created roles are made the seniors of the original role (refer to lines 18 and 35 in Fig. 2) , whereas, in the second case, the original role is made the senior of the new roles (refer to line 18 in Fig. 1). Because of this similarity, we can primarily focus on the *user-role* assignments, as similar results can be obtained for the *role-permission* assignments. Also, algorithm `TransformUR` transforms both the *periodicity* and *duration* constraints in a similar way, i.e., each such constraint is replaced by a new role. Hence, the complexity analysis used for *periodicity* constraints is applicable for the duration constraints as well. Hence, we focus on the *periodicity* constraints only and point out important considerations related to the *duration* constraints whenever they apply. A temporal constraint on *user-role* assignment states that the user can activate a role in the specified periods or for a specified duration, provided the role is enabled. Instead of using a temporal constraint on *user-role* assignment (assuming the user is still assigned to the role using default assignment), we enforce the desired access control by using the temporal constraints on role enabling. Next, we will present the complexity issues related to the representations of a set of access requirements using $GTRBAC_0$ and $GTRBAC_{1,U}$ models. For our purpose, we use the following example:

Example 4. Assume a DayDoctor role in a hospital. Five doctors A, B, C, D, and E are assigned to this role in the periods given by the periodic expressions PE_A, PE_B, PE_C, PE_D, and PE_E of Example 2. We assume $GTRBAC_{1,U}$ representation of these constraints and, hence, there are no activation constraints. We also focus on the two different representations using the $GTRBAC_0$ model, denoted as $GTRBAC_0^1$ and $GTRBAC_0^2$.

$GTRBAC_{1,U}$ **representation**. For each doctor, a periodicity constraint on his assignment to the DayDoctor role is specified using periodic expressions shown in Fig. 9a. For example, for doctor A, the periodic expression PE_A is used, i.e., there is a constraint (PE_A,

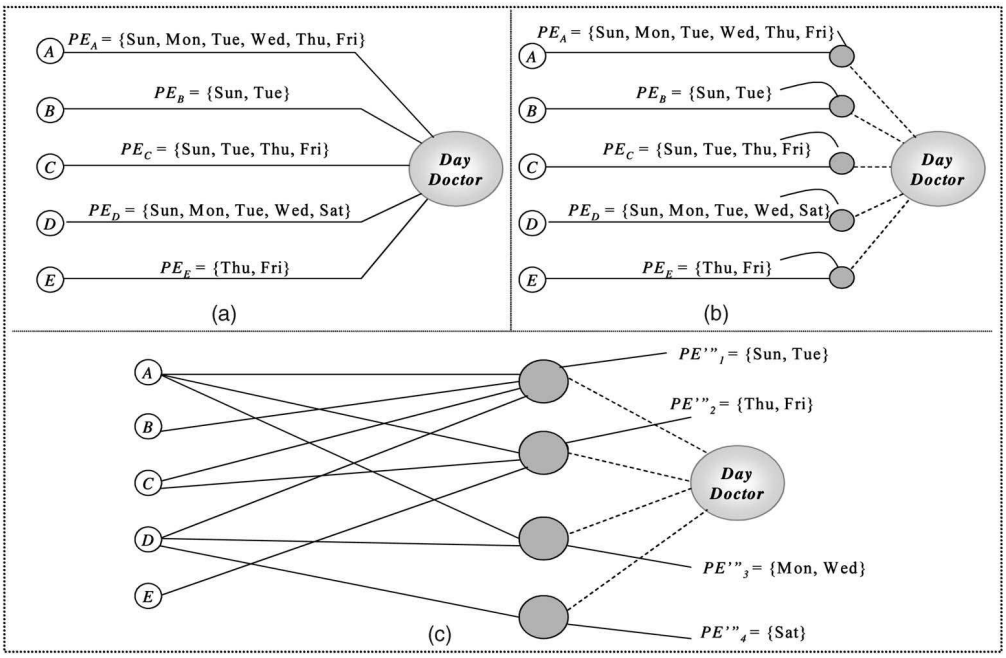

Fig. 9. Access requirements of Example 4 using (a) $GTRBAC_{1,U}$ representation, (b) $GTRBAC_0^1$ representation, and (c) $GTRBAC_0^2$ representation.

$assign_U$ DayDoctor `to` A) in T. Similarly, the assignment constraints for the remaining doctors with the respective periodic expressions are specified.

$GTRBAC_0^1$ **representation**. In this alternative representation, we use algorithm `TransformUR` with the above $GTRBAC_{1,U}$ representation as the input. Accordingly, for each user-role assignment, a role is created, and a *default* assignment and a periodicity constraint on the new role are added. For instance, for a constraint (PE_A, $\texttt{assign}_\text{U}$ DayDoctor`to` A), a role, say r_A, is created and a new constraint (PE_A, `enable` r_A) is added, whereas the constraint (PE_A, $\texttt{assign}_\text{U}$ DayDoctor `to` A) is replaced by *default* assignment ($\texttt{assign}_\text{U}$ r_A `to` A). Similarly, all other temporal assignments are replaced. The result is depicted in Fig. 9b.

$GTRBAC_0^2$ **representation**. This alternative uses the *minimal disjoint* set approach using algorithm `TransformMDS` (see Fig. 7). The result is as shown in Fig. 9c. From Example 2, we know that

$$\begin{aligned} &MDS_{\{PEA,PEB,PEC,PED,PEE\}} \\ &= \{PE_1''', PE_2''', PE_3''', PE_4'''\} \\ &= \{\{\text{Sun, Tues}\}, \{\text{Thu, Fri}\}, \{\text{Mon, Wed}\}, \{\text{Sat}\}\}. \end{aligned}$$

A role is created for each periodic expression of $MDS_{\{PEA,PEB,PEC,PED,PEE\}}$. As

$$|MDS_{\{PEA,PEB,PEC,PED,PEE\}}| = 4,$$

four new roles are created. Each doctor is assigned to a set of new roles which correspond to the periodic expressions that constitute the MS of the periodic expression associated with the doctor (see Example 3), e.g., since

$$MS_{PEC}(MDS_{\{PEA,PEB,PEC,PED,PEE\}}) = \{PE_1''', PE_2'''\},$$

doctor C is assigned to the new roles corresponding to the expressions PE_1''' and PE_2'''.

In the complexity expressions, we ignore the original role and any activation constraints associated with it as they remain the same in all the representations. Note, for the $GTRBAC_{1,U}$ representation, the complexity is $n.T_{UR}$. The following theorem provides complexities associated with the alternative representations using the $GTRBAC_0$ model.

Theorem 4 (Complexity expressions for $GTRBAC_0^1$ and $GTRBAC_0^2$ representations). *Let n be the number of users assigned to a role r and let $PE = \{PE_1, PE_2, \ldots, PE_n\}$ be the set of the periodic expressions in the user-role assignment constraints corresponding to n users assigned to r, i.e., there is a $(PE_i, \texttt{assign}\ r\ \texttt{to}\ u_i)$ for each $i = 1$ to n. The general complexity expressions for the alternative representations $GTRBAC_0^1$ and $GTRBAC_0^2$ are as follows:*

1. *$GTRBAC_0^1$ representation:* $n.S + n.T_R + n.R + n.H$,
2. *$GTRBAC_0^2$ representation:* $p_n.S + s_n.T_R + s_n.R + s_n.H$,

where

$$\begin{aligned} p_n =& |MS_{PE_1}(MDS_{PE})| + |MS_{PE_2}(MDS_{PE})| + \ldots \\ &+ |MS_{PE_n}(MDS_{PE})|, \end{aligned}$$

and $s_n = |MDS_{PE}|$.

See the Appendix (which is available as a free pdf at www.computer.org/portal/pages/transactions/tdsc/content/archives.html) for proof. Based on this theorem, we get the following complexities for each representation of Example 4, which is shown in Fig. 9.

- $GTRBAC_{1,U}$ representation: $5.T_{UR}$.
- $GTRBAC_0^1$ representation: $5.S + 5.T_R + 5.R + 5.H$.

- $GTRBAC_0^2$ representation: $10.S + 4.T_R + 4.R + 4.H$ (using algorithm `TransformUR`).

For the above example, the $GTRBAC_{1,U}$ representation is the best choice in terms of complexity as it has the least number of roles, no hierarchy overhead, and no default assignments. Furthermore, this representation is simple and intuitive to use and, hence, is convenient. The main difference between the $GTRBAC_0^1$ and the $GTRBAC_0^2$ representations is that the latter always produces roles that are temporally disjoint. The $GTRBAC_0^1$ representation, on the other hand, associates one role for each user for whom there is a temporal assignment constraint. However, the $GTRBAC_{1,U}$ representation may not be the best for all the cases, as illustrated below.

Note the complexities of the $GTRBAC_{1,U}$ and $GTRBAC_0^1$ representations remain unchanged for a given n, irrespective of how the periodic expressions are pair wise related. The complexity of the $GTRBAC_0^2$ representations for a given n, however, depends on the MS and MDS of the set PE. The following corollary states the effect of MS and MDS on the complexity of the $GTRBAC_0^2$ representations.

Corollary 3 (Complexity cases for $GTRBAC_0^2$ representations). *Let n be the number of users assigned to a role r, and let $PE = \{PE_1, PE_2, \ldots, PE_n\}$ be the set of the periodic expressions in the user-role assignment constraints corresponding to n users, i.e., there is a $(PE_i, \texttt{assign}_\text{U}\ r \text{ to } u_i)$ for each $i = 1$ to n. Then:*

1. *If $(PE_i \diamond PE_j)$ holds for all i, j pairs such that $1 \leq i, j \leq n$ (i.e., they are pair wise disjoint), then the complexity of*

$$GTRBAC_0^2 = complexity\ of\ GTRBAC_0^1 = n.S + n.T_R + n.R + n.H.$$

2. *If $(PE_i = PE_j)$ holds for all i, j pairs such that $1 \leq i, j \leq n$ (i.e., they are pair wise equivalent), then the complexity of $GTRBAC_0^2 = n.S + T_R + R + H$.*
3. *The worst case for $GTRBAC_0^2$ is*

$$n2^n.S + 2^n.T_R + 2^n.R + 2^n.H.$$

Proof of this corollary is given in the Appendix. The first part of the corollary deals with the case when all the periodic expressions associated with the *user-role* assignments are disjoint. In this case, the $GTRBAC_0^2$ representation is the same as the $GTRBAC_0^1$ representation. When $(PE_i = PE_j)$ for all $i, j = 1$ with n being large, the $GTRBAC_0^2$ representation is substantially better than the $GTRBAC_{1,U}$ representation. This is due to the fact that *temporal constraints* incur more processing costs than the *default* assignments. Note the new role created can be combined with the original role. However, the worst case for $GTRBAC_0^2$ representation, as indicated by the third part of Corollary 3 is $O(2^n)$ in terms of the number of new roles created, temporal constraints on roles, and the number of new hierarchical relations, and $O(n2^n)$ in the number of default assignments. Based on these observations, we can summarize the following design guidelines:

1. The $GTRBAC_{1,U}$ representation is preferable to the $GTRBAC_0^1$ representations; the complexity of the former in terms of the number of roles, the number of temporal constraints, and/or the number of hierarchical relations created is always lower.
2. The $GTRBAC_{1,U}$ and $GTRBAC_0^1$ representations may result in unnecessary temporal constraints because of some common periodic expressions. For example, there may be a large number of doctors who need to use the DayDoctor role at *daytime*, requiring *daytime* a common period for many users. Using the $GTRBAC_0^1$ representations in such cases also results in the same periodicity constraints on the different roles as algorithm `TransformUR` does not attempt to reduce the number constraints based on common periodicity expressions. The $GTRBAC_0^2$ is a good solution in all such cases where some user-role assignments have common periodic expressions. If all the periodic expressions are equivalent, then this model produces a single role and all the users are assigned to that role, as indicated by the results in the second part of Corollary 3. According to Theorem 4 and Corollary 3, $GTRBAC_0^2$ is advantageous when the MS set of each periodic expression is very small (the smallest case is when it has only one member as indicated by the second part of the corollary, i.e., when all the periodic expressions are equivalent). Furthermore, a small MDS set is desirable as it determines the number of newly created roles.

   Similarly, if all the periodic expressions are pairwise disjoint, then $GTRBAC_0^2$ and $GTRBAC_0^1$ representations become equivalent, as shown by the first part of Corollary 3.
3. The $GTRBAC_{1,U}$ representation is highly flexible in terms of access specification since, in addition to the role enabling constraints, it also supports the temporal constraints on *user-role* assignments. For example, consider the following constraints:

   ([Mon, Wed, Fri], $\texttt{assign}_\text{U}$ John `to` DayDoctor).
   ([Tue, Thurs], $\texttt{assign}_\text{U}$ John `to` NightDocotor).
   ([10am, 3pm], $\texttt{assign}_\text{U}$ Greg `to` DayDoctor).

   By using the above constraints, we can keep the roles with enabling times fixed in the system and express the individual user requirements using the periodicity constraints. In this case, system-wide DayDoctor and NightDoctor roles are more or less fixed and, as illustrated, users are assigned to these roles as required. Furthermore, these constraints are semantically much clearer than its $GTRBAC_0^1$ and $GTRBAC_0^2$ forms with temporal constraints on the role enabling only.
4. Note, in case there are *per-user-role* activation constraints, the $GTRBAC_0^2$ representations may not offer any advantage. For example, as shown in Fig. 9c, each user is assigned to multiple new roles. In such a case, if there had been a *per-user-role* constraint for each user, we would have been required to take extra steps during its transformed

representation. Here, we note that the algorithm `TransformMDS` creates an A_s-hierarchy (*strongly restricted activation* hierarchy) between the new roles and the original role. Therefore, if we leave the *per-user-role* constraints unaltered in the transformed representation, the *per-user-role* constraints will still be specified in terms of the original role. The new representation, however, will still be valid, as the users assigned to the new role will have to explicitly activate the new role. However, such semantics is not intuitive, as the users are only implicitly assigned to the original role. Therefore, in presence of the *per-user-role* activation constraints, the $GTRBAC_0^1$ and $GTRBAC_{1,U}$ representations provide more intuitive and convenient representations than the $GTRBAC_0^2$ representation.

5. Unlike the periodicity constraints, replacing the duration constraints on user-role assignments by the duration constraints on role enabling is less flexible. As duration constraints have nondeterministic start times, such constraints depend on some other events. Such dependencies often have some implications on application semantics. Even though it may be possible to replace a duration constraint on *user-role* assignment, care must be taken to ensure that the dependency semantics is not lost, as illustrated by the following example:

   Example 5. Consider the Manager and Employee roles in an enterprise and assume that the Employee role is enabled on weekdays from 9am to 5pm, whereas the Manager role is enabled everyday. At other times, the Employee role is enabled only if Mr. Smith, the manager who is also the owner, has activated his Manager role. This constraint can be expressed using the following trigger:

   (`activate` Manager `for` Smith → `enable` Employee) (t1)

   Suppose Smith needs to allow John, another employee in his office, to work on Saturday and Sunday, when he himself is also working, for at most four hours. Such a change in the policy can be carried out by adding the following constraints:

   (Every *Saturday and Sunday*, 4 hours, $\texttt{assign}_{\text{U}}$ John to Employee) (c_1)
   (`activate` Manager `for` Smith → $\texttt{assign}_{\text{U}}$ John `to` Employee) (t_2)
   (`deactivate` Manager `for` Smith → `disable` Employee) (t_3)

   When Smith activates the Manager role on Saturday, it enables the Employee role using trigger t_1 and assigns John to the Employee role using trigger t_2. Because of the constraint c_1 active at the time, the assignment gets restricted to four hours during which John can work. In this case, if we try to use the duration constraint on the Employee role instead, the implicit dependency between the activation of the Manager role and allowing John to work is lost.

6. We note that the transformation such as in $GTRBAC_0^2$ is not possible for a *user-role* assignment with duration constraints. Although there may be common duration values associated with the different *user-role* assignments, there is an inherent dependency semantic associated with each duration constraint that relates it to a trigger or a constraint enabling expression.
7. Except for guideline 4, all other guidelines are applicable to *role-permission* assignments as well.

Thus, except for some cases where $GTRBAC_0^2$ is better in terms of the complexity of representations, the $GTRBAC_{1,U}$ provides the best representational form in terms of complexity, user convenience, and semantic clarity.

3.3.2 Activation Constraints

In this section, we compare the $GTRBAC_0$ and $GTRBAC_{1,A}$ models in terms of their expressiveness of the same set of activation constraints. For simplicity, we assume that $GTRBAC_{1,A}$ has only *total active duration* constraints in addition to the constraints in $GTRBAC_0$. A similar analysis applies to other activation constraints. In the complexity expressions, we use A_{UR} and A_R to represent *per-user-role* activation constraint and *per-role* activation constraint, respectively, as shown in Table 5. In addition, we do not include the original role and any of its associated *per-role* constraints in the complexity expressions. For the discussion that follows, we use the following example:

Example 6. Let A, B, C, D, and E be the users subscribing for 100, 100, 100, 250, and 50 hours of active time per week, respectively, from a video library. A straightforward representation of these constraints using the $GTRBAC_{1,A}$ model is shown in Fig. 10a. To represent these constraints using $GTRBAC_0$, we can either use the part of algorithm `TransformUR` that removes *per-user-role* activation constraints or we can simply assume that there are no temporal assignment constraints and run the `TransformUR` on this configuration. Such a representation, later referred to as the $GTRBAC_0^s$ representation, is shown in Fig. 10b.

From the example, it is clear that the straightforward representation of a set of n *per-user-role* constraints for n users assigned to a role (a *per-role* constraint on the role may or may not be present), using the two models incur the following costs:

1. $GTRBAC_{1,A}^s$ representation: $n.A_{UR}$. (1).
2. $GTRBAC_0^s$ representation (using algorithm `TransformUR`): $n.A_R + n.R + n.H$. (2)

Note that we have not included the original role and any of the *per-role* constraints on it as they always remain the same. We can note that, between the two cases illustrated above, the $GTRBAC_{1,A}^s$ model gives a better representation in terms of the reduced number of roles. The total number of activation constraints is the same in both. However, it is important to know whether or not these two models give the best representations. Note that, in Fig. 10a, users *A*, *B*, and *C* have the same *per-user-role* access requirements which can be expressed as one *per-role* constraint. Similarly,

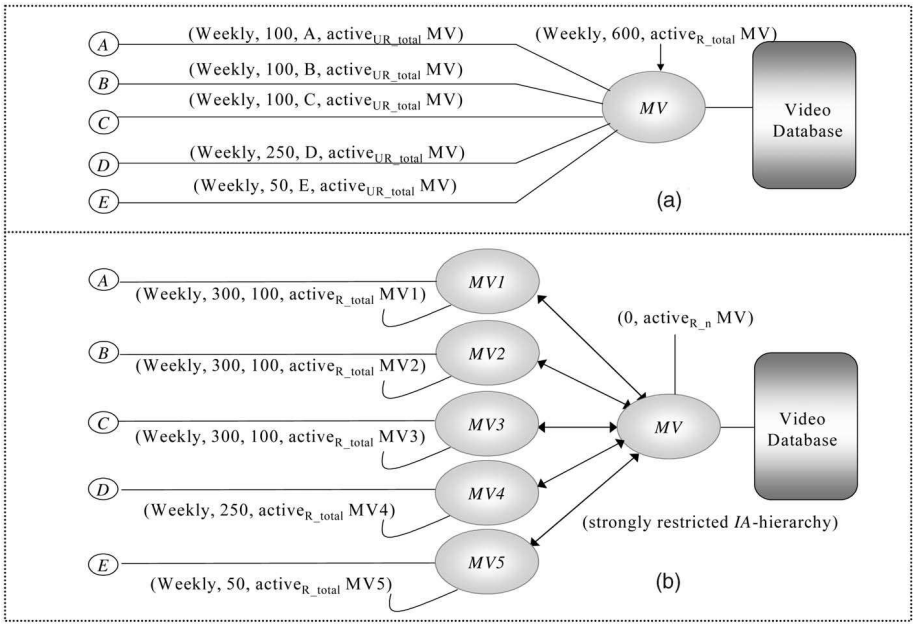

Fig. 10. Access requirements of Example 6 using (a) $GTRBAC^s_{1,A}$ representation and (b) $GTRBAC^s_0$ representation by running algorithm `Transform2` on a $GTRBAC^s_{1,A}$ configuration.

note that, in Fig. 10b, roles *MV1*, *MV2*, and *MV3* have the same *per-role* constraint values, which can be combined. Such common per-user-role values can be used to generate a better representation than the two discussed above. The following theorem formally shows the complexity of the representation by considering such common values.

Theorem 5 (Complexity expression for $GTRBAC_0$ and $GTRBAC_1$ representations). *Let n be the number of users assigned to role r, $D = \{d_1, d_2, \ldots, d_n | d_i$ is the total active duration that the ith user is allowed over role $r\}$, $D_m = \{d'_1, d'_2, \ldots, d'_m\} \subseteq D$ be the set of distinct elements of D, and $C_m(d)$ be the number of times d occurs in D. Then, the complexities of the following two representations are as follows:*

1. *$GTRBAC_{1,A}$ representation: $(n_x - n_y).A_{UR} + n_y.A_R + c.(b.n_y + 1).(R + H)$.*
2. *$GTRBAC_0$ representation: $n_x.A_R + n_x.R + n_x.H$, where*
   - *$n_x = |D_m|$ and $n_y = |D'|$ such that 1) $D' \subseteq D_m$ and 2) if $d \in D'$, then $C_m(d) > 1$.*
   - *$b = 1$ if $(n > n_x)$; $b = 0$ otherwise.*
   - *$c = 1$ if $(n > n_x > 0)$; $c = 0$ otherwise.*

Proof of this theorem is given in the Appendix (which is available as a free pdf at www.computer.org/portal/pages/transactions/tdsc/content/archives.html). The complexities of the previously mentioned representations of the constraints as shown in Figs. 10a and 10b can be derived by forcing each element in D to be considered as unique. Note the values of some d_is in D may be equal. In that case, D' is nonempty and contains those elements of D that occur more than once. Here, $n_x = |D_m| = n$, $n_y = 0$, $b = 0$, and $c = 0$ and, hence, the complexities are as follows:

$GTRBAC^s_{1,A}$ representation:
$(n_x - n_y).A_{UR} + n_y.A_R + c.(b.n_y + 1).(R + H) = n.A_{UR}$ (same as (1)).

$GTRBAC^s_0$ representation:
$= n_x.A_R + n_x.R + H = n.A_R + n.R + n.H$ (same as (2)).

Thus, for Example 6, we have the following complexities, as given by Theorem 5 (the constraints are as shown in Fig. 10):

$GTRBAC^s_{1,A}$ representation: $5.A_{UR}$;
$GTRBAC^s_0$ representation: $5.A_R + 5.(R + H)$.

It can be noted that, in the $GTRBAC^s_0$ representation, there are five temporal constraints for the five new roles and one for the original role. The $GTRBAC^s_{1,A}$ representation has the original roles and five *per-user-role* and one *per-role* constraints. Based on these observations, we summarize the following guidelines.

1. If there are many users having a common active duration requirement, then using a role and a constraint that specifies both the *total* and *default* duration constraint minimizes both the number of roles and the number of temporal constraints, as indicated by Theorem 5.
2. If the expected requirements for the active durations for individual users vary substantially, $GTRBAC_{1,A}$ representation is preferable.
3. If more flexibility is needed in specification, using *per-user-role* constraints (and, hence, $GTRBAC_{1,A}$ representation) is better. For example, if the users *A*, *B*, *C*, *D*, and *E* request different active durations every week, then the use of *per-user-role* constraints is more appropriate.
4. In some cases, a hybrid approach utilizing both the *per-role* and *per-user-role* constraints will give a more

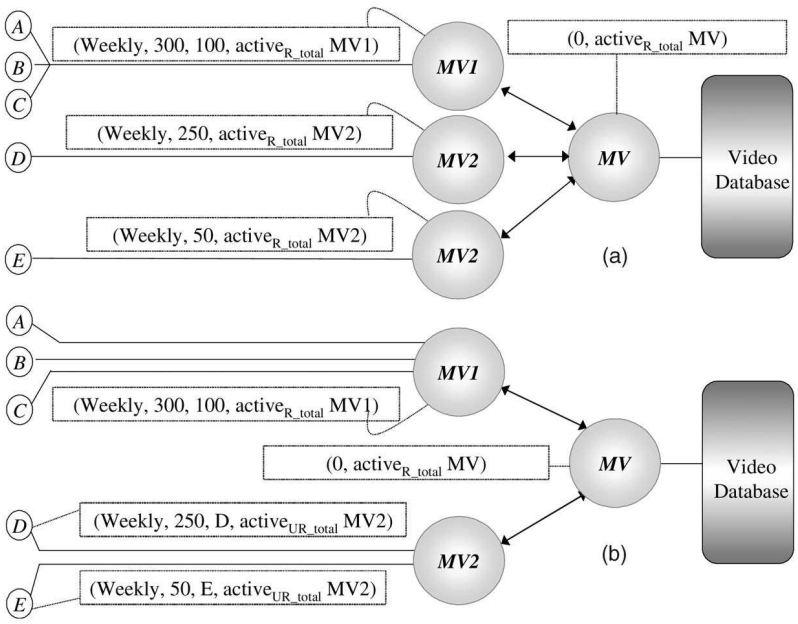

Fig. 11. Constraints of Example 6 (a) using $GTRBAC_0$ representation and (b) using $GTRBAC_{1,A}$ representation.

efficient representation, as shown by Fig. 11b. This is the $GTRBAC_{1,A}$ representation as mentioned in Theorem 5.

Thus, it can be noted that the $GTRBAC_{1,A}$ representation has distinct advantages over the $GTRBAC_0$ representation.

4 Related Work and Discussion

The TRBAC model proposed by Bertino et al. is the first known model that addresses temporal constraints for an RBAC model [5]. It, however, incorporates temporal constraints on the role enabling only and, hence, it does not provide support for a diverse set of requirements. The GTRBAC model presented in [17] is a generalization of the TRBAC model and incorporates an extensive set of new temporal constraint expressions. However, issues such as whether the exhaustive set of the GTRBAC constraints has any practical benefit are not addressed in [17]. Bertino et al. propose a time-based access control model that supports temporal authorization and derivation rules [4]. However, the model does not address roles and assignments and, hence, differs significantly from the GTRBAC model. Many researchers have addressed the need for supporting constraints in an RBAC model [1], [6], [13], [17], [18], [21], [25], [27]. However, they do not address time-based access restrictions. Bertino et al. have proposed a logic-based constraint specification language that can be used to specify constraints on roles and users and their assignments to workflow tasks [6]. However, it also does not include temporal constraints in their specification models. Atluri and Gal have recently proposed a *Temporal Data Authorization Model* (TDAM) that can express access control policies based on the temporal characteristics of the data, such as valid and transaction time [2]. The GTRBAC model discussed in this paper can capture this aspect of authorization by using dynamic role-permission assignments through periodicity and duration constraint, as well as triggers.

To the best of our knowledge, analysis similar to what we have presented in this paper regarding the minimality, expressiveness, and usability of an access control model has not been pursued earlier. Although it deals with a theoretical analysis of policy design issues, the paper's focus has been on analyzing the alternative model configurations for a given policy to investigate the practical design considerations rather than addressing typical policy correctness issues (e.g., safety). With such a formal basis for producing practical policy design guidelines, efficient tools can be developed to assist the task of access control administration. In particular, in current and emerging applications, fine-grained and often complex models, such as the GTRBAC model, are necessary to specify a diverse set of dynamic access control requirements [9], [18], [19], [20]. The fact that capturing a single context attribute such as time in an extended RBAC model has resulted in the fairly complex GTRBAC model provides a glimpse about significant complexity in the model with more generic set of context attributes. In such a case, the issue of expressiveness and usability will become much more complex but crucial. Extension of the analysis presented in this paper can lead to the developing of efficient access control administration tools.

5 Conclusions

In this paper, we have addressed the issue of expressiveness of the GTRBAC model. The main contribution is to address the issue of minimality, complexity of constraint specification, and user convenience of GTRABC models. Through detailed analysis, it is shown that a

comprehensive set of GTRBAC constraints can provide distinct advantages over the minimal GTRBAC model in terms of user convenience and the complexity of constraint representation. This is a significant result in terms of practicality as it shows that the constraints set of a nonminimal GTRBAC model can provide a security policy designer with the flexibility and intuitive choices over various constraint expressions as well as much better and less complex representations in certain cases. Based on these results, we have presented a set of design guidelines that can assist security policy designers in choosing more convenient and less complex constraint expressions. As a future work, we plan to extend the analysis to include time-based separation of duty as well as dependency constraints. Such results can eventually lead to the development of efficient tools for security administration and developing better support for secure e-commerce applications [28]. An important issue related to expressive models like GTRBAC is that of safety. A restricted notion of safety has been presented for the GTRBAC model that is aimed at capturing ambiguous event-based execution semantics [17]. We plan to extend the safety analysis issue in light of the results presented in this paper. Another important future work is to address the privacy issue [28] within an RBAC framework and investigate the expressiveness versus usability issues within the context of information privacy.

ACKNOWLEDGMENTS

This research was partially funded by US National Science Foundation Grants IIS 020911 and IIS 0242419.

REFERENCES

[1] G. Ahn and R. Sandhu, "Role-Based Authorization Constraints Specification," *ACM Trans. Information and System Security,* vol. 3, no. 4, Nov. 2000.

[2] V. Atluri and A. Gal, "An Authorization Model for Temporal and Derived Data: Securing Information Portals," *ACM Trans. Information and System Security,* vol. 5, no. 1, pp. 62-94, Feb. 2002.

[3] J. Barkley, A. Cincotta, D. Ferraiolo, S. Gavrila, and D.R. Kuhn, "Role Based Access Control for the World Wide Web," *Proc. 20th Nat'l Information System Security Conf. (NIST/NSA),* 1997.

[4] E. Bertino, C. Bettini, E. Ferrari, and P. Samarati, "An Access Control Model Supporting Periodicity Constraints and Temporal Reasoning," *ACM Trans. Database Systems,* vol. 23, no. 3, pp. 231-285, Sept. 1998.

[5] E. Bertino, P.A. Bonatti, and E. Ferrari, "TRBAC: A Temporal Role-Based Access Control Model," *ACM Trans. Information and System Security,* vol. 4, no. 4, 2001.

[6] E. Bertino, E. Ferrari, and V. Atluri, "The Specification and Enforcement of Authorization Constraints in Workflow Management Systems," *ACM Trans. Information and System Security,* vol. 2, no. 1, pp. 65-104, 1999.

[7] R. Bhatti, J.B.D. Joshi, E. Bertino, and A. Ghafoor, "XML-Based Specification for Web-Services Document Security," *Computer,* vol. 37, no. 4, Apr. 2004.

[8] R. Bhatti, B. Shafiq, J.B. D. Joshi, E. Bertino, and A. Ghafoor, "X-GTRBAC Admin: A Decentralized Administration Model for Enterprise Wide Access Control," *ACM Trans. Information and System Security,* to appear.

[9] M.J. Covington, W. Long, S. Srinivasan, A.K. Dey, M. Ahamad, and G.D. Abowd, "Securing Context-Aware Application Using Environment Roles," *Proc. ACM Symp. Access Control Models and Technologies,* May 2001.

[10] D.F. Ferraiolo, D.M. Gilbert, and N. Lynch, "An Examination of Federal and Commercial Access Control Policy Needs," *Proc. NISTNCSC Nat'l Computer Security Conf.,* pp. 107-116, Sept. 1993.

[11] E. Ferrari and B. Thuraisingham, "Security and Privacy for Web Databases and Services," *Proc. Int'l Conf. Extending Database Technology,* pp. 17-28, 2004.

[12] L. Giuri, "Role-Based Access Control: A Natural Approach," *Proc. First ACM Workshop Role-Based Access Control,* 1997.

[13] J.B.D. Joshi, W.G. Aref, A. Ghafoor, and E.H. Spafford, "Security Models for Web-Based Applications," *Comm. ACM,* vol. 44, no. 2, pp. 38-72, Feb. 2001.

[14] J.B.D. Joshi, R. Bhatti, E. Bertino, and A. Ghafoor, "An Access Control Language for Multidomain Environments," *IEEE Internet Computing,* pp. 40-50, Nov.-Dec. 2004.

[15] J.B.D. Joshi, A. Ghafoor, W. Aref, and E.H. Spafford, "Digital Government Security Infrastructure Design Challenges," *Computer,* vol. 34, no. 2, pp. 66-72, Feb. 2001.

[16] J.B.D. Joshi, E. Bertino, and A. Ghafoor, "Temporal Hierarchy and Inheritance Semantics for GTRBAC," *Proc. Seventh ACM Symp. Access Control Models and Technologies,* June 2002.

[17] J. Joshi, E. Bertino, U. Latif, and A. Ghafoor, "Generalized Temporal Role Based Access Control Model," *IEEE Trans. Knowledge and Data Eng.,* vol. 17, no. 1, pp. 4-23, Jan. 2005.

[18] A. Kumar, N. Karnik, and G. Chafle, "Context Sensitivity in Role-Based Access Control," *ACM SIGOPS Operating Systems Rev.,* vol. 36, no. 3, pp. 53-66, July 2002.

[19] G. Neumann and M. Strembeck, "An Approach to Engineer and Enforce Context Constraints in an RBAC Environment," *Proc. Eighth ACM Symp. Access Control Models and Technologies,* pp. 65-79, 2003.

[20] M. Niezette and J. Stevenne, "An Efficient Symbolic Representation of Periodic Time," *Proc. First Int'l Conf. Information and Knowledge Management,* 1992.

[21] M. Nyanchama and S. Osborn, "The Role Graph Model and Conflict of Interest," *ACM Trans. Information and System Security,* vol. 2, no. 1, pp. 3-33, 1999.

[22] S. Osborn, R. Sandhu, and Q. Munawer, "Configuring Role-Based Access Control to Enforce Mandatory and Discretionary Access Control Policies," *ACM Trans. Information and System Security,* vol. 3, no. 2, pp. 85-106, May 2000.

[23] J.S. Park, R. Sandhu, and G.J. Ahn, "Role-Based Access Control on the Web," *ACM Trans. Information and System Security (TISSEC),* vol. 4, no. 1, pp. 37-71, Feb. 2001.

[24] R. Sandhu, "Role Activation Hierarchies," *Proc. Second ACM Workshop Role-Based Access Control,* Oct. 1998.

[25] R. Sandhu, "Separation of Duties in Computerized Information Systems," *Database Security IV: Status and Prospects,* pp. 179-189, 1991.

[26] R. Sandhu, E.J. Coyne, H.L. Feinstein, and C.E. Youman, "Role-Based Access Control Models," *Computer,* vol. 29, no. 2, pp. 38-47, Feb. 1996.

[27] R. Simon and M.E. Zurko, "Separation of Duty in Role-Based Environments," *Proc. 10th IEEE Computer Security Foundations Workshop,* June 1997.

[28] B.M. Thuraisingham, C. Clifton, A. Gupta, E. Bertino, and E. Ferrari, "Directions for Web and E-Commerce Applications Security," *Proc. Int'l Workshops Enabling Technologies: Infrastructures for Collaborative Enterprises,* pp. 200-204, 2001.

James B.D. Joshi received the PhD degree in computer engineering from the School of Electrical and Computer Engineering at Purdue University in 2003, the MS degree in computer science from Purdue University in 1998, and the BE degree in computer science and engineering from Motilal Nehru Regional Engineering College, Allahabad, India, in 1993. He is an assistant professor in the Department of Information Sciences and Telecommunications, School of Information Sciences, at the University of Pittsburgh. He is a coordinator of the Laboratory of Education and Research in Security Assured Information Systems (LERSAIS) at the University of Pittsburgh. From 1993 to 1996, he was a lecturer in computer science and engineering at Kathmandu University, Nepal. His research interests are information systems security, database security, distributed systems, and multimedia systems. He is a member of the ACM and the IEEE. He has served as a program committee member of the eighth and ninth ACM Symposium of Access Control Models and Technologies (SACMAT 2003, SACMAT 2004). He was a program cochair of the IEEE Workshop on Information Assurance held in conjunction with the 23rd IEEE International Performance Computing and Communications Conference (IPCCC).

Elisa Bertino is a professor of computer sciences in the Computer Science Department at Purdue University and serves as research director of CERIAS. Previously, she was a faculty member in the Department of Computer Science and Communication of the University of Milan where she directed the DB&SEC laboratory. She has been a visiting researcher at the IBM Research Laboratory (now Almaden) in San Jose, California, at Microelectronics and Computer Technology Corporation, at Rutgers University, and at Telcordia Technologies. Her main research interests include security, privacy, database systems, object-oriented technology, and multimedia systems. In those areas, she has published more than 250 papers in all major refereed journals, and in proceedings of international conferences and symposia. She is a coauthor of the books *Object-Oriented Database Systems—Concepts and Architectures* (Addison-Wesley, 1993), *Indexing Techniques for Advanced Database Systems* (Kluwer Academic, 1997), and *Intelligent Database Systems* (Addison-Wesley, 2001). She is a co-editor-in-chief of the *Very Large Database Systems Journal* (VLDB) and a member of the advisory board of the *IEEE Transactions on Knowledge and Data Engineering*. She also serves on the editorial boards of several scientific journals, incuding *IEEE Internet Computing*, *ACM Transactions on Information and System Security*, *Acta Informatica*, the *Parallel and Distributed Database Journal*, the *Journal of Computer Security*, *Data & Knowledge Engineering*, the *International Journal of Cooperative Information Systems*, and *Science of Computer Programming*. She has been a consultant to several italian companies on data management systems and applications and has given several courses to industries. She has been involved in several projects sponsored by the EU. She has served as a program committee member for several international conferences, such as ACM SIGMOD, VLDB, ACM OOPSLA, as program cochair of the 1998 IEEE International Conference on Data Engineering (ICDE), as program chair of 2000 European Conference on Object-Oriented Programming (ECOOP 2000), and as program chair of the Seventh ACM Symposium of Access Control Models and Technologies (SACMAT 2002). She recently served as program chair of the 2004 EDBT Conference. She is a fellow of the IEEE and ACM and has been been named a Golden Core Member for her service to the IEEE Computer Society. She received the 2002 IEEE Computer Society Technical Achievement Award for "for outstanding contributions to database systems and database security and advanced data management systems."

Arif Ghafoor is currently a professor in the School of Electrical and Computer Engineering, at Purdue University, West Lafayette, Indiana, and is the director of the Distributed Multimedia Systems Laboratory and Information Infrastructure Security Research Laboratory. He has been actively engaged in research areas related to database security, parallel and distributed computing, and multimedia information systems and has published extensively in these areas. He has served on the editorial boards of various journals, including *ACM/Springer Multimedia Systems Journal*, the *Journal of Parallel and Distributed Databases*, and the *International Journal on Computer Networks*. He has served as a guest/co-guest editor for various special issues of numerous journals including *ACM/Springer Multimedia Systems Journal*, the *Journal of Parallel and Distributed Computing*, *International Journal on Multimedia Tools and Applications*, *IEEE Journal on Selected Areas in Communications*, and the *IEEE Transactions on Knowledge and Data Engineering*. He has coedited a book entitled *Multimedia Document Systems in Perspectives* and has coauthored a book entitled *Semantic Models for Multimedia Database Searching and Browsing* (Kluwer Academic, 2000). Dr. Ghafoor is a fellow of the IEEE. He received the IEEE Computer Society 2000 Technical Achievement Award for his research contributions in the area of multimedia systems.