A Temporal Semantic-Based Access Control Model

Ali Noorollahi Ravari, Morteza Amini, and Rasool Jalili

Network Security Center, Computer Engineering Department,
Sharif University of Technology, Tehran, Iran
noorollahi@ce.sharif.edu, m_amini@ce.sharif.edu,
jalili@sharif.edu

Abstract. With the advent of semantic technology, access control cannot be done in a safe way unless the access decision takes into account the semantic relationships between entities in a semantic-aware environment. SBAC model considers this issue in the decision making process. However, time plays a crucial role in new computing environments which is not supported in this model. In this paper we introduce temporal semantic based access control model (TSBAC), as an extension of SBAC model, which enhances the specification of user-defined authorization rules by constraining time interval and temporal expression over users' history of accesses. A formal semantics for temporal authorizations is provided and conflicting situations (due to the semantic relations of the SBAC model and a sub-interval relation between authorizations) are investigated and resolved in our proposed model.

Keywords: Access control, semantic-awareness, temporal authorization, access history.

1 Introduction

Access control is a mechanism that allows owners of resources to define, manage and enforce access conditions applicable to each resource [1]. An important requirement, common to many applications, is related to the temporal dimension of access permissions. In these systems, permissions are granted based on previous authorizations given to the users of the system in specific time points.

Another critical requirement is the possibility of expressing the semantic relationships that usually exist among different authorization elements, i.e. subjects, objects, and actions. To overcome this challenge, our model is constructed based on SBAC model [2] which is a semantic-based access control model. SBAC authorizes users based on the credentials they offer when requesting an access right. Ontologies are used for modeling entities along with their semantic interrelations in three domains of access control, namely subjects domain, objects domain, and actions domain. To facilitate the propagation of policies in these three domains, different semantic interrelations can be reduced to the subsumption relation.

In this paper we propose an access control model characterized by temporal authorizations and based on SBAC model. In the proposed model, a temporal expression is associated with each authorization, identifying the instants in which the authorization applies. Furthermore, a temporal interval bounds the scope of the temporal expressions (e.g., $[1,16]$ shows that the authorization is valid for time interval starting at '1' and ending at '16'). Thus, the main feature provided by our model is the possibility of specifying authorization rules which express temporal dependencies among authorizations. These rules allow derivation of new authorizations based on the presence or absence of other authorizations in specific past time instants (stored in *History Base* in the form of a "time point" and an authorization tuple $(s,o,\pm a)$). By using authorization rules, many protection requirements can be specified. For example, to specify that a user has an authorization as long as another user has this authorization; or that a user should receive the authorization to access an object in certain periods, only if nobody else was ever authorized to access the same object in any instant within those periods.

Besides proposing a basic set of operators to specify temporal dependencies, we introduce a formalism to concisely express various types of dependencies. For example, a single statement can specify that a user can read all the files that another user can read, in a specific time interval.

A formal semantics is defined for temporal authorizations. The subject of Temporal Authorization Base (TAB) administration (by using *addRule* and *dropRule*) and the conflict situations are investigated and resolved.

The rest of this paper is as follows: Section 2 gives a brief introduction of the SBAC model and describes the model of time used throughout our work. In section 3, we represent our authorization rules in detail and offer the formal semantics of them. We also briefly describe the administration of the authorization base and conflict resolution in access decision point. Section 4 describes the related works on this topic and finally, in section 5, we conclude the paper.

2 Preliminaries

In this section we give a brief introduction of SBAC model, proposed by Javanmardi *et al*. [2], and introduce our model of time.

2.1 Introduction to SBAC

Fundamentally, SBAC consists of three basic components: Ontology Base, Authorization Base and Operations. Ontology Base is a set of ontologies: Subjects–Ontology (SO), Objects–Ontology (OO) and Actions–Ontology (AO).

By modeling the access control domains using ontologies, SBAC aims at considering semantic relationships in different levels of ontology to perform inferences to make decision about an access request. Authorization Base is a set of authorization rules in form of $(s,o,\pm a)$ in which s is an entity in SO, o is an entity defined in OO, and a is an action defined in AO. In other words, a rule determines whether a subject which presents a credential s can have the access right a on object o or not.

The main feature of the model is reduction of semantic relationships in ontologies to subsumption relation. Given two concepts C and D and a knowledge base Σ, $C \prec D$ denotes that D subsumes C in Σ. This reasoning based on subsumption proves that D (the subsumer) is more general than C (the subsumee).

By reducing all semantic relationships to the subsumption, the following propagation rules are enough.

– *Propagation in subjects domain*: given $(s_i, o, \pm a)$, if $s_j \prec s_i$ then $(s_j, o, \pm a)$.
– *Propagation in objects domain*: given $(s, o_i, \pm a)$, if $o_j \prec o_i$ then $(s, o_j, \pm a)$.
– *Propagation in actions domain*:
  - Given $(s, o, +a_i)$, if $a_j \prec a_i$ then $(s, o, +a_j)$.
  - Given $(s, o, -a_j)$, if $a_j \prec a_i$ then $(s, o, -a_i)$.

2.2 Model of Time

We assume a discrete model of time. In the database community a chronon usually identifies the smallest indivisible unit of time [3]. We can take a chronon as our time unit in this paper. On this basis, our model of time is isomorphic to the natural numbers $\mathbb{N}$ with the total order relation $\leq$.

It is worthwhile to note that, we suppose that the response time of the access control system is trivial and thus we ignore the time duration required by the system to check whether a requested access is granted or denied. This assumption allows us to take an access request time as the access time recorded in the history.

3 Temporal Semantic Based Access Control Model

In this section we introduce our authorization model, Temporal Semantic base Access Control model (TSBAC), which is an extension of SBAC model. In our model, we extend the basic authorization model in two directions: adding authorization validation time interval, and associating a temporal expression over a *history base*.

3.1 Temporal Authorization Rules

In our model we consider a temporal constraint to be associated with each authorization. This constraint is based on the privileges granted to subjects of the system (on objects), or access requests denied, in a specific *time point* in the past. These elements of history are stored in *History Base*, in the form of $(t, s, o, +a)$ and $(t, s, o, -a)$. We refer to an authorization together with a temporal constraint as a temporal authorization rule. A temporal authorization rule is defined as follows.

Definition 1 (Temporal Authorization Rule): A temporal authorization rule is a triple $([t_s, t_f], (s, o, \pm a), F)$, where $t_s \in N$, $t_f \in N \cup \{\infty\}$ ($t_s \leq t_f$) represents the authorization validation time interval, and formula F is a temporal constraint which is formally defined as in (1).

$$\begin{aligned}
&F ::= \mathit{true} \mid \mathit{false} \mid \mathit{done}(s,o,a) \mid \mathit{denied}(s,o,a) \mid \\
&!F \mid F \wedge F \mid F \vee F \mid F \rightarrow F \mid F \leftrightarrow F \mid \\
&\mathit{prev}(F) \mid \mathit{past}\#(F) \mid H(F) \mid F sb\# F \mid F ab F \mid F ss F \mid F during F
\end{aligned} \quad (1)$$

Temporal authorization rule $\left(\left[t_s,t_f\right],(s,o,\pm a),F\right)$ states that subject s is allowed (or not allowed) to exercise access a on object o in the interval $\left[t_s,t_f\right]$, including time instants t_s and t_f, in the case that F is evaluated to true.

Definition 2 (Temporal Authorization Base (TAB)): A temporal authorization base (TAB) is a set of temporal authorization rules in the form of $([t_1,t_2],(s,o,\pm a),F)$.

Definition 3 (History Base): A History Base is a set of authorizations and time points, in the form of $(t,s,o,+a)$ which means access a has been granted to subject s on object o at time point t, and $(t,s,o,-a)$ which means the system has denied access a on object o at time point t requested by subject s.

Definition 4 (Valid Authorization): an authorization $(s,o,\pm a)$ is valid at time t if one of the following situations occurred:

1. At time t, a temporal authorization rule $([t_1,t_2],(s,o,\pm a),F)$ with $t_1 \le t \le t_2$ exists in TAB and F is evaluated to true based on the elements exist in *History Base* (in section 3.2 we define function f for performing such an evaluation),
2. There exists a temporal authorization rule $([t_1,t_2],(s',o',\pm a'),F)$ in TAB with $t_1 \le t \le t_2$ in which F is evaluated to true, and $(s',o',\pm a')$ is derived from $(s,o,\pm a)$ following the inference rules of SBAC.

The intuitive meaning of temporal authorization rules is as follows. In these statements *auth* is representative of $(s,o,\pm a)$.

– $\left(\left[t_s,t_f\right],auth,true\right)$: Authorization *auth* is always valid in interval $\left[t_s,t_f\right]$.

– $\left(\left[t_s,t_f\right],auth,false\right)$: Authorization *auth* is always invalid.

– $\left(\left[t_s,t_f\right],auth,done(s,o,a)\right)$: Authorization *auth* is valid in all time instants t, in interval $\left[t_s,t_f\right]$ in which $done(s,o,a)$ is evaluated to true.

– $\left(\left[t_s,t_f\right],auth,denied(s,o,a)\right)$: Authorization *auth* is valid in all time instants t, in interval $\left[t_s,t_f\right]$ in which $denied(s,o,a)$ is evaluated to true.

– $\left(\left[t_s,t_f\right],auth,!F\right)$: Authorization *auth* is valid for each time instant t in interval $\left[t_s,t_f\right]$ in which F is *not* evaluated to true.

– $\left(\left[t_s,t_f\right],auth,F_1 \wedge F_2\right)$: Authorization *auth* is valid for each time instant t in the interval $\left[t_s,t_f\right]$ in which F_1 and F_2 are both evaluated to true.

– $\left(\left[t_s,t_f\right],auth,F_1 \vee F_2\right)$: Authorization *auth* is valid for each time instant t in the interval $\left[t_s,t_f\right]$ in which F_1 or F_2 or both of them are evaluated to true.

- $\left(\left[t_s,t_f\right],auth,F_1 \to F_2\right)$: Authorization *auth* is valid for each time instant *t* in the interval $\left[t_s,t_f\right]$ in which if F_1 is evaluated to true, then F_2 is also evaluated to true.
- $\left(\left[t_s,t_f\right],auth,F_1 \leftrightarrow F_2\right)$: Authorization *auth* is valid for each time instant *t* in the interval $\left[t_s,t_f\right]$ in which F_1 is evaluated to true if and only if F_2 is evaluated to true.
- $\left(\left[t_s,t_f\right],auth,prev(F)\right)$: Authorization *auth* is valid at the time of request (*t*) in interval $\left[t_s,t_f\right]$ if *F* is evaluated to true at the previous moment (*t-1*).
- $\left(\left[t_s,t_f\right],auth,past\#(F)\right)$: Authorization *auth* is valid at the time of request (*t*) in interval $\left[t_s,t_f\right]$ if *F* is evaluated to true, at least # times from t_s till *t*.
- $\left(\left[t_s,t_f\right],auth,H(F)\right)$: Authorization *auth* is valid at the time of request (*t*) in interval $\left[t_s,t_f\right]$ if *F* is evaluated to true in all time instants from t_s till *t*.
- $\left(\left[t_s,t_f\right],auth,F_1 sb\# F_2\right)$: Authorization *auth* is valid at the time of request (*t*) in interval $\left[t_s,t_f\right]$ if F_1 is evaluated to true, at least # times before the last occurrence of F_2, in interval $\left[t_s,t\right]$.
- $\left(\left[t_s,t_f\right],auth,F_1 ab F_2\right)$: Authorization *auth* is valid at the time of request (*t*) in the interval $\left[t_s,t_f\right]$ if F_1 is evaluated to true in $t'\left(t_s < t' < t\right)$, then there exist a time point $t''\left(t' < t'' < t\right)$, in which F_2 is evaluated to true.
- $\left(\left[t_s,t_f\right],auth,F_1 ss F_2\right)$: Authorization *auth* is valid at the time of request (*t*) in interval $\left[t_s,t_f\right]$ if F_1 is evaluated to true in all the time points from the first occurrence of F_2 in interval $\left[t_s,t\right]$.
- $\left(\left[t_s,t_f\right],auth,F_1 during F_2\right)$: Authorization *auth* is valid at the time of request (*t*) in interval $\left[t_s,t_f\right]$ if F_1 is not true before the first, or after the last time instant in which F_2 is true.

Another convention that could be useful here is the notion of parametric authorization rules. A parametric authorization rule is an authorization rule where keyword *all* appears for subjects, objects, or access rights in the authorizations. Keyword *all* is a parameter which denotes any subject, object, or access right depending on its position in the authorization.

3.2 Formal Semantics

To formalize the semantics of temporal authorization rules, we first define an evaluation function *f*. This function evaluates the predicate *F* of temporal authorization rules at a time point *t* and based on the elements stored in *History Base*. Function *f* is defined as in (2).

$$
\begin{aligned}
&F \textit{ is defined as in } (1);\\
&\textit{we define an interpretation of function } f \textit{ as follows:}\\
&f\left(t, done(s,o,a)\right) = \begin{vmatrix} true & if\ (t,s,o,+a) \in HB \\ false & if\ (t,s,o,+a) \notin HB \end{vmatrix}\\
&f\left(t, denied(s,o,a)\right) = \begin{vmatrix} true & if\ (t,s,o,-a) \in HB \\ false & if\ (t,s,o,-a) \notin HB \end{vmatrix}\\
&f(t,true) = true; \qquad f(t,false) = false; \qquad f(t,!F) = \neg f(t,F);\\
&f(t,F_1 \wedge F_2) = f(t,F_1) \wedge f(t,F_2); \qquad f(t,F_1 \vee F_2) = f(t,F_1) \vee f(t,F_2);\\
&f(t,F_1 \rightarrow F_2) = f(t,F_1) \rightarrow f(t,F_2); \qquad f(t,prev(F)) = f(t-1,F);\\
&f(t,F_1 \leftrightarrow F_2) = \left(f(t,F_1) \rightarrow f(t,F_2)\right) \wedge \left(f(t,F_2) \rightarrow f(t,F_1)\right);\\
&f(t,past\#(F)) = \exists t_{i_1},\ldots,t_{i_\#} \le t,\ \bigwedge_{k=1}^{\#}\left(f(t_{i_k},F)\right)\\
&f(t,H(F)) = \forall t_i \le t,\ f(t_i,F)\\
&f(t,F_1 sb\#F_2) = \exists t_2 \le t,\ \exists t_{i_1},\ldots,t_{i_\#} \le t_2,\ f(t_2,F_2) \wedge \bigwedge_{k=1}^{\#} f(t_{i_k},F_1)\\
&f(t,F_1 ab F_2) = \left[\exists t_1 \le t, f(t_1,F_1)\right] \rightarrow \left[\exists t_2, t_1 \le t_2 \le t \wedge f(t_2,F_2)\right]\\
&f(t,F_1 ss F_2) = \forall t_2 \le t, \left(f(t_2,F_2) \rightarrow \forall t_i, t_2 \le t_i \le t \rightarrow f(t_i,F_1)\right)\\
&f(t,F_1 during F_2) = \begin{bmatrix} \exists t_{\min}, t_{\min} \le t \wedge f(t_{\min},F_2) \wedge \left(\neg\exists t_x, t_x < t_{\min} \wedge f(t_x,F_2)\right) \wedge \\ \exists t_{\max}, t_{\max} \le t \wedge f(t_{\max},F_2) \wedge \left(\neg\exists t_y, t_{\max} < t_y \wedge f(t_y,F_2)\right) \end{bmatrix} \rightarrow\\
&\qquad\qquad \left[\forall t_1, t_1 < t \rightarrow \left(f(t_1,F_1) \rightarrow t_{\min} \le t_1 \le t_{\max}\right)\right]
\end{aligned}
\tag{2}
$$

Note that a temporal authorization rule can be removed from TAB and therefore not be applicable anymore. For formalizing this issue, we associate with each temporal authorization rule the time t_d at which it is removed. Note that time t_d is not a constant and it is not known from the former. We use it as shorthand for expressing the point, up to which, a temporal authorization rule is applicable.

By the definition of evaluation function *f* and by the assumption described above, the semantics of authorization rules are in (3). In the following, $grant(t,(s,o,a))$ denotes subject *s* is granted to exercise action *a* on object *o* and analogously $deny(t,(s,o,a))$ denotes the access request of *s* for exercising an access *a* on object *o* is denied.

$$
\begin{aligned}
&\left(\left[t_s,t_f\right],(s,o,+a),F\right) \Leftrightarrow \forall t\left(t_s \le t \le \min\left(t_f,t_d-1\right) \wedge f(t,F)\right) \rightarrow grant\left(t,(s,o,a)\right)\\
&\left(\left[t_s,t_f\right],(s,o,-a),F\right) \Leftrightarrow \forall t\left(t_s \le t \le \min\left(t_f,t_d-1\right) \wedge f(t,F)\right) \rightarrow deny\left(t,(s,o,a)\right)
\end{aligned}
\tag{3}
$$

3.3 Access Control and Conflict Resolution

The centric security mechanism in each system is an access control system. By receiving an access request in such a system, we need to make a decision whether to grant the requested access or deny it. Following the proposed model of temporal authorization in the previous sections, by receiving an access request (s_r, o_r, a_r) at time t, the access control system performs the following steps:

1. Determine the explicit and implicit valid authorization rules in TAB at time *t* (following the definition of valid authorization rules),
2. Extract the set of valid authorization rules like $\left(\left[t_s, t_f\right], (s, o, \pm a), F\right)$ which match the access request i.e. $s = s_r,\ o = o_r,\ a = a_r$ (we call this set, MVA),
3. If there exist just positive valid authorization rule(s) like $\left(\left[t_s, t_f\right], (s, o, +a), F\right)$ in MVA, grant the requested access,
4. If there exist just negative valid authorization rule(s) like $\left(\left[t_s, t_f\right], (s, o, -a), F\right)$ in MVA, deny the access request,
5. If there exist both positive and negative authorization rules in MVA, do conflict resolution (following the approach described in section 3.4) and follow the result,
6. If there is not any valid authorization rule, which matches the requested access, follow the default access policy,
7. Record $(t, s, o, +a)$ in case of the requested access is granted and $(t, s, o, -a)$ in case of the access request is denied.

In this model, the default access policy might be *positive* to grant all undetermined accesses or *negative* to deny them. The default access policy is determined by the administrator.

In access control, due to the modal conflicts between the valid matched authorization rules (in the set MVA); it is required to have a conflict resolution strategy to resolve the conflicts. The conflict might be a result of semantic relationships between the entities (i.e. subjects, objects, and actions) and applying the inference rules of SBAC model, or the sub-interval relation between authorizations (i.e. $\left[t_{s_2}, t_{f_2}\right]$ is a sub-interval of $\left[t_{s_1}, t_{f_1}\right]$).

The model supports two predefined strategies for conflict resolution; negative authorization rule takes precedence (NTP) strategy, and positive authorization rule takes precedence (PTP) strategy. Similar to default access policy, the conflict resolution strategy is determined by the administrator.

3.4 Temporal Authorization Base Administration

Authorization rules can be changed upon the execution of administrative operations. In this paper, we consider a centralized policy for administration of authorizations where administrative operations can be executed only by the administrator.

Administrative operations allow the administrator to add, remove, or modify (a *remove* operation followed by an *add* operation) temporal authorizations rules. Each temporal authorization rule in the TAB is identified by a unique label assigned by the system at the time of its insertion. The label allows the administrator to refer to a specific temporal authorization rule upon execution of administrative operations. A brief description of the administrative operations is as follows:

– *addRule*: To add a new temporal authorization rule. When a new rule is inserted, a label (*rule identifier* or *rid*) is assigned by the system.
– *dropRule*: To drop an existing temporal authorization rule. The operation requires as argument, the label of the rule to be removed.

4 Related Work

Access control systems for protecting Web resources along with credential based approaches for authenticating users have been studied in recent years [1]. With the advent of Semantic Web, new security challenges were imposed to security systems. Bonatti *et al.*, in[4] have discussed open issues in the area of policy for Semantic Web community such as important requirements for access control policies. Developing security annotations to describe security requirements and capabilities of web service providers and requesting agents have been addressed in [5]. A concept level access control model which considers some semantic relationships in the level of concepts in the object domain is proposed in [6]. The main work on SBAC, which is the basis for our model, is proposed in [7] by Javanmardi *et al.*. SBAC is based on OWL ontology language and considers the semantic relationships in the domains of subjects, objects, and actions to make decision about an access request.

The first security policy based on past history of events is introduced as Chinese Wall Security Policy (CWSP) [8]. The objective of CWSP is to prevent information flows which cause conflict of interest for individual consultants. Execution history also plays a role in Schneider's security automata [9] and in the Deeds system of Edjlali [10]. However, those works focus on collecting a selective history of sensitive access requests and use this information to constrain further access requests; for instance, network access may be explicitly forbidden after reading certain files. Another approach which considers the history of control transfers, rather than a history of sensitive requests is presented in [11].

In a basic authorization model, an authorization is modeled by a triple $(s,o,\pm a)$, interpreted as "subject s is authorized to exercise access right a on object o". Recently, several extensions to this basic authorization model have been suggested. One of them is the temporal extension of it which increases the expressive power of the basic authorization model [3, 12-15]. Bertino *et al.* [12] specification of temporal parameters. In the model proposed by Bertino *et al.* in [12], an authorization is specified as $(time, auth)$, where $time = [t_b, t_e]$ is a time interval, and $auth = (s,o,m,pn,g)$ is an authorization. Here, t_b and t_e represent the start and end times respectively, during which *auth* is valid. s represents the subject, o the object, and m the privilege. *pn* is a binary parameter indicating whether an authorization is negative or positive, and g represents the grantor of the authorization. This model also allows operations *WHENEVER*, *ASLONGAS*, *WHENEVERNOT*, and *UNLESS* on authorizations. For example, *WHENEVER* can be used to express that a subject s_i can gain privilege on object o whenever another subject s_j has the same privilege on o. later Bertino *et al.* in [14] extended the temporal authorization model to support periodic authorization. They completed their research in [16] by presenting a powerful authorization mechanism that provides support for: (1) periodic authorizations (both positive and negative), that is, authorizations that hold only in specific periods of time; (2) user-defined deductive temporal rules, by which new authorizations can be derived from those explicitly specified; (3) a hierarchical organization of subjects and objects, supporting a more adequate representation of their semantics. From the authorizations explicitly specified, additional authorizations are automatically derived by the system based on the defined hierarchies.

5 Conclusions

In this paper, we presented TSBAC as an access control model that considers temporal aspects of access authorizations in semantic-aware environments like semantic web. The proposed model is an extension of SBAC model, which takes into account the semantic relationships in different levels of subjects, objects, and actions ontologies to perform inferences to make decision about an access request.

TSBAC adds two new elements to SBAC authorization model; temporal intervals of validity, and temporal expression over the history of accesses. This allows us to specify temporal dependencies between authorizations, in specific periods of time. The model is formally defined and its semantics is presented in this paper.

References

1. Samarati, P., Vimercati, S.C.: Access control: Policies, models, and mechanisms. In: Focardi, R., Gorrieri, R. (eds.) FOSAD 2000. LNCS, vol. 2171, pp. 137–196. Springer, Heidelberg (2001)
2. Javanmardi, S., Amini, A., Jalili, R.: An Access Control Model for Protecting Semantic Web Resources. In: Web Policy Workshop, Ahens, GA, USA (2006)
3. Bertino, E., Bettini, C., Samarati, P.: A temporal authorization model. In: Second ACM Conference on Computer and Communications Security, Fairfax, Va (1994)
4. Bonatti, P.A., Duma, C., Fuchs, N., Nejdl, W., Olmedilla, D., Peer, J., Shahmehri, N.: Semantic web policies: a discussion of requirements and research issues. In: Sure, Y., Domingue, J. (eds.) ESWC 2006. LNCS, vol. 4011, pp. 712–724. Springer, Heidelberg (2006)
5. Rabitti, F., Bertino, E., Kim, W., Woelk, D.: A Model of Authorization for Next-Generation Database Systems. ACM TODS 16 (1991)
6. Qin, L., Atluri, V.: Concept-level access control for the Semantic Web. In: 2003 ACM workshop on XML security (2003)
7. Javanmardi, S., Amini, A., Jalili, R., Ganhisafar, Y.: SBAC: A Semantic-Based Access Control Model. In: NORDSEC 2006 (2006)
8. Brewer, D.F.C., Nash, M.J.: The Chinese Wall Security Policy. In: IEEE Symposium on Security and Privacy, Oakland, California (1989)

9. Dias, P., Ribeiro, C., Ferreira, P.: Enforcing History-Based Security Policies in Mobile Agent Systems (2003)
10. Edjlali, G., Acharya, A., Chaudhary, V.: History-based access control for mobile code. In: 5th ACM conference on Computer and communications security (1998)
11. Abadi, M., Fournet, C.: Access control based on execution history. In: 10th Annual Network and Distributed System Security Symposium (2003)
12. Bertino, E., Bettini, C., Ferrari, E., Samarati, P.: A temporal access control mechanism for Database Systems. IEEE Trans. Knowl. Data Eng. 8, 67–80 (1996)
13. Thomas, R.K., Sandhu, R.S.: Sixteenth National Computer Security Conference. Baltimore, Md. (1993)
14. Bertino, E., Bettini, C., Ferrari, E., Samarati, P.: An access control model supporting periodicity constraints and temporal reasoning. ACM Trans. Database Systems 23, 231–285 (1998)
15. Ruan, C.: Decentralized Temporal Authorization Administration. CIT/27/2003 (2003)
16. Bertino, E., Bonatti, P.A., Ferrari, E., Sapino, M.L.: Temporal authorization bases: From specification to integration. Journal of Computer Security 8, 309–353 (2000)