Multi-level authorisation model and framework for distributed semantic-aware environments

M. Amini R. Jalili
Sharif Network Security Center (NSC), Department of Computer Engineering, Sharif University of Technology, Azadi Avenue, Tehran, Iran
E-mail: m_amini@ce.sharif.edu

Abstract: Semantic technology is widely used in distributed computational environments to increase interoperability and machine readability of information through giving semantics to the underlying information and resources. Semantic-awareness, distribution and interoperability of new generation of distributed systems demand an authorisation model and framework that satisfies essential authorisation requirements of such environments. In this study, the authors propose an authorisation model and framework based on multi-security-domain architecture for distributed semantic-aware environments. The proposed framework is founded based on the MA(DL)² logic, which enables policy specification and inference (based on the defined semantic relationships) in both conceptual and ground (individual) levels. Also, it enables authorities to have cooperative security management in their shared domain of resources with different administration styles.

1 Introduction

Semantic wave is a new wave in information evolution that entailed a revolution in distributed computing environments. ‘Semantic technology’ [1], which caused such a wave to be arisen, is a collection of software standards and methodologies aimed at providing more explicit meaning for the information in a computational environment. In fact, semantic technology encodes meanings in an abstract conceptual layer, which is separated from the underlying data and content files, and application codes layer. A ‘semantic-aware environment (SAE)’ is a computational environment, where semantic technology is leveraged to provide meaning (semantics) for its resources and elements, and to facilitate interoperability and resource sharing. The examples of SAEs are semantic web, Web 3.0, semantic grid and semantic cloud computing [1].

Shifting from current computing environments to the semantic-aware ones imposes new security and authorisation requirements. In SAEs, we need to have authorisation policy specification and inference in both conceptual (abstract) level and ground (individual data or content) level. Policy inference regarding semantic relationships in the abstract layer (named ‘ontology’) is the most challenging problem in security models proposed for the conceptual level in SAEs [2–4]. For example if we have a permission rule defined over the resource concept ‘article’, should be propagated to the resource concepts subsumed by ‘article’, such as ‘journal article’ and ‘conference article’. If we take the obligations, prohibitions and gratuities other than permissions and their possible conflicts into account and also consider the effect of semantic relationships in subjects, objects and actions, this kind of policy propagation becomes a complex problem.

Different authorisation models and policy specification languages have been proposed for SAEs, especially for semantic web (e.g. Rei [5], KAoS [6, 7], XACML [8], CLAC [9], and SBAC [10]). However, to the best of our knowledge, none of them could satisfy the essential aforementioned security requirements of SAEs completely and based on the clear semantics. Thus, the frameworks founded based on them (e.g. Rein [11] and KAoS framework [12]) inherent the weaknesses of the models their function based on. For example, it is not clear what is
the meaning of defining a ‘permission’ for a subject of concept $S$ to do an action of concept $A$ on an object of concept $O$, what is the logical relationship between such a permission rule and a prohibition or obligation rule defined over the concepts that have intersection with the concepts in the permission rule. This is resulted from this fact that some of the existing models are based on the description logics that do not support $n$-ary predicates over the concepts (such as OWL-DL in Rei [5] and KAoS [6, 7]) or the policy propagation rules leveraged in some of them are defined manually without a formal and clear basis that guarantees the correctness and consistency of the rules (such as SBAC [10] and CLAC [9]).

MA(DL)$^2$ [2, 13] is a logic of security policy specification and inference in SAEs. The logic is the combination of description logic and poly-modal version of standard deontic logic adopted for policy specification by different authorities in different security domains. Using this logic, we can easily specify $n$-ary predicates over the described concepts and define normative statement (associated to a special authority in a special domain) over the $n$-ary predicates. In fact, this logic has been proposed to provide a basis for a security model satisfying the essential security requirements in SAEs. In this paper, a comprehensive authorisation model and framework based on the MA(DL)$^2$ language family [13] is proposed. The proposed framework has a multi-security-domain (MSD) structure and allows us to specify and infer security policy rules (SPR) in distributed manner. It is also possible to have cooperative security management in this framework that is applicable in virtual organisations (VOs) [14] and federated systems.

The rest of the paper is organised as follows: we continue the paper by introducing related work in the next section. The authorisation requirements of SAEs as well as the overall structure on which we propose our authorisation model and framework are discussed in Section 3. Formal specification of our proposed authorisation model based on the paradigms considered in MA(DL)$^2$ language and investigated authorisation requirements is presented in Section 4. Section 5 discusses how to specify SPR in both conceptual (abstract) and ground (individual) levels. Policy administration, conflict detection and resolution, and access control procedure are illustrated in Section 6. Section 7 presents the current implementation of an authorisation system based on the proposed model and framework, and shows the evaluation results. Section 8 describes a case study on distributed semantic digital library and presents the experimental results obtained by the evaluation of the implemented system in this case. Finally, Section 9 concludes the paper.

2 Related work

There exist different models and frameworks that might be used for authorisation in SAEs. Each of them covers parts of the authorisation requirements of these environments.

The most related works are researches on security in semantic web.

Rei [5] by Kagal et al. and KAoS [6, 7] by Uszok et al. are two famous policy specification languages proposed for semantic web based on deontic logic concepts. None of these languages can fully reason about security policy based on the semantic relationships in the conceptual level. In fact, these languages make policy specification possible in the individual level. For both of these languages, policy management and enforcement frameworks [11, 12] have been proposed. XACML is another policy language proposed by Moses in OASIS [8]. OASIS proposed a framework, in which the language can be also employed. Not employing semantic relationships is the main drawback of this language and framework. Some attempts, such as the one by Damiani et al. [15], were done to employ semantic relationships in their policy language. However, no considerable success was achieved. Priebe et al. [16] extended the XACML framework [8] with ontologies of user, resource, and environment attributes and inference over them. The model just uses ontologies to capture semantic relationships between attributes and, similar to the core XACML, does not support policy inference.

Besides the above policy languages and their frameworks, some research has been done on policy specification and inference in the conceptual level. Qin et al. [9] proposed a concept-level access control (CLAC) model that takes into account some semantic relationships in the objects domain. The more complete model to this regard is our proposed access control model, named SBAC [10, 17]. SBAC considers inferable subsumption relationships in all domains of access control (i.e. subjects, objects and actions). Both CLAC and SBAC models suffer from being centralised model and the lack of a clear semantics for their policy inference rules. Thus, they lack a proof for the soundness and consistency of their inference rules.

The semantic access control (SAC) model [18, 19], proposed by Yague et al., is a certificate-based model for applying semantic web layers (by four meta models) to access control in different environments. Although SAC tackles the problems of distribution and multi-granularity of policies, it does not consider the effect of semantic relationships on policy propagation. Analogous to the SAC model, Naumenko in his thesis [20] tried to adopt semantic web standards for the creation of unified view on the access control area. This model has the same problem as the SAC model.

Apportioning the environment into multiple security domains is an acceptable approach to make distributed security management possible in new computing environments including semantic-aware ones. This approach is employed in security framework proposed in [21] for a pervasive computing environment. In each domain, a security agent (called virgin) works as a proxy of services in
the domain, and controls the requested accesses to the resources and services.

In MSD environments, because of resource sharing or required collaborative activities, cooperative administration is a fundamental security requirement. Pearlman et al. [14] introduce VOs and virtual communities, in which collaborative activities are made possible through resource sharing among multiple institutions. They address policy specification and enforcement in VOs as a key problem in such environments. The cooperative security management (i.e. cooperative policy specification and enforcement) is considered in some of the researches with partial solutions for shared resources in MSD environments (see e.g. [22] by Joshi et al. [23] by Demchenko et al. and [24] by Tang et al.).

MA(DL)$^3$ [2, 13] is a logical language that we proposed for the conceptual-level policy specification and inference in SAEs. In further extensions of basic MA(DL)$^3$ [2] (i.e. MA(DL)$^3$[U]) [13]), policy specification in multiple security domains by different authorities is taken into account. This logical language enables us to infer security policies for cooperative management purpose based on different management styles (similar to the approach we proposed in [25]). In this paper, we present a complete policy specification and enforcement model and framework for multiple security domain SAEs based on the MA(DL)$^3$ language family. Capability of policy specification in both conceptual and ground (individual) level, supporting multiple security domains as well as cooperative management (for VOs), and considering semantic relationships in policy inference based on a sound and complete logical basis are the key features of the proposed model and its framework.

3 Authorisation framework

To propose an appropriate model and framework for SAEs, we need to investigate the authorisation requirements of such environments.

3.1 Requirements of authorisation model for SAEs

Since the model presented in this paper is a ‘general’ model for SAEs (that covers a broad range of semantic computational environments), we should limit ourselves to the authorisation requirements that are common in all of them. It is clear that for a special SAE (e.g. semantic web) we may have other minor authorisation requirements beside the ones listed here.

The most essential authorisation requirements that should be considered in an authorisation model for an SAE are, but not limited to, the following:

R1 – Access policy rules should be defined in different levels of granularity and in both conceptual and ground (individual) levels.

R2 – It should be able to specify policies in distributed manner, so that it would be scalable as well.

R3 – Obligation policy specification as well as access policy [26, 27], and considering the logical relations between obligations, permissions and prohibitions should be supported.

R4 – Owing to the existence of semantic relationships between entities in such environments, the model needs to take semantic relationships in different domains of entities (e.g. subjects, objects and actions) into account in policy propagation and policy inference [3, 4].

R5 – It should have a formal clear semantics for its policy specification language. This ensures the soundness of the developed policy inference and propagation rules in the model.

R6 – It should support the specification of subjects based on their attributes, not their identities in policy rules [28, 29], because the users or subjects who want to access the resources in such open and distributed environments are not predefined in most cases.

R7 – It should be context-aware to dynamically activate policy rules based on the environmental (contextual) conditions [30, 31].

R8 – It should detect and resolve the probable conflicts between explicit and implicit policy rules [3, 5, 32, 33].

R9 – It should provide some administrative facilities for easier policy management, such as exception policy specification, delegation of administration and supporting the definition of roles and groups.

3.2 Overall framework

As shown in Fig. 1, an SAE is divided into a number of security domains in our framework. Each security domain (see Fig. 2) contains a set of under-protection resources, which are registered in the security domain. The authority of the domain specifies the SPR (in both conceptual and ground levels) for resources (objects) registered in the domain. The authority might be either a real primitive authority or a virtual composite authority based on the nature of the security domain (more details are presented in Section 4). A security agent in the domain infers and enforces the SPR (specified by the authority).

3.3 Security agent architecture

For implementing a security agent in a security domain, we suggest the architecture shown in Fig. 3. The architecture is developed by adopting the existing standard security frameworks, that is, ITU-T access control framework [34] and XACML framework [8]. The main components of the proposed architecture of a security agent are as follows:
Policy Administration Point (PAP) provides an interface for authorities to state SPR, to negotiate with other authorities to determine the cooperative management style applicable for cooperative and shared domains, and to configure the meta policy of the domain. At the request of PDP (introduced in the following), PAP may require to communicate with other domains’ PAPs (through inter-domain communication and negotiation channel) to fetch the policy rules specified over the shared resources for cooperative administration.

Policy Decision Point (PDP) decides about the access requests (using the procedure described in Section 6.3) and determines the obligation rules related to the access request. To this aim, PDP infers applicable SPR using MA(DL)$^2$ inference engine.

Policy Enforcement Point (PEP) receives a subject’s request, performs access control and enforces applicable obligations determined by PDP. PEP contains

- Access Control enforcement Point (ACP), which works as a proxy of resources and controls all the access requests,
- Obligation Enforcement Point (OEP), which enforces the obligations related to the system or the access requester.

MA(DL)$^2$ SKB (Security Knowledge Base) contains specification of ontologies (of subjects, resources or objects, and actions), assertions about the individuals, SPR and current context information (CI).

---

**Figure 1** Overall authorisation framework for an SAE

**Figure 2** Overall framework of a security domain
Context handler gathers required CI from context sensors and inserts it as a set of contextual propositions into the SKB.

Credential verifier verifies the validity of the provided credentials (in an access request) using a source of authority (SOA) (Readers may refer to [35] for more information about the components of privilege management infrastructure.).

The proposed architecture enables the security agent to verify users' certificates, infer the applicable SPR, and enforce them. The details of access control procedure that a security agent follows is described in Section 6.3.

3.4 Canonical resource model

We have a canonical model for under-protection resources in a security domain. In the canonical model (shown in Fig. 4), each resource has a resource service provider. The resource service provider has an standard API for receiving and replying the access requests from subjects. We leverage web service API as an standard API in our model.

A web service specifies a collection of services. Each possible action on a resource (object) is considered as a service in this model. Using OWL-S [36], we can specify the properties of services (actions) of a resource in the canonical model.

By taking an action as a service in the canonical model, the ground (individual) level policy rules related to the action on the resource are annotated in ‘service profile’ (in OWL-S) of the service (see [37] for more details on policy rule annotation). Ground (individual)-level policy rules are introduced in Section 5.2.

3.5 Running case study – (SVO)

There are different SAEs that can be taken as a case study to employ the proposed model for authorisation. Examples are semantic grid (and virtual organisation), semantic peer-to-peer, semantic social networks and semantic web. Between these cases, semantic virtual...
organisation (SVO) based on the semantic grid platform is considered as a case study, because it uses all the features and capabilities of the model proposed based on the MA(DL)² logic.

Grid computing environments provide a platform for multiple institutes to share their resources in a wide range. Grid technology in combination with semantic technology constitutes semantic grid [38–40], where grid resources and services are described by explicit semantics. The descriptions in the forms of ontologies enables computers and people to easily discover, aggregate and use the resources and work in cooperation [38, 41].

Semantic grid is in fact a multi-institutional (multi-domain) environment, where the authorities (e.g. the owner) of different institutional or individual domains need to specify their authorisation policies in both conceptual (semantics) layer and ground (individual) level. Since a grid environment is a combination of domains of resources with variety of users who are unknown to other domains, it is required to identify users based on their attributes specified in some credentials (see e.g. the approach taken in Globus [42], or its extension for semantic grid oriented e-tourism [43]).

The grid has emerged as a platform that enables multiple institutions/organisations/groups/individuals to build a shared space known as VO to achieve a shared goal by collaboration [44]. A VO encompass users and resources supplied by the different partners for achieving the VO's creation goal. Employing semantic technology in a collaborative problem solving in VOs (based on semantic grid platform), entails SVOs [45]. In this paper, we define an SVO as a running case study.

Company \(A\) as an IT company wants to outsource the implementation of one of its IT projects to cheap programmers in the world (with more priority to the native ones). For this purpose, \(A\) collaborates with company \(B\) as a grid resource provider company to provide the required resources for programmers in an SVO. In this case, \(d_B\) is a security domain of \(B\) for its shared resources, \(d_A\) is a security domain determined by \(A\) for the project and \(d_X\) is the shared subdomain of \(d_A\) and \(d_B\).

The examples presented in the rest of this paper are organised based on this case study.

4 Authorisation model – formal specification

Following the overall framework proposed for security in SAEs, and narrative description of overall models of its main elements, we formally define the authorisation model of SAEs in the rest of this section. The following model is obtained by adapting and completing the model proposed for MSD environments in [25] and the basic model proposed for SAEs in [2].

**Definition 4.1 (Authorisation model of SAEs):** A multi-level authorisation model of an SAE is a four-tuple \((\mathcal{FDS}, \mathcal{SDS}, \mathcal{PAO}, \mathcal{ACP})\), where

1. \(\mathcal{FDS} = (\mathcal{S}, \mathcal{O}, \mathcal{A}, \mathcal{DS}, \mathcal{X}, \mathcal{U}, \mathcal{D})\) is the fundamental data set of the model containing the following elements:
   - \(\mathcal{S}\) (resp. \(\hat{\mathcal{S}}\)) as a type (resp. set) of subjects,
   - \(\mathcal{O}\) (resp. \(\hat{\mathcal{O}}\)) as a type (resp. set) of objects or resources,
   - \(\mathcal{A}\) (resp. \(\hat{\mathcal{A}}\)) as a type (resp. set) of actions or operations on resources,
   - \(\mathcal{DS}\) as the set of deontic statuses \((\mathcal{DS} = \{\mathcal{OB}\text{(obligatory that)}, \mathcal{PE}\text{(permissible that)}, \mathcal{IM}\text{(impermissible that)}, \mathcal{GR}\text{(gratuitous that)}\})\),
   - \(\mathcal{X}\) as a set of contextual propositions,
   - \(\mathcal{U}\) as a set of primitive authorities, \(\mathcal{U}'\) denotes the closure set of composite authorities obtained from \(\mathcal{U}\) [25]. Primitive authorities (belong to \(\mathcal{U}\)) are also composite authorities. If \(u_i\) and \(u_j\) are composite authorities, \(u_i|u_j\) (disjoint composition), \(u_i\&u_j\) (joint composition) and \(u_i\triangleright u_j\) (delegative composition) are composite authorities as well,
   - \(\mathcal{D}\) as a finite set of security domain names.

2. \(\mathcal{SDS} = \{\mathcal{SD}_1, \mathcal{SD}_2, \ldots\}\) is a set of security domains in the environment. A security domain is formally defined in Definition 4.2.

3. \(\mathcal{PAO} = \{\mathcal{hFPR}, \mathcal{dFPR}, \mathcal{CPR}, \mathcal{WPC}, \mathcal{SPC}\}\) is a set of policy administration operators that are used by the primitive authorities. In this set, forced policy revision (FPR), consistent policy revision (CPR), weak policy contraction (WPC), and strong policy contraction (SPC). More details are described in Section 6.2.

4. \(\mathcal{ACP}\) is an access control procedure that is used by security agents to infer and enforce SPR. The detailed steps of \(\mathcal{ACP}\) are presented in Section 6.3.

To formally define a security domain (belongs to SDS), we require to define a SKB at first.

**Definition 4.2 SKB:** An SKB is an MA(DL)² security knowledge base (introduced in [13]), which is defined as a three-tuple \((\mathcal{TB}, \mathcal{AB}, \mathcal{SB})\), where \(\mathcal{TB}\) is the terminology box (includes ontologies of subjects, objects and actions), \(\mathcal{AB}\) is the assertional box and \(\mathcal{SB}\) is the SBR box. As is described later in this paper, \(\mathcal{SB}\) includes a set of conceptual and ground-level policy rules and other related facts such as contextual information.
As shown in Fig. 2, an SKB is locally defined for each security domain. Note that in SKBs, the TB component is the common knowledge between different domains. It defines the ontologies of participating entities in an SAE, and thus it is a common specification of conceptualisation which is shared in different security domains; however, the other two components are domain-specific knowledge.

**Definition 4.3 (Security domain):** A security domain SD $i \in SD$ is formally defined as a five-tuple $SD_i = (d_i, u_i, O_i, K_i, MP_i)$, where $d_i \in D$ is the name of the security domain, $u_i \in U^*$ is the authority of the security domain (whose policy rules are applied), $K_i$ is the local SKB of the security domain, $O_i \subseteq O$ is the set of under-protection objects registered in the security domain, and $MP_i$ is the meta policy of the security domain, which is defined in the following.

$O_j \subseteq O_i$ if and only if $u_l \in U^*$ is the authority of the security domain; however, the policy rules of the authority who is determined for the domain are applied by the security agent.

By the above formal definition of security domains, a security domain $SD_j = (d_j, u_j, O_j, K_j, MP_j)$ is a subdomain of security domain $SD_i = (d_i, u_i, O_i, K_i, MP_i)$ if and only if $O_j \subseteq O_i$. A security domain $SD_d = (d_d, u_d, O_d, K_d, MP_d)$ is the shared domain of security domains $SD_i$ and $SD_j$, if and only if $O_l \subseteq O_i \cap O_j$.

A security domain like $SD_i$ (with identity $d_i$) is called an individual domain, if $u_i$ is a primitive authority (i.e. $u_i \in U$) and it is called a cooperative domain, if it is a shared or subdomain and $u_i$ is a composite authority (i.e. $u_i \in U^*$).

**Definition 4.4 (Meta policy):** Meta policy is the policy about the SPR and it is defined as a two-tuple (ResSt, DefAcc), where ResSt $\in \{PO, NO\}$ is a resolution strategy (NO and PO strategies are defined in Section 6.1), and DefAcc = {Grant, Deny} is the default access decision, which is used in access control procedure described in Section 6.3.

Note that each authority can state policy rules over each security domain; however, the policy rules of the authority who is determined for the domain are applied by the security agent.

By the above formal definition of security domains, a security domain $SD_j = (d_j, u_j, O_j, K_j, MP_j)$ is a subdomain of security domain $SD_i = (d_i, u_i, O_i, K_i, MP_i)$ if and only if $O_j \subseteq O_i$. A security domain $SD_d = (d_d, u_d, O_d, K_d, MP_d)$ is the shared domain of security domains $SD_i$ and $SD_j$, if and only if $O_l \subseteq O_i \cap O_j$.

A security domain like $SD_i$ (with identity $d_i$) is called an individual domain, if $u_i$ is a primitive authority (i.e. $u_i \in U$) and it is called a cooperative domain, if it is a shared or subdomain and $u_i$ is a composite authority (i.e. $u_i \in U^*$).

**Example 4.5:** In the SVO case study, the elements of FDS in the model are defined as follows:

- $\hat{S}$ set of programmers
- $\hat{O} = \{\text{Proc}_c, \ldots, \text{Proc}_{10}, \text{Mem}_1, \ldots, \text{Mem}_{20}, \text{Stg}_G, \ldots, \text{Stg}_{20G}\}$
- $\hat{A} = \{\text{read}, \text{write}, \text{lock}, \text{unlock}, \text{exec}, \text{wait}\}$
- $\hat{X} = \{\text{isWorkTime}, \text{is1stOct09To30thDec09}, \text{isHoliday}\}$
- $\hat{U} = \{\text{adm}_A, \text{adm}_B\}$

The set of security domains is defined as $SD = \{SD_A, SD_B, SD_X\}$, where shared domain $SD_X$ is defined as follows

$$SD_X = \langle d_X, adm_d, adm_B, \hat{O}, K_X, (\text{NO, Deny}) \rangle$$

$SB_X$ in SKB $K_X$ is defined later in the paper, where the SPR are defined.

### 5 Security policy specification

We use MA(DL)$^2$ [2, 13] language family for policy specification in our proposed authorisation model. MA(DL)$^2$ is the combination of deontic logic and description logic that enables authorities of different security domains to specify their SPR in the conceptual level (over the provided abstract semantic layer) in terms of deontic statuses: obligation, prohibition, permission and gratuitousness. The logical foundation of this language enables it to infer implicit SPR from the explicit ones based on the semantic relationships defined in ontologies of subjects, objects and actions.

We describe in the following how to use MA(DL)$^2$ to specify policy rules in both conceptual and ground levels.
5.1 Conceptual level security policy

A security policy is a set of SPR that are defined in conceptual level as follows:

Definition 5.1 (Conceptual level policy rule): A conceptual-level policy rule is a formula in MA(DL)\(^2\) with the schema of the form \(\alpha \rightarrow d_s d_A d_o (S, O, A)\), where \(\alpha\) is a formula specifying the contextual constraint, and it is a logical combination of the contextual propositions (\(\alpha\) can be defined more complicated; however, we limit it here.), \(d_s \in DS\) is a deontic normative status, \(s \in U^*\) is an authority, \(d \in D\) is the name of the security domain, where the policy rule is defined, do is a ternary predicate that determines a possible access of a subject to a resource (object) for doing an action, \(S\) of type \(S\) using interpretation function \(I\) is defined as \(C^I \subseteq \sigma\), and interpretation of an individual \(x\) of type \(\sigma\) is defined as \(x^I \in \sigma^I\) is a concept of subjects, \(O\) of type \(O\) is a concept of under-protection resources (objects) and \(A\) of type \(A\) is a concept of actions (Note that in MA(DL)\(^2\), the interpretation of a concept \(C\) of type \(\sigma\) (e.g. type \(S\)).

Example 5.2: Some samples of the conceptual-level policy rules for the SVO case study are as follows:

- Regarding the contract between the two companies, company \(B\) allows access to the shared resources just during the project (1 October 2009 till 30 December 2009). For example, we have the following rule for the processors

  \[
  \text{is1stOct09To30thDec09} \quad \Rightarrow \quad PE_{adm\uparrow@d_s}(\text{SysProg}, \text{Processor}, \text{ProcAct})
  \]

- Foreign programmers cannot access the resources in working time of working days (\(T_O\) and \(T_A\) refer to concept of all objects and concept of all actions, respectively).

  \[
  \neg \text{isHoliday} \land \text{isWorkTime} \quad \Rightarrow \quad IM_{adm\downarrow@d_s}(\text{ForeignProg}, T_O, T_A)
  \]

- System programmers are allowed to access the 64 bits processors.

  \[
  \neg \text{isHoliday} \land \text{isWorkTime} \land \text{is1stOct09To30thDec09} \quad \Rightarrow \quad PE_{adm\downarrow@d_s}(\text{SysProg}, 64bitsProc, \text{ProcAct})
  \]

5.2 Ground level security policy

The ground-level policy is taken into consideration, because we may require to specify finer grained policy rules for instances of the resources registered in the domain. Also we may need to have contextual constraints for access in the ground level beside the ones specified in the conceptual level, and there might exist policy rules from legacy systems in the ground level, which should be imported in the new security system founded on our proposed model.

Definition 5.3 (Ground-level policy rule): A ground-level policy rule for an action \(a\) on an under-protection resource (object) \(o\) is specified as a six-tuple \((+u, d, cx, \text{Reqs}, \text{Caps})\), where

- \((+)\) represents permission and \((-)\) represents prohibition.
- \(u, d, \text{and } cx\) are similar to the ones defined in conceptual-level policy rule.
- \(\text{Reqs} = \{S_i]\{(S; S) \in C\}\) is a set of subject concepts. It shows credentials which a subject requires to present in order to access the object. In other words, it determines the classes of subjects (in the defined ontology) that are allowed to access the resource.
- \(\text{Caps} = \{C_i\} \{C; O \in C\}\) is a set of attributes that the resource supplies when the security policy is satisfied. In other words, it determines the classes of objects (in the defined ontology) that the object belongs to, based on its attributes. It is worthwhile to note that in prohibition rules (where the sign is \(-\)), \(\text{Caps}\) is empty (i.e. \(\text{Caps} = \{\}\)).

The security capabilities provided by the resource in a ground-level policy rule should satisfy the requirements of the requester in order to the requested access be granted. For example, the requester may need the output data of a web service (the requested action) to be encrypted using a specific algorithm. This requires the resource to be an individual of WithEncryptedOutput concept based on its capabilities.

By representing the under-protection resource \(o\) with a nominal concept \(N_o\), and the action \(a\) with a nominal concept \(N_a\), we can define a ground-level policy rule in MA(DL)\(^2\) language as follows. (Note that a nominal is a special concept that has only one individual [46]. We assume that the name of a nominal of an individual is a unique name made based on the identity (name) of the individual.)

Definition 5.4 (Ground-level policy rule in MA(DL)\(^2\)): A ground-level policy rule for permitting an action \(a\) on an under-protection resource \(o\) (i.e. for \((+, u, d, cx, \text{Reqs}, \text{Caps})\) is specified as an MA(DL)\(^2\) formula with the schema of the form

\[
\begin{align*}
\text{cx} & \rightarrow (PE_{dS\downarrow dA} (\cap S_i N_o N_a) \land OB_{dS\downarrow dA} \text{Cap} (\cap C_i N_a))
\end{align*}
\]

where \(\text{cap}\) is a binary predicate that assigns the capabilities of a resource to the resource, and a rule for prohibiting the action (i.e. for \((-\), \(u, d, cx, \text{Reqs}, \{\}\)) is specified as a formula of the form

\[
\text{cx} \rightarrow IM_{dS\downarrow dA} (\cap S_i N_o N_a).\
\]
Note that stating the ground-level policy rules in MA(DL)² has many advantages, such as deriving ground-level policy rules for shared resources in cooperative domains using MA(DL)² proof theory (for composite authorities), simplifying the access control procedure using MA(DL)² inference service and uniforming security policy specification in conceptual and ground levels.

Example 5.5: Some samples of the ground-level policy rules for the SVO case study that are specified by company B (which is the real owner of resources) are as follows:

- Foreign programmers cannot have write access to the storage with 10 GB capacity.

\[
\text{IM}_{\text{adm}@d_1} \text{do}(\text{ForeignProg}, N_{R{\text{PG}}, 1})
\]

- Programmers can write their data on the storage with 1 GB capacity encrypted. Encryption of data is the capability of resource stg10.

\[
1\text{st Oct} 09 \text{To} 30\text{th Dec} 09
\rightarrow \text{PE}_{\text{adm}@d_1} \text{do}(\text{Programmer}, N_{R{\text{PG}}, 1}, N_{\text{write}})
\land \text{OB}_{\text{adm}@d_1} \text{cap}(\text{EncryptedMem}, N_{R{\text{PG}}, 1})
\]

6 Security policy administration and enforcement

Conflicts between policy rules is an important issue that might be handled to have a sound and reliable security system. This issue has a considerable effect on security policy base administration and also on policy enforcement. Hence, in the rest, at first we have a discussion on the conflict types and our approach on resolving the conflicts. Then we describe the administration operators and access control procedure in the proposed model.

6.1 Conflict types and resolution

Since primitive authorities are independent of each other, the conflicts between the policy rules of different authorities do not make any problem. Also note that, in cooperative management, existence of such conflicts does not make any problem in policy inference for composite authorities. Thus, conflicts might occur only in policy space of an authority.

The possible conflicts can be classified into

1. the intra-level conflicts, that is, between the policy rules of the same level (ground or conceptual level),
2. the inter-level conflicts, that is, between the policy rules of the ground level with the ones in the conceptual level.

In the rest, we just consider intra-level conflicts and resolve the inter-level ones by taking a special sequence for the steps of the access control procedure.

**Definition 6.1 (Persistent conflict):** Two policy rules are persistently in conflict, if they are in conflict in all circumstances. Formally, there is a ‘persistent conflict’ between a pair of policy rules \( p_1 = \alpha_1 \rightarrow \text{ds}_{\text{adm}@d_1} \text{do}(S_1, O_1, A_1) \) and \( p_2 = \alpha_2 \rightarrow \text{ds}_{\text{adm}@d_2} \text{do}(S_2, O_2, A_2) \), if:

1. \( \vdash \alpha_1 \leftrightarrow \alpha_2 \),
2. there exists \( S, O, A \) and \( d \) such that

\[
\text{TB} \cup \{\text{ds}_{\text{adm}@d_1} \text{do}(S_1, O_1, A_1)\} \vdash \text{ds}_{\text{adm}@d_2} \text{do}(S, O, A)
\]

and

\[
\text{TB} \cup \{\text{ds}_{\text{adm}@d_2} \text{do}(S_2, O_2, A_2)\} \vdash \text{ds}_{\text{adm}@d_2} \text{do}(S, O, A)
\]

3. \( \text{ds} \) and \( \text{ds}' \) are one of the possible conflicting pairs of normative statuses, that is, \( \langle \text{PE}, \text{IM} \rangle, \langle \text{OB}, \text{IM} \rangle \) or \( \langle \text{OB}, \text{GR} \rangle \).

For example, policy rules \( \neg \text{isWorkTime} \rightarrow \text{PE}_{\text{adm}@d_1} \text{do}(\text{Programmer}, \text{Memory}, \text{MemAct}) \) and \( \neg \text{isWorkTime} \rightarrow \text{IM}_{\text{adm}@d_2} \text{do}(\text{NetProg}, \text{VolatileMem}, \text{MemAct}) \) are persistently in conflict.

**Definition 6.2 (Potential conflict):** There is a potential conflict between two policy rules, if they are in conflict in some contextual conditions. Formally, there is a ‘potential conflict’ between a pair of policy rules \( p_1 = \alpha_1 \rightarrow \text{ds}_{\text{adm}@d_1} \text{do}(S_1, O_1, A_1) \) and \( p_2 = \alpha_2 \rightarrow \text{ds}_{\text{adm}@d_2} \text{do}(S_2, O_2, A_2) \), if \( \vdash \alpha_1 \leftrightarrow \alpha_2 \) and conditions (2) and (3) in Definition 6.1 are fulfilled.

For example two policy rules \( \neg \text{isWorkTime} \rightarrow \text{PE}_{\text{adm}@d_1} \text{do}(\text{Programmer}, \text{Memory}, \text{MemAct}) \) and \( \neg \text{isHoliday} \rightarrow \text{IM}_{\text{adm}@d_2} \text{do}(\text{NetProg}, \text{VolatileMem}, \text{MemAct}) \) are not always in conflict. However, when it is a working time in a working day, both policy rules are activated and a conflict occurs.

The time points where the aforementioned conflicts might occur are as follows:

- A persistent conflict might occur in adding a new policy rule to an SKB. More formally, it occurs when we want to add a new policy rule \( p_{\text{new}} = \alpha \rightarrow \text{ds}_{\text{adm}@d_1} \text{do}(S, O, A) \) to an SKB \( \mathcal{K} \), and we have \( \mathcal{K} \vdash \neg p_{\text{new}} \). In other words \( \mathcal{K} \cup \{p_{\text{new}}\} \vdash \bot \), which means \( p_{\text{new}} \) has conflict with some of the existing policy rules.

- A conflict might occur in access control time (by receiving an access request) and updating a contextual information. In this step, some potential conflicts might be realised. More precisely, if we update an SKB \( \mathcal{K} \) with the new contextual facts, the SKB becomes inconsistent, and thus \( \mathcal{K} \vdash \bot \).

We can easily prevent the occurrences of persistent conflicts in the operators used for adding a new policy (see Section 6.2). However, detecting and resolving potential conflicts should be performed dynamically in access control procedure. For this purpose, we construct a potential conflict graph (similar to
the approach we proposed in [47]) when adding policy rules to the SKB. We use this graph to detect the realised potential conflicts and resolve them.

Definition 6.3 (Potential conflict graph): A potential conflict graph is an edge-labelled directed graph such that each vertex represents a policy rule that is potentially in conflict with another policy rule, and each edge represents a potential conflict between a pair of policy rules \( p_1 = \alpha \rightarrow \text{ds}_{\text{nd}}(S_1, O_1, A_1) \) and \( p_2 = \alpha \rightarrow \text{ds}_{\text{nd}}(S_2, O_2, A_2) \), and has a label of the form \((\alpha_1 \land \alpha_2)\), OR. \( \alpha_1 \land \alpha_2 \) determines the contextual condition in which the potential conflict becomes realised, and OR determines overriding relation and is one of NO (negative obligation precedence) and PO (positive obligation precedence). OR is determined based on the second column of Table 1. NO and PO are used for conflict resolution and are defined for normative statuses (belong to TD) in Table 1.

Fig. 5a shows a sample of potential conflict graph.

Definition 6.4 (Realised conflict graph): A realised conflict graph is a subgraph of a potential conflict graph. It contains only the edges that the first element of their labels are equal to the determined policy rule. The realised conflict graph is a subgraph of a potential conflict graph. It is determined based on the second column of Table 1. NO and PO are used for conflict resolution and are defined for normative statuses (belong to TD) in Table 1.

Fig. 5b shows the realised conflict graph of the graph shown in Fig. 5a in a special contextual condition.

For dynamic resolution of realised conflicts, we just need to remove the policy rules on the tails of the edges that the second elements of their labels are equal to the determined conflict resolution strategy in the meta policy of the security domain (i.e. in ResSt in meta policy MP of a security domain SD). For example, if we have the PO strategy, the filled vertices (policy rules) in Fig. 5c are removed temporarily from the SKB for access decision.

6.2 Security policy base administration

Each authority can administrate his specified SPR in his/her security domain using the following administration operators:

---

Table 1 Normative conflict resolution based on the NO (negative obligation takes precedence) and PO (positive obligation takes precedence) strategies [2]

<table>
<thead>
<tr>
<th>Conflicting pair</th>
<th>Overriding relation label</th>
<th>NO strategy</th>
<th>PO strategy</th>
</tr>
</thead>
<tbody>
<tr>
<td>(PE, IM)</td>
<td>PE ( \to ) NO, IM, IM</td>
<td>IM</td>
<td>PE</td>
</tr>
<tr>
<td>(OB, IM)</td>
<td>OB ( \to ) NO, IM, IM</td>
<td>IM</td>
<td>OB</td>
</tr>
<tr>
<td>(OB, GR)</td>
<td>OB ( \to ) GR, GR</td>
<td>GR</td>
<td>OB</td>
</tr>
</tbody>
</table>

---

6.2.1 Policy revision operators: Add policy rules to the SKB, and update the potential conflict graph if it is required. These operators are as follows:

1. FPR: Adds a new policy rule by removing all the conflicting old policy rules. We have two kinds of FPR operators:

   - hFPR: harmonises the conflicting policy rules w.r.t. the new one. Harmonising a policy rule \( p_1 = \alpha \rightarrow \text{ds}_{\text{nd}}(S_1, O_1, A_1) \) regarding the new policy rule \( p_{\text{new}} = \alpha \rightarrow \text{ds}_{\text{nd}}(S_2, O_2, A_2) \) results in a policy rule \( p_1 = \alpha \rightarrow \text{ds}_{\text{nd}}(S_1 \land \neg S_2, O_1 \land \neg O_2, A_1 \land \neg A_2) \), which replaces \( p_1 \).

   - dFPR: deletes or removes the policy rules conflicting with the new one.

2. CPR: Adds a new policy rule only if it is consistent with the existing ones in the SKB.

6.2.2 Policy contraction operators: Remove policy rules from the SKB, and update the potential conflict graph if it is required.

- WPC: Removes an explicit policy rule existing in the SKB.

- SPC: Removes a policy rule and all the policy rules entails such a rule. Note that the determined policy rule might be an explicit rule existing in the SKB, or might be an implicit rule, which is inferred from the other ones.

6.3 Access control procedure

The access control procedure, which is used by the security agent of a security domain \( SD = (d, u, O, K, MP) \), has the following steps:

1. Request reception: PEP receives an access request in the form \((s, a, a_i, \text{Crd}_i, \text{Req})\) from a subject, where \( s \), \( a \), and \( a_i \) are subject (access requester), resource (object) and requested action, respectively. \( \text{Crd}_i \) is the set of credentials that \( s \) presents, and \( \text{Req} \) is the set of requirements that \( s \) requires to be satisfied by the capabilities of the requested resource.

2. Request validation

   (a) PEP checks whether \( a \) is registered in the security domain, and \( a_i \) is an eligible action on \( a \) (by checking the description of the resource’s web service specified in OWL-S).

   (b) PEP checks the validity of the received credentials using credential verifier and SOA, and sends the validated credentials to PDP.
3. SKB initialisation

(a) PDP generates a set of assertions based on the received credentials and inserts them to $AB$ of SKB $K$. The assertions are of the form $C(s, r)$, where $C \subseteq Crd$. For example, if it receives a credential that says ‘Ali is a student’ and concept ‘Student’ is defined in $TB$ of SKB $K$, we add the assertion ‘Student(Ali)’ to $AB$.

(b) By the request of PDP, SKB $K$ updates itself with the last changes in contextual information (using the context handler component).

(c) PDP derives the realised conflict graph (from the potential conflict graph constructed during the specification of the SPR), performs conflict resolution based on the determined resolution strategy, and removes the selected conflicting policy rules from the SKB.

4. Ground-level access decision making: PDP makes initial access decision based on the ground-level policy rules of the requested resource in the following steps.

(a) If $K \vdash PE_{\text{req} \in Req}(\cap_{\text{crd} \in Crd} N_{s_i}, N_{a_i})$, the access is permitted and the access control procedure terminates after executing the last step (SKB purging).

(b) If $K \vdash IM_{\text{req} \in Req}(\cap_{\text{crd} \in Crd} N_{s_i}, N_{a_i})$, the access is denied and the access control procedure terminates after executing the last step (SKB purging).

(c) If $K \vdash PE_{\text{req} \in Req}(\cap_{\text{crd} \in Crd} N_{s_i}, N_{a_i}) \land \neg OB_{\text{req} \in Req}(\cap_{\text{req} \in Req} N_{s_i})$, it means the requester is permissible to take the requested access; however, all of its requirements cannot be satisfied. In this case, the requester can be negotiated and then it can cancel the request, and the procedure terminates after executing the last step (SKB purging), or it can enquire for the available capabilities and then make its final decision (to change the requirements in the request or cancel the request).

5. Conceptual-level access decision making: PDP makes its final access decision based on the conceptual-level policy rules in the following steps:

(a) PDP maps $s_i$, $a_i$ to appropriate concepts in the ontologies by the following equations. Note that these concepts are different from the nominals for the individuals. PDP leverages its inference engine over the SKB to infer the most specific concepts ($msc$) (Formally, $msc$ of an individual $a$ of type $\sigma$ is a set of incomparable
concepts $C_1, \ldots, C_n$ of type $\sigma$ satisfying $\boxplus B \vdash C_i(a)$ $(1 \leq i \leq n)$ and if there exist a concept $D$ such that $\boxplus B \vdash D(a)$, then there exist a $C_i$ in the above set such that $\boxplus B \vdash C_i \subseteq D$) of each of the individuals $i$, $o$, and $a$, for this purpose.

$C_i = \bigcap_{S_i \in \text{Inc}(i)} S_i \quad C_o = \bigcap_{O_i \in \text{Inc}(o)} O_i \quad C_a = \bigcap_{A_i \in \text{Inc}(a)} A_i$

(b) If $\mathcal{K} \vdash \text{PE}_{\text{sgr}, \text{ad}}(C_o, C_i, C_a)$, the request is granted.

(c) If $\mathcal{K} \vdash \text{IM}_{\text{sgr}, \text{ad}}(C_o, C_i, C_a)$, the request is denied.

(d) If none of the above inferences are made and default takes precedence approach is employed for inter-level conflict resolution.

6. SKB purging: After completing the decision-making process, PDP

(a) removes the current contextual facts from the SKB,

(b) removes the assertions related to the requester subject from $\boxplus B$ of the SKB, and

(c) returns back the policy rules that are removed for realised potential conflicts to the SKB.

In the above procedure, the inter-level conflicts do not make any problem, because the policy rules of the two levels are considered in two separated steps. In fact, negative takes precedence approach is employed for inter-level conflict resolution.

Example 6.5: As an example, suppose the security agent of domain $d_X$ receives an access request $\langle\text{Ali}, \text{SysProg}, \text{Native}, \{\text{SysProg}, \text{Native}, \{\text{EncryptedStorage}\}\} \rangle$ 10 am on 6th October 2009. It can infer the following formula from the ground-level policy rules in the SKB $\mathcal{K}_{\mathcal{X}}$:

$\text{PE}_{\text{adm}, \text{adm2}}(\text{do}(\text{SysProg} \cap \text{Native}, N_{\text{write}}))$ \land $\text{OB}_{\text{adm}, \text{adm2}}(\text{do}(\text{cap}(\text{EncryptedStorage}, N_{\text{write}}))$

and can infer the following formula from the conceptual-level policy rules

$\text{PE}_{\text{adm}, \text{adm2}}(\text{do}(\text{SysProg} \cap \text{Native}, \text{PermanentMem}, \text{MemAct}))$

Hence, the access request is granted.

7 Implementation and evaluation

To show the applicability of the proposed model and framework, and also to evaluate them in practice, a simple version of a security agent has been implemented as a prototype. For this purpose, considerable efforts devoted to the development of the MA(DL)$^2$ inference engine, which is introduced in the following. Also, a brief introduction to the implemented prototype of a security agent and the results of the evaluation of the model and implemented system are presented in the rest of this section.

7.1 MA(DL)$^2$ inference engine

To automate reasoning based on MA(DL)$^2$, an analytical tableau system is developed. The expansion rules of the developed tableau system are implemented in prolog, which is available at http://ce.sharif.edu/~m_amini/madl2/reasoner/madl2_v2_07.pl under GNU GPL license.

7.2 Security agent prototype

A prototype of a security agent based on the architecture shown in Fig. 3 is implemented using Google web toolkit-GWT (available at: http://code.google.com/webtoolkit) on Java platform. In the current implementation, Jena (available at: http://jena.sourceforge.net) framework is used for inference over $\boxplus B$ (for subsumption inference) and $\boxplus B$ (for instance checking inference). The MA(DL)$^2$ inference engine (implemented in prolog) is employed by the JIP (available at: http://jipprof.sourceforge.net) profiling tool for policy inference in PDP.

The current implemented system can read ground-level policy rules, which are specified in OWL-S in their related web services. It uses Protegé (available at: http://protege.stanford.edu) as the PAP’s UI for conceptual-level policy specification, and can receive an access request and decide based on the aforementioned procedure.

The source code of the system is available at: http://ce.sharif.edu/~m_amini/madl2/prototype under GNU GPL license.

7.3 Evaluation and experimental results

To evaluate the proposed model, in Table 2, it is compared with other famous models proposed for SAEs. Some parts of this table are taken from [2]. The following points in evaluating the proposed model and filling this table are taken into account:

‘(R1, R4, R7, R8)’ Regarding the specification of the proposed model, it is clear that how the model satisfies the requirements R1 (policy specification in both conceptual and ground levels), R4 (supporting policy propagation and inference based on the ontologies of subjects, objects and actions), R7 (supporting contextual conditions in policy rule specification), and R8 (conflicting rules detection and resolution).

‘(R2)’ To satisfy authorisation requirement R2 (distributivity of policy specification and scalability of the model), the model follows the MSD structure, where the environment is divided into some security domains and policy
specification done in the distributed manner. In each security domain, we may have some authorities who are authorised to specify the security policies, and we can use the cooperative security management approach to policy inference over the distributed policy rules. Note that conceptual-level policy rules of each domain is maintained centrally in the SKB of the domain. However, ground-level policy rules of each resource is stored in service provider of the resource (as it is mentioned in canonical model of resources in Section 3.4).

(R3) Supporting different deontic (normative) statuses (defined in set $\mathcal{DS}$) in the MA(DL)$^2$ language for policy rule specification enables the model to support the specification of positive and negative authorisation as well as positive and negative obligations.

(R5) The model is based on MA(DL)$^2$ logic which is placed in logic, layer in the semantic layer cake (see Fig. 3 in [2]) and it has a clear formal semantics presented in [13].

(R6) Each concept in the subjects ontology is a representative of a set of principles (subjects) that have some common attributes. Thus, possibility of the model in specification of policy rules over the subject concepts categorises it in the class of the attribute-based (credential-based) authorisation models. Refer to the description of stage 3(a) in the access control procedure to see how the provided attributes of the requester (by some credentials) used for access decision making.

(R9) Each role and group in our proposed model can be defined as a concept in subjects ontology. Thus, roles and groups can be defined easily in our model. Note that the hierarchy of roles can be defined, too. If a role $R_1$ (e.g. manager) is a subrole of $R_2$ (e.g. employee), in our model corresponding concept $R_1$ should be defined as a subsumer of corresponding concept $R_2$ in subjects ontology (i.e. $R_1 \sqsubseteq R_2$). By delegative cooperative management (provided by MA(DL)$^2$ [12] in our model), the administrative delegation is also possible. For more details see [25].

To evaluate the current implementation of the proposed framework, some test cases were run on a system with 1.66 GHz Core(TM)2 Duo CPU and 2 GB of RAM.

In Figs. 6 and 7, the number of policy rules (in both conceptual and ground levels) has a near to linear relation with the number of the concepts. As depicted in the diagrams, by increasing the number of the concepts and policy rules, the response time increases in a very slight exponential manner, near to linear. This means that the agent decides and responds in an acceptable time, in practice. In Fig. 8, the response time with different number of concepts is shown. In this diagram, one conceptual-level policy rule beside a ground-level policy rule exist in the SKB. In this diagram, we just consider the worst case.

Comparing the three diagrams show that the size of $SB$ in the SKB has more effect than the size of $TB$ on the response time of the agent.

### 7.4 Response time optimisation

We can propose different kinds of access control enforcement mechanisms based on the proposed model and framework in this paper. Since logical inference is the time-consuming
process in the systems developed based on the proposed model, the techniques that might be leveraged in proposing the mechanisms with the low response time are as follows:

• The MA(DL)\(^2\) inference engine is implemented using analytical tableaux approach. The tableaux systems have clear opportunities for parallelisation, which significantly decreases the inference time \([48, 49]\) (The different branches of the tableaux proof tree can be processed in parallel.).

• One of the major techniques in logical systems is ‘client proof carrying’ approach. In this approach, the requester (subject) provides some of the proofs that the security agent requires (e.g. most specific concepts or the authorisation rules that shows the requester can access the requested resource), and the security agent just needs to verify the proof (which has a very low cost). In fact, in this technique, requesters sustain part of the inference cost.

• In the environments with the small SKB (and also rarely change in the SKB), we can infer all the possible subsumption relations in TB and infer the required implicit authorisation rules in SB because of the semantic relationships defined (inferred) in TB. Note that since the number of inferable c-formulae in SB, we should infer the rules applicable to the possible access requests.

Figure 6 Response time of granting an access against the number of concepts, whereas policy rules increase relative to the number of concepts

Figure 7 Response time of denying an access against the number of concepts, whereas policy rules increase relative to the number of concepts

Figure 8 Response time of denying an access against the number of concepts, whereas there is only a single policy rule
Note that since the security agent act as a proxy server to access the resources in the domain, the computational cost is on the security agent, which is always a powerful machine. Thus, the mobile devices with limited resources do not have any problem to access the other resources or being accessed by others.

8 Case study – distributed semantic digital library

In Sharif University of Technology, each department had a set of digital scientific resources in its local digital library. To make all the scientific resources accessible to different users and having inter-university collaboration for science evolution, the deans of the departments decided to share their resources with the central library of the university. The central library has a set of resources other than the resources shared by the departments, and also has access to the resources of some scientific institutes (e.g. IET, IEEE and ScienceDirect). Fig. 9 shows some of the security domains of this distributed digital library. In this case, we take two departments (i.e. Department of Computer Engineering and Department of Mathematical Sciences) and one scientific institute (i.e. IET).

Each department has its SPR for its shared scientific resources other than the policies specified by the central library. Each resource itself may has some policies regarding the author’s or publisher’s copyrights (good examples of such policies can be found at http://book.goolge.com). Some of the departments (e.g. Department of Mathematical Sciences) delegate the security management of the shared resources to the central library. Thus, central library specifies the security policies on behalf them to access the shared resources. However some of the others (e.g. Department of Computer Engineering) cooperatively with the central library manage their shared digital resources. The shared subdomain between each foreign scientific institute and the central library is cooperatively managed based on the disjunctive management style. This means, everybody can access these resources if the rules of the institute or the rules of the central library allow.

8.1 Semantic technology and security management

Owing to the high volume of digital scientific resources and different types of the users access to them, semantic technology is used to describe the resources, the users and the action types to access the resources. Figs. 10–12 show parts of the ontologies of the users (subjects), the resources (objects), and the actions, respectively. Note that the ontologies shown in these figures are not complete and many details (e.g. the attributes that each concept has and many other concepts) are eliminated to decrease the complexity of the case study.

Following the formal specification of the authorisation model in Section 4, it is clear that the definition of the

![Figure 9 Security domains in distributed semantic digital library](image-url)
elements of FDS is not difficult in this case study (e.g. $\hat{O}$ is the set of all resources in the distributed digital library).

Note that the authority of each security domain is denoted by $\text{auth}_X$, where $X$ is the abbreviation of the domain’s name. The shared domain between domains $X$ and $Y$ is also denoted by $X \setminus Y$.

For each security domain $X$ in this case study, $\text{SKB}_X = (\mathcal{T}_X, \mathcal{A}_X, \mathcal{S}_X)$, where $\mathcal{T}_X$ contains the descriptions of the ontologies shown graphically in Figs. 10–12.

The assertions introducing the resources registered in the domain $X$ are stored in $\mathcal{A}_X$. For example for a conference paper $p_1$, we have the assertion ConferencePaper($p_1$) in $\mathcal{A}_X$. For each action concept, we define an instance with the same name (e.g. ‘edit’ for the action concept ‘Edit’). Thus, the assertions about the actions are defined as follows:

$\text{Edit}(\text{edit}), \text{Upload}(\text{upload}), \text{Delete}(\text{delete}),$
$\text{Read}(\text{read}), \text{Download}(\text{download}),$
$\text{OnlineRead}(\text{onlineread}), \text{OnlinePreview}(\text{onlinepreview}),$
$\text{Search}(\text{search}), \text{Print}(\text{print})$
Note that in this way, for example ‘read’ is an instance of ‘Read’ and also is an instance of ‘Edit’, which means ‘read’ is a kind of edit action; where we read, but do not write.

The assertions about the requesters (subject instances) are inserted dynamically during the access control procedure as described in Section 6.3.

8.2 Security policy rules

The samples of concept-level SPR specified in each security domain in this case study (stored in $B_X$) are as follows:

**Department of Computer Engineering (CE):** The policy rules specified in domain CE and its shared subdomains are as follows:

- The students of CE can have read access to the CE’s resources.

$$T \rightarrow PE_{(auth_{CE} \cap CE)} do(\exists AffiliatedTo \bullet \{\text{compEng}, T_{Object}, \text{Read}\})$$

- Visiting professors and students in CE can access to CE’s resources except theses.

$$T \rightarrow PE_{(auth_{CE} \cap CE)} do((\exists VisitingProfessor \lor ExtFacultyMember) \cap Visitor, T_{Object} \setminus \text{Thesis, Read})$$

- The shared domain CE $\parallel CL$ is defined as a subdomain of CE, that is, CE $\parallel CL \preceq CE$.

**Central library (CL):** The policy rules specified in domain CL and its related shared subdomains are as follows:

- Graduate-level students (i.e. MSc and PhD students) and Sharif faculty members can obtain read access to the resources of external scientific institutes including IET.

$$T \rightarrow PE_{(auth_{CL} \cap CL)} do(MScStudent \lor PhDStudent \lor SharifFacultyMember, SciInstResource, Read)$$

- The members of Ghadir plan and faculty members of other universities can only read the CE’s theses online.

$$T \rightarrow PE_{(auth_{CL} \cap CE \parallel CL)} do(GhadirPlanMember \lor ExtFacultyMember, Thesis, OnlineRead)$$

- PhD students of Sharif can upload their theses and technical reports to the digital library; however, MSc and BSc theses should be added by the library’s administrator.

$$T \rightarrow PE_{(auth_{CL} \cap CL)} do(PhDStudent, Thesis \lor TechnicalReport, Upload)$$

$$T \rightarrow IM_{(auth_{CL} \cap CL)} do(Student \lor PhDStudent, Thesis, Upload)$$

$$T \rightarrow PE_{(auth_{CL} \cap CL)} do(LibAdmin, Thesis, Edit)$$

- Authority of domain Department of Mathematical Sciences (MS) should grant read right to all Sharif members as well as the external members to access all registered resources in domain MS. Similarly, full access to the administrators of central library.

$$T \rightarrow PE_{(auth_{CL} \cap MS)} do(SharifMember \lor ExtMember, T_{Object}, \text{Read})$$

$$T \rightarrow PE_{(auth_{CL} \cap MS)} do(LibAdmin, T_{Object}, T_{Action})$$

**IET institute:** The policy rules specified in domain IET and its shared subdomains are as follows:

- The users connected with Sharif’s IP can obtain read access to the shared resources with Sharif during year 2010 (that they have contract).

$$Year\sim 2010 \rightarrow PE_{(auth_{IET} \cap IET) \cap CL} do(SharifIPholder, T_{Object}, Read)$$

- The free subscribed users of IET digital library (with username and password) freely have read access to the papers of conferences and workshops in the case that the traffic is less than 120 M.

$$Traffic\leq 120M \rightarrow PE_{(auth_{IET} \cap IET) \cap CL} do(IETsubscriber, ConferencePaper \lor WorkshopPaper, Read)$$

- The shared domain IET $\parallel CL$ is defined as a subdomain of IET, that is, IET $\parallel CL \preceq IET$.

**Ground-level policy rules:** The samples of ground-level policy rules in this case study are as follows:

- The book entitles ‘Computer security basics’ with ID ‘qa76.9’ cannot be downloaded by non-library administrators.

$$T \rightarrow IM_{(auth_{CL} \cap CE \parallel CL)} do(T_{Subject} \setminus LibAdmin, N_{qa76.9}, Download)$$

- The content of the above book (with ID ‘qa76.9’) can be searched by all users and if they require the results of the
search can be encrypted.

\[
\begin{align*}
T \rightarrow & \mathbb{PE}_{(\text{auth}_{\text{CE} @ \text{CE}} @ \text{CL})} \text{do}(\text{Search}, N_{\text{qa76.9}}) \\
& \text{\textbf{\&} } \mathbb{OB}_{(\text{auth}_{\text{CE} @ \text{CE}} @ \text{CL})} \text{cap}(\text{Encrypted}, N_{\text{qa76.9}})
\end{align*}
\]

- The users who are not the faculty members of Sharif University cannot see the aerial map with ID ‘m103’.

\[
T \rightarrow \mathbb{IM}_{(\text{auth}_{\text{CL} @ \text{CL}})} \\
\text{do}(\text{OnlinePreview}, N_{\text{m103}})
\]

### 8.3 Experimental results

To evaluate the system in this case study, we can measure the response time of the access requests submitted to the system. To this aim, a small sample distributed digital library was established based on the case described earlier in this section. The deployment diagram of the case is shown in Fig. 13. Each system in this diagram, plays two roles; the role of the ‘security agent’ of a domain (and acts as a proxy server of the registered resources in the domain), and the role of the ‘resource service provider’ based on the model depicted in Fig. 4.

Table 3 shows the average response time of different kinds of access requests submitted to the security agents in this case study. Two kinds of requests are taken into account in this evaluation: the granted requests and the denied requests. Also the requests submitted to access the local resources of the domains are separated from the access requests to the shared resources on which the cooperative security management is applied. The last row in the table shows the average response time of the access control system in the case that we have the combination of the requests that are granted or denied. The ratio of the granted requests to the denied ones is taken 11:2. The ratio is obtained from the logs of the different real access control systems.

### 9 Conclusion

Considering the semantic relationships beside the distribution of SAEs makes the authorisation and access control a complicated problem in them. In this paper, we showed how we can use the MA(DL)² logic for modelling security in SAEs, specify SPR in both conceptual and ground levels, and also perform access control based on this logical framework.
MA(DL)² provides more expressive power than what is employed in this paper; however, the framework, authorisation model, conflict resolution, detailed access control and SKB administration, mentioned in this paper, provide a basis for solving the fundamental problems exist in other models and frameworks to satisfy the authorisation requirements of SAEs.

10 Acknowledgments
The authors cordially thank Mr Mousa Amir Ehsan and Mrs Fathieh Faghih for their significant help in design, implementation and evaluation of the security agent prototype. It is also a pleasure to acknowledge Iran Telecommunication Research Center (ITRC) for partial support of this research.

11 References


[38] WU Z., CHEN H.: ‘Semantic grid: model, methodology, and applications’; in ‘Advanced topics in science and technology in China’ (Springer-Verlag, 2008)


