# A Semantic-based Access Control Mechanism Using Semantic Technologies*

Moussa A. Ehsan
Sharif Network Security Center (NSC)
Department of Computer Engineering
Sharif University of Technology
Tehran, Iran
a_ehsan@ce.sharif.edu

Morteza Amini
Sharif Network Security Center (NSC)
Department of Computer Engineering
Sharif University of Technology
Tehran, Iran
m_amini@ce.sharif.edu

Rasool Jalili
Sharif Network Security Center (NSC)
Department of Computer Engineering
Sharif University of Technology
Tehran, Iran
jalili@sharif.edu

## ABSTRACT

In order to overcome the shortcomings of the recent frameworks and mechanisms for semantic-based access control, this paper presents a semantic-based, context-aware, and multi-domain enabled framework implementing a semantic-based access control mechanism for Semantic Web. The access control framework is based on the MA(DL)$^2$ model, which takes the semantic relationships among different entities into account. The framework handles the Semantic Web context by classifying and representing it through an ontology. Considering the MA(DL)$^2$ model, the framework assumes Semantic Web having some overlapped domains, which each contains an authority and a security agent. As a domain authority responsibility is to specify the domain policies, its agent is to enforce them. The mechanism is designed using the semantic technologies, which make it fully consistent with the environment. The paper clarifies the usability of the designed mechanism through some examples of an elections system case study.

## Categories and Subject Descriptors

K.6.5 [**Computing Milieux**]: MANAGEMENT OF COMPUTING AND INFORMATION SYSTEMS—*Security and Protection*

## General Terms

Security

## Keywords

Semantic-based Access Control, Semantic Technologies, MA(DL)$^2$, Security Mechanisms

*This paper is partially supported by Iran Telecommunication Research Center (ITRC).


*SIN'09,* October 6–10, 2009, North Cyprus, Turkey.


## 1. INTRODUCTION

As Semantic Web evolves, the security concerns increase. These concerns are because of the users requirement to control access to their resources. Meanwhile, the distributed nature of Semantic Web and its shared resource model, leads to more doubt on how people should trust on this environment. Access Control is an approach to control who accesses the resources in such an environment. Accordingly:

1. An access control system is to allow only privileged users to access a resource.

2. Each authorized user should be able to access a resource without any problem.

Differences between the access control models are resulted from different security requirements, or in other words the security policies they abstract. In semantic-based access control models, access control is done by considering semantic relationships and users attributes. This means that the authorized users are not explicitly defined and there should be some engine to decide about the access request using semantic relationships and users attributes. Such models are abstractions of access policies. Enforcing these policies needs some mechanisms to be designed.

In this paper, a designed access control mechanism is presented to enforce the policies specified by one or more authorities in different domains, based on the MA(DL)$^2$ model in Semantic Web. MA(DL)$^2$ is a semantic-based context-aware multi-domain access control model. The model is based on MA(DL)$^2$ logic, which is a combination of a type of Deontic Logic and Description Logic. In MA(DL)$^2$, the distributed environment is assumed as some overlapped security domains. We employ the model for Semantic Web and design an access control mechanism to enforce the policies specified by the domains authorities. Using the semantic features makes it fully compatible with the environment. The mechanism is implemented within a security agent framework for a typical elections system application.

In the remainder of this paper, first the related work is surveyed. In section 3, the MA(DL)$^2$ model is described shortly. Section 4 discusses how semantic-based access control is done in Semantic Web. A case study is illustrated to clarify the designed mechanism in section 5. How the mechanism employs semantic technologies, is discussed in section 6. Section 7 compares the used access control trend with other related works. Last section concludes the paper.

## 2. RELATED WORKS

Most of access control models presented in recent years, are based on the relationships between the entities and available instances in the decentralized environment. Such relationships are usually represented by ontologies and developed with some popular languages such as OWL and RDF. Some samples of these models are SBAC (Javanmardi *et al.*) [9], SBAC(Naumenko *et al.*) [16], and SAC (Yague *et al.*) [21].

The SBAC Model [9, 10], a semantic-based access control for Semantic Web, was proposed in 2006. The model uses ontologies as its basis and attempts to reduce semantic relations into the subsumption relations and use them in the access control process. In 2008, the model was extended to support temporal and history-based conditions[19, 18, 20]. The access control model has been evolved and logic is used as the core of inference [3]. A logic-based language named MA(DL)$^2$ was introduced and used for specifying security policies and reasoning over the semantic information[2].

Naumenko *et al.* proposed another semantic-based access control model in 2006 [16, 17]. The model uses some ontologies with different granularities for employing semantic relations in access control and SWRL[1] for expressing provisions. Yague *et al.* proposed another semantic-aware access control model named SAC [22, 21] based on their previously proposed security policy rule language, SPL [23], in 2003. Its main contribution was the independence of access control to the location of the resources and not requiring clients identification.

Security policy language, Rei [13], was proposed in 2003 by Kagal *et al.*. Some semantic relations such as privilege delegation are enabled in this language. They proposed a semantic access control framework named Rein [11, 12] based on Rei in 2005 and 2006. One of its most important capabilities is that defining security policies is enabled in any policy language, of which an inference engine exists. XACML is another policy language proposed by Moses in OASIS in 2003 and 2005[14, 15]. OASIS proposed a framework, in which the language can be employed too. Not employing semantic relations, is the main drawback of this language and framework. Some attempts such as Damiani's *et al.*[5] in 2004 were done to employ semantic relations for the policy language but no important success was achieved.

## 3. THE MA(DL)$^2$ ACCESS CONTROL MODEL

The logical foundation of semantic-aware environments (based on description logic) as well as the benefits a logical system offers to satisfy important security requirements in semantic-aware environments motivates us to use such a logic-based security model. In the selected model a logical language (named MA(DL)$^2$) [2] is employed for specification and inference of security policies[2].

MA(DL)$^2$ is the combination of Deontic Logic and Description Logic that enables administrators of different security domains to specify their security policies in terms of deontic statuses: obligations, prohibitions, permissions, and waives. Using this logic, enables us to take the impact of subsumption relations between the classes of entities on propagation of security policies into account. The subsumption relation is the key relation between entity types (concepts) in a semantic-aware environment that construct a special hierarchy on them.

This security model is for a semantic-aware computational environment and contains the following components:

- logical structure of the system participants, *i.e.*, subjects, objects, and possible actions
- logical structured collection of security policies
- sets of inference rules

The system can use these to conduct automatic reasoning for conceptual level security control on the semantically structured collection of data that are specified as an ontology. In this model, the environment is divided into some security domains. Each security domain has a security manager, henceforth called *authority*. As is shown as a sample in Figure 1, in each domain a set of resources (objects) are protected based on its authority policies. The duty of an authority is composing the security policies of the domain. It is worthwhile to note that authorities domains may get overlapped with each other. In this way, the owner or creator of each resource (object) can register it in one or more domains and rely on their authorities policies to secure his/her resource. Details about the model and its security elements are discussed in [3].

Based on the aforementioned computational model of the environments and its security elements, the formal description of the model components are represented as follows.

**DEFINITION 3.1** Normative based Security Model Data Set A data set of the model is a 5-tuple $(FDS, SPB, MSP, OPR, KB)$. Each component of the data set is as follows.

**FDS:** FDS is the fundamental data set of the model that contains ontologies of subjects, objects , and possible actions of the under protection system. FDS is a 4-tuple $(ONT, DS, CX, DOM)$ in which each component is defined as follows:

- $ONT = (ONT_S, ONT_O, ONT_A)$, an ontology base that includes ontology of subjects ($ONT_S$), objects ($ONT_O$), and actions ($ONT_A$), and two special concepts denoted by $\top$ (universal concept) and $\bot$ (bottom concept).
- $DS = \{OB, PE, IM, GR\}$ is a set of deontic statuses that respectively illustrate Obligation, Permission, Impermission, Gratuitousness.
- $CX = \{cx_0, cx_1, ..., cx_n\}$ is a set of context propositions that are used to specify dynamic security policies based on the context.
- $DOM = (AU, AO, AX)$, represents domains of authorities in a semantic-aware environment. $AU = \{u_0, u_1, ..., u_k\}$ is a set of authorities that establish security policies in their specified domains. $AO$ is a function that maps each authority to a set of objects in his/her domain. $AX$ is another function that assigns to each authority a set of local context propositions.

**SPB:** A Security Policy Base stores a set of security policy rules. Security policies, including *authorization policies* as well as *obligation policies*, capture the security requirements of an organization. In other words, they separate authorized states from unauthorized ones. In this model security policy rules are specified based on the language of MA(DL)$^2$ logic [2].

**MSP:** The policies about security policies are described as meta security policies. MSP is a 2-tuple $(ResSt, DefSt)$ in which $ResSt$ is a resolution policy and $DefSt$ is a default status in the system.

**OPR:** OPR is the collection of administrative operations required to encode security policies and enforce them in the system. It

[1] http://www.w3.org/Submission/SWRL/

[2] The paper is based on version 2008 of MA(DL)$^2$. The logic is now in progress to be enhanced.

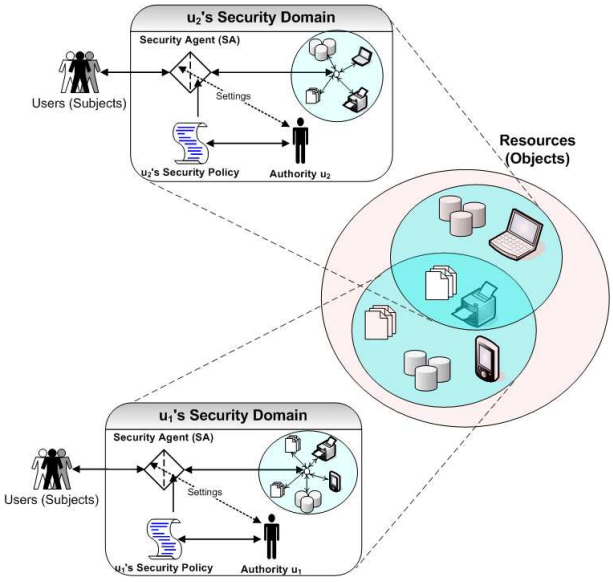


**Figure 1: The proposed definition for Semantic Web.**

is a set of three elements *AddPolicyRule* , *RemovePolicyRule*, and *AccessDecision*.

**KB:** A knowledge base of the system that is a union of subsumption relations (*SUB*), context propositions (*CX*), security policy rules (*SPB*), and conflict resolution rules (*RR*). Policy inference in the model is done based on the facts stored in the knowledge base *KB*.

# 4. APPLYING SEMANTIC-BASED ACCESS CONTROL IN SEMANTIC WEB

In order to control accesses based on MA(DL)$^2$ semantic access control in a decentralized manner, we divide Semantic Web into some security domains. In each domain, a security authority and a security agent exists. The domain authority is to specify the security policies in its domain; and the security agent task is to enforce the specified policies and control the accesses to the registered resources in the domain. In this new definition of Semantic Web, we assume that the resources owner registers them in one or some other security domains and only authorized users may have access to them. The proposed definition for Semantic Web is shown in Figure 1.

The resources may be annotated by the authority. The annotation contains some information about the security requirements and capabilities that the annotated resource has, and therefore, should be satisfied [7]. The access control process is done by reasoning over different information of the entities, which are involved in the access request. These entities include the requester, the resource, and other environmental (*i.e.* contextual) entities. MA(DL)$^2$ was proposed as a logic-based policy language[2], which can be used to specify and infer conceptual-level security policies in semantic-aware environments such as Semantic Web. An ontology was suggested to represent the policies based on this language [7]. In addition, another ontology was presenteed to classify the contextual information of Semantic Web environment, in [8]. In the next section, we describe the basis of the security agent framework.

## 4.1 The Security Agent

The framework of our-proposed security agent is shown in Figure 2. The framework is designed compatible with the two well-known standards, XACML [15] and ITU-T [1]. The external entities that are related to the framework are:

- Subjects (users, or requesters): A subject is a user or its agent, who wants to access some resources. The subject provides some of its security capabilities and requirements.
- The environment: This is the source of the contextual information.
- The security authority: The authority of a security domain is to specify the security policies and define the security requirements and capabilities of a resource via PAP, which is an interface to the agent framework.
- Source Of Authority(SOA): SOA validates the certificates provided by a subject to the security agent. In this framework, this external system is a part of the trust infrastructure SPKI/SDSI [4].

The internal components of a security agent are described as follows:

- Policy Administration Point(PAP): This is an interface for the security authority to specify security policies, security requirements and capabilities, and assign the meta data.
- Policy Decision Point(PDP): PDP decides about a requested access using its internal unit, Inference Engine. The decision is made through reasoning over the information of different entities such as subject, object, and other contextual information, which are stored in different ontologies. Inference Engine makes two important inferences: first, checking the credentials whether adequate certificates are provided, and, second, deciding about an access request.
- Policy Enforcement Point(PEP): This component is responsible for controlling accesses to a resource. By receiving an access request, PEP asks KB for all essential information, and redirects them to PDP for decision making, and finally, enforces the decision made by PDP on the request. PEP other task is to contact with Credential Verifier to verify the credentials.
- Context Handler: The necessary contextual information is gathered by this component. This information is stored in a specific format inside an OWL file after acquisition.
- Credential Verifier: This component verifies subjects prepared credentials by communicating with SOA. In our proposed framework, verifying credentials are based on SPKI/SDSI infrastructure [4].
- Ontology Manager: This module manages the base ontologies, and consists of Ontology Base, Ontology Parser, and Ontology Encoder. Some ontologies such as Subjects Ontology, Objects Ontology, and Actions Ontology are stored in Ontology Base. These ontologies are defined in different schemas and are in types of respectively madl2:SubjectEntity, madl2:ObjectEntity, and madl2:ActionEntity [7]. Ontology Parser is responsible for extracting subsumption relations needed for Inference Engine and storing them in the respective ontologies located in KB. The extraction is done only once per changes in the three base ontologies. This is done due to the efficiency of the inference.

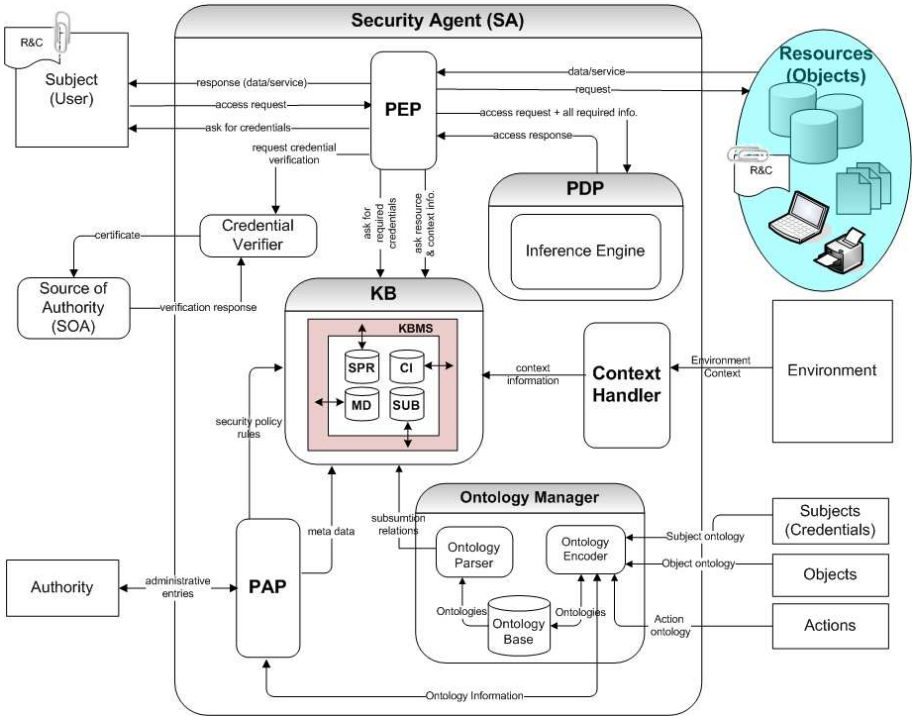


**Figure 2: The security agent framework.**

- Knowledge Base(KB): This module is a set of repositories of all information needed by Inference Engine to decide for an access request. All needed information is updated in KB by KBMS and used by Inference Engine. KB contains the following repositories which are in OWL format:

  - Security Policy Rules base(SPR): This repository contains the specified security policy rules, which are defined by the security authority. The rules are represented in OWL using madl2 ontology [7]. This repository is used to extract the required credentials and contextual information to decide about an access request.
  - Meta Data base(MD): The meta data needed for implementing the framework is stored in this repository. This information includes information about conflict resolution and the ontologies locations.
  - Subsumption base(SUB): The subsumption relations extracted from the Subject, Object and Action ontologies are stored in this base.
  - Context Information base(CI): CI contains the contextual information gathered from the environment. This repository is to maintain the history of the contextual information. Contextual information are represented in OWL using NSCContext ontology [8]. This ontology classifies the Semantic Web context from the access control point of view.

The process of access control is described in the next section through a case study.

# 5. CASE STUDY

In order to exemplify the details of the semantic access control mechanism, we describe a case study on an Elections System. In this section, first the elections system environment and its assumptions are demonstrated and then an access control scenario to show how our proposed framework performs access control is shown.

## 5.1 Elections System

In the following case study, a distributed elections system (ES) is assumed in which three elections subsystems exist: Presidential Elections Subsystem (PES), parLiament Elections Subsystem (LES), and Mayor Elections Subsystem (MES). These subsystems are the resources which were defined and registered by an authority. The users (Subjects) can access these resources using any gate connected to Semantic Web. In this case study, we assume that the domains are created considering regional matters, which each region is a village or a city. People living in each region, have their own privilege to access different subsystems. Each region contains a security authority that is responsible to manage and control the elections in the region. For simplicity, we assume only one authority, *i.e.* Interior Minister, manages all the regions and their subsystems ES.

Some actions such as Voting, Counting the Votes, Candidate Registration, Viewing Results, exist which can be acted on the resources. These actions compose an ontology named Action Ontology. All the actions are single-individual concepts. To access the resources, the users should provide some credentials. These credentials, which are a collection of attributes that a user has, compose the Subject Ontology which is depicted in Figure 3 and will be known as subjects. The users requiring to access the resources should provide credentials which match such subjects attributes. These credentials should be based on the policies defined by a se-

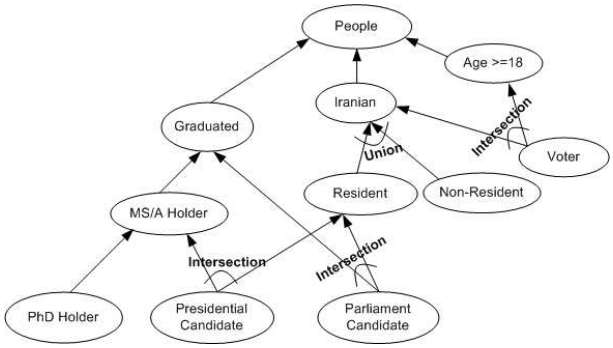


**Figure 3: The subjects involved in the elections system.**

curity authority.

Some other information are involved in accessing these resources such as the vote counters location, the download speed of a connection, or the time that a voter can vote. We consider all these as contextual information. The properties of these sort of information are being stored using Context Ontology. We will discuss about the ontologies designed to control the accesses later.

## 5.2 An Access Control Scenario

In this scenario, assume that user $u\_1$ located in city $c\_1$ wants to vote in the elections. The elections are going to be held on June 12, 2009, 8am to 9pm. Also, We assume that it is essential for the voter to connect to the system within a download speed rate more than 128kb/s. To access the Elections System, the user should provide essential credentials that shows he/she is eligible to vote. The eligibility conditions are specified by Interior Minister. In the scenario, the meaning of eligibility is providing enough credentials showing that the user is Iranian, and is at least 18 years old. On the other hand, to vote for the parliament, $u\_1$ should access $les\_1$ which is an individual of $LES$. The security authority (*i.e.* Interior Minister) has annotated this individual to specify some individual-level policies. These policies say that to vote for parliament in city $c\_1$, $u\_1$ should prove his/her $c\_1$ residency. We assume that all credentials provided by user $u\_1$ are validated by Credential Verifier. Now consider the following scenario:

1. User $u\_1$ submits its request to PEP to access $les\_1$.

2. PEP redirects the access request to KB to extract the required credentials. The access request which includes the resource ($les\_1$) and the requested action ($vote$),

3. KB extracts the list of the required credentials to access the resource $les\_1$. The extraction is done in two steps:

   (a) First, the most specific concept of which the resource is member ($LES$), is found by KB. Then all the conceptual policies defined by the authority for this concept that are related to the requested action, are extracted. Therefore all the required credentials needed to access all individuals of type $LES$ are extracted.

   (b) Second, by checking the section $SecurityProfile$ in the file les_1.owl, all the security requirements and capabilities that are defined for performing voting on $les\_1$ are extracted. These are the individual-level policies of this resource.

4. PEP passes the list of the credentials needed for $u\_1$ to access PEP. These credentials include having Iranian nationality, residency of $c\_1$, and being older than 18.

5. $u\_1$ delivers the required credentials to PEP. These credentials should show that the user is Iranian, older than 18 and a resident of $c\_1$.

6. So now PEP does two things:

   (a) First, redirects the credentials to Credential Verifier to validate them.

   (b) Second, by passing the access request, asks KB to achieve and return all the required information for making decision for it.

7. The credentials are validated by Credential Verifier and the validity of the credentials are returned back to PEP. Meanwhile KB extracts the required information, and those which should be gathered from the environment (contextual information) are asked from Context Handler. In this case, the Context Handler should gather and return back the date, time and the location of the requested user.

8. The required information are gathered from the environment by the Context Handler and represented using OWL and returned to KB. (The way that the context is handled is discussed in [8]. we do not describe more in details here.)

9. The returned contextual information is stored in the CI by KBMS and besides all other required information is redirected to PEP.

10. Now PEP has all the required information including the validated credentials, the required credentials, the contextual information, the subsumption relations of the subject (credentials), object (resources), and action (the relationship between the actions related to the requested action), and the specified conceptual and individual-level policies. It passes these information to PDP, for decision making.

11. PDP by the help of its main component, *Inference Engine*, makes decision whether $u\_1$ is allowed to access $les\_1$ or not. The inference is done in two steps:

    (a) First, The individual-level credentials are checked to see whether the provided credentials are essentials or not. This is done by a matching algorithm using subsumption [7].

    (b) Second, if the first step is qualified, the correct format needed for the implemented prolog engine is created and the engine is run. This engine makes decision about the request using all the semantic information provided to PDP. This engine works based on the $MA(DL)^2$ logic [2].

12. If based on the provided information PDP decides that $u\_1$ is allowed to access $les\_1$, PEP lets him/her to access the resource and therefore vote.

# 6. USING SEMANTIC WEB TECHNOLOGIES

In designing the access control mechanism, semantic technologies are employed. These technologies are used in two different manners:

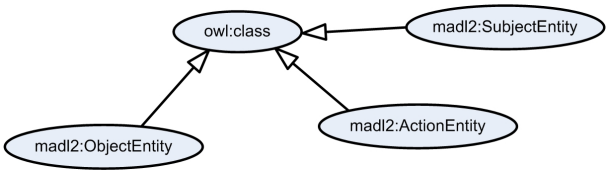


**Figure 4: The Subject, Object, and Action Ontologies structure.**

1. Using ontologies to present the structure of representing, and infering different information. The designed ontologies are as some languages to represent the semantic information used in inference. Some of these ontologies include ontologies for Semantic Web context, meta data, and security policy rules in conceptual level, or individual level. All these languages are being developed based on OWL to be consistent with Semantic Web.

2. Using repositories in type of OWL files. Information in Semantic Web is stored in some file types such as OWL, RDF, and DAML. Using databases is being considered as another approach. In the proposed framework, OWL files are employed. However the differences between the two approaches, and the reason of choosing this file type is discussed later.

In the rest of the section, the two manners are illustrated in more detail.

## 6.1 Ontologies for representing an inference language

Ontologies are powerful and expressive, and therefore can be used as some languages to represent information and infer other information. In the proposed framework, different ontological languages to represent different information are being employed. Some of these ontologies are discussed in this section.

### *6.1.1 Subject, Object, and Action Ontologies*

Every access control framework involves three essential entities: subject, object and action. These three components, respectively, represent the requesters, the requested resources and the requested actions to be performed on the resource. Using the MA(DL)$^2$ model, these three entities impact on access control indirectly. This means first the subsumption relations are being extracted and then these relation, are being employed in decision making. To represent these three entities, three ontologies named Subject Ontology (SO), Object Ontology (OO), and Action Ontology (AO) are designed. After populating these ontologies, they are parsed using Ontology Parser located in Ontology Manager and the resulting subsumption relations are stored in SUB repository in KB. As being depicted from Figure 4, three classes have been defined. Each class is a subclass of *owl:class* therefore any instance of these classes is a class itself. Such designing helps us to define different individuals of such entities and use them in access control. Now consider the elections system case study. A voter (subject) wants to vote (action) in an elections subsystem (resource). The class Voter can be described as bellow:

```
<SubjectEntity rdf:ID="Voter">
 <owl:intersectionOf rdf:parseType="Collection">
  <SubjectEntity rdf:about="#Upper18" />
  <SubjectEntity rdf:about="#Iranian" />
 </owl:intersectionOf>
</SubjectEntity>
```

and an instance of Voter can be defined like this:

```
<Voter rdf:ID="voter1" />
```

The resources LES, PES and MES can be defined using Object Ontology as bellow:

```
<ObjectEntity rdf:ID="ES" />
<ObjectEntity rdf:ID="PES">
 <rdfs:subClassOf rdf:resource="#ES" />
</ObjectEntity>
<ObjectEntity rdf:ID="LES">
 <rdfs:subClassOf rdf:resource="#ES" />
</ObjectEntity>
<ObjectEntity rdf:ID="MES">
 <rdfs:subClassOf rdf:resource="#ES" />
</ObjectEntity>
```

and an instance of LES can be declared:

```
<LES rdf:ID="lec_1"/>
```

As the last example, the action Vote and its instance can be defined as:

```
<ActionEntity rdf:ID="Vote" />
<Vote rdf:ID="vote" />
```

### *6.1.2 The Conceptual-level Security Policy Rules Ontology*

As discussed before, security policy rules in conceptual level are specified using the MA(DL)$^2$ logic language. In order to specify the rules in a consistent manner, we presented an ontology with the same expressiveness. In the ontology, an etitiy named SPRule exists which is equal to the Security Policy Rule in MA(DL)$^2$. This class and its properties are depicted in Figure 5. SPRule contains four properties including sprPrecondition, sprDeonticStatus, sprAuthority, and sprDoPredicate. These properties respectively represent the environmental preconditions which should happen, the deontic status that the rule implies, the authority that has specified the rule, and the do predicate to determine the subject, object and the action of the access control rule. As an example in the elections system case study, consider the following security policy rule in MA(DL)$^2$:

$cx_0 \wedge cx_1 \rightarrow PE_{InteriorMinister}\ \mathrm{do}(Voter, ES, Vote)$

This rule demonstrates that any subject with Voter credentials, is permissible to Vote in ES if the two propositions cx0 and cx1 exist. The rule, which was specified by Interior Minister can be represented by the madl2 ontology as bellow:

```
<SPRule rdf:ID="spr3">
  <sprPrecondition>
   <Formula>
    <first>
     <Proposition rdf:resource="#cx0"/>
    </first>
    <rest><first>
     <Proposition rdf:resource="#cx1"/>
    </first></rest>
   </Formula>
  </sprPrecondition>
  <sprDeonticStatus>
   <DeonticStatus rdf:resource="#PE"/>
  </sprDeonticStatus>
  <sprAuthority>
```

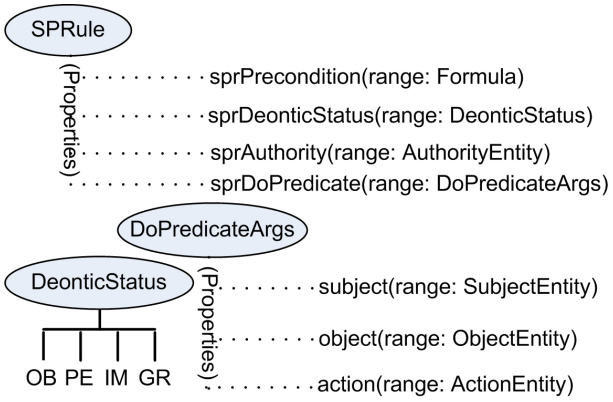


**Figure 5: The conceptual-level security policy rule ontology.**

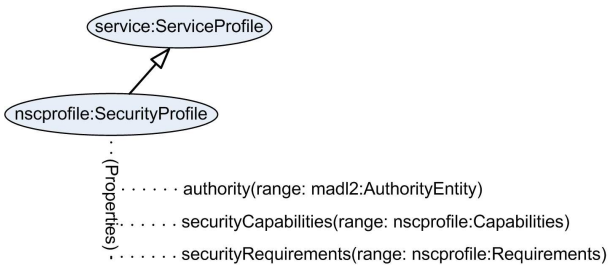


**Figure 6: Security Profile to specify individual-level policy rules.**

```
    <AuthorityEntity
          rdf:resource="#InteriorMinister"/>
   </sprAuthority>
   <sprDoPredicate>
    <DoPredicateArgs>
     <subject rdf:resource="#Voter"/>
     <object rdf:resource="#ES"/>
     <action rdf:resource="#Vote"/>
    </DoPredicateArgs>
   </sprDoPredicate>
</SPRule>
```

### 6.1.3 *The Individual-level Security Policy Rules Ontology*

The individual-level security policy rules can not be specified by the MA(DL)$^2$ logic. To specify these policies, another ontology is designed based on security capabilities and requirements. We presented a model for the resources in Semantic Web, in which every resource is assumed to be as a service provider and all actions a resource has are assumed to be a service [7]. Therefore, by considering this canonical model, the individual-level security policy rules can be annotated on their Semantic Web service definition. These security policy rules are a subclass of a ServiceProfile[6] and named as SecurityProfile (Figure 6).

For clarification consider the following example in the Elections System case study. As specified by Interior Minister, to access *les_c*1 it is required to be resident of city *c*_1. This rule can be specified as bellow:

```
<SecurityProfile rdf:ID="securityProfile_les_c1">
```

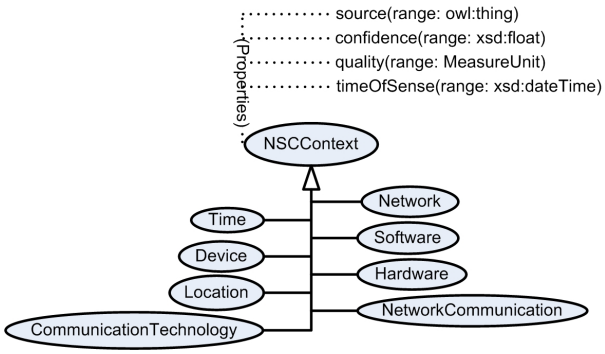


**Figure 7: The NSCContext ontology to represent the contextual information in Semantic Web.**

```
 <securityRequirements>
  <SecurityRequirements
        rdf:ID="SecurityRequirements_les_c1">
   <rdf:first>
    <madl2:SubjectEntity rdf:ID="ResidentOfC1">
     <rdfs:subClassOf rdf:resource="#Resident"/>
    </madl2:SubjectEntity>
   </rdf:first>
  </SecurityRequirements>
 </securityRequirements>
</SecurityProfile>
```

### 6.1.4 *The Context Ontology*

The propositions in conceptual-level security policy rules are all contextual information. To classify the Semantic Web contextual information, an ontology named NSCContext was presented [8]. The ontology root is named NSCContext and all represented contextual information are its subclasses instances. The classes of the ontology are shown in Figure 7. As an instance of represented contextual information consider an example from the Elections System case study. In the Elections System, an eligible voter can vote at June 12, 2009, 8am to 9pm. So this proposition can be represented as bellow:

```
<Time rdf:ID="cx0">
 <time:hasBeginning>
  <time:Instant rdf:ID="C0Beginning">
   <time:inXSDDateTime>
        2008-06-12T08:00:00
   </time:inXSDDateTime>
  </time:Instant>
 </time:hasBeginning>
 <source>
  <Company rdf:ID="Timeserver"/>
 </source>
 <time:hasEnd>
  <time:Instant rdf:ID="C0End">
   <time:inXSDDateTime>
        2008-06-12T21:00:00
   </time:inXSDDateTime>
  </time:Instant>
 </time:hasEnd>
 <timeOfSense>
        2007-03-23T00:00:00
 </timeOfSense>
```


**Figure 8: The MD ontology to represent the framework meta data.**

```
  <confidence>1.0</confidence>
</Time>
```

### 6.1.5 The Meta Data Ontology

The framework needs to store some meta data. These information are used to resolve conflicts, and store the repositories addresses and the default namespaces. The meta data information are represented using an ontology named MD. The ontology is shown in Figure 8.

## 6.2 OWL Files As Repositories

All information needed in the framework for decision making are stored in OWL files. These files are populated using the ontological languages described in the last subsection. Another way can also be used; using relational databases as information storage. In the following subsection these two solutions are compared and the reasons why we've chosen OWL files are discussed.

### 6.2.1 OWL Files VS. Database

Using OWL files have the following advantages:

1. These files are fully consistent with Semantic Web and this environment supports this file type.

2. By using these files, importing information from other namespaces and other domains in a distributed manner is available.

3. The ability to share, and exchange information between different domains using these files is facilitated.

Although there are several advantages using OWL files, some disadvantages also exist, which can be solved with database. These disadvantages include:

1. Considering the limited physical memory for implementing the framework, the OWL files may be troublesome. This problem occurs because of the need of collecting all ontologies located in the OWL file itself and others located in the imported files.

2. The volume of data used for access control become more and more during time; therefore the OWL files become larger and larger. As OWL files evolve, processing the data in these files will become time-consuming.

These two problems occur because in accessing OWL files, there is no certain way to collect only the needed information. Generally database management systems can solve the first problem by extracting the needed data on demand. As nowadays efficient DBMSs exist, the second problem can be solved too. Therefore, in the first mind, we thought that although using OWL files has several advantages, databases are more efficient and should be employed. After doing several tests, we understood there is no more efficiency using databases in our framework. The reason is that logic access control needs all information to make decision. So database management systems can not solve the first problem and all information should be passed and the only solution for that problem is to add more hardware. Although the second problem could have been solved by databases, because it is a minor requirement and can be partially corrected by some OWL tools such as Jena[3], is ignored. Finally, OWL files are being used as the repositories of the framework.

### 6.2.2 OWL Storing Files

There are several repositories, which store the information represented by the structural ontological languages discussed before. These files are as bellow:

- md.owl: This file contains all meta data needed for access control and is located in the predefined address */ontologies/md.owl*. This is the only predefined address used in the framework. The file contains the information that are logically stored in MD located in KB.

- so.owl, oo.owl, and ao.owl: These files are logically stored in Ontology Base and contain the information of the Subject, Object and Action Ontologies respectively.

- madl2Instances.owl: This file contains all the specified security policy rules which are represented using the OWL version of MA(DL)$^2$. These security policy rules are stored logically in SPR located in KB.

- [resourceID].owl: each resource contains a file in which the services it can serve are stored. These files are written in OWL-S and for each service its individual-level security policy rules (requirements and capabilities) are specified in it.

- sub.owl: In this file the extracted subsumption relations are stored. The file contains the data stored logically in SUB located in KB.

- ciInstances.owl: This file is to store the contextual information obtained from the environment. This is the place in where CI information located in KB are stored.

# 7. DISCUSSION

In order to compare the access control framework features with the other related works, we use two different trends. First, we compare the applied access control model, with some others introduced in section **??**. Second, we compare the framework itself with two other frameworks. As depicted in Table 1, We employed an access control model, which covers all the features of its decendents, in addition to supporting multi domains. Similar to other semantic access control models, MA(DL)$^2$ uses semantic relations among entities to decide about an access request. Moreover, contextual information are used in inferece performed in MA(DL)$^2$, but SBAC (Javanmardi *et al.*) and SAC do not. Similar to SBAC (Javanmardi

[3]http://jena.sourceforge.net/

**Table 1: Comparison between MA(DL)$^2$ and other access control models.**

| Model | Semantic-Aware | Context-Aware | Credential-Based | Subsumption Employed | Multi-domain Enabled |
|---|---|---|---|---|---|
| SBAC (Javanmardi *et al.*) | + | - | + | + | - |
| SBAC (Naumenko *et al.*) | + | + | - | - | - |
| SAC (Yague *et al.*) | + | - | + | - | - |
| MA(DL)$^2$ (Amini *et al.*) | + | + | + | + | + |

**Table 2: Comparison between the proposed framework and other access control frameworks.**

| Framework | Semantic-Aware | Context-Aware | Credential-Based | Using Policy Language | Multi-domain Enabled |
|---|---|---|---|---|---|
| XACML (Moses) | - | + | - | + | - |
| Rein (Kagal *et al.*) | + | - | - | + | - |
| Proposed framework | + | + | + | + | + |

*et al.*), MA(DL)$^2$ uses sumsumption relations and attribute certificates in the access control process.

The framework is compared with the Rein, and XACML frameworks.(Table 2). None of the frameworks use attribute certificates nor multi domain security policies, while the proposed framework does. Although XACML provides some definition in the policy language and offers some solutions to handle contextual information, there is no proposal detail to specify it. Our proposed framework facilitates context handling by defining an ontology to classify Semantic Web context and a framework to obtain it from the environment. Rein and our proposed framework both employ semantic relations for access control, however XACML does not.

# 8. CONCLUSION

While Semantic Web is evolving, security concerns increase. In order to overtop the concerns using semantic-based access control may be a good solution. Semantic-based access control models can be employed by authorities to specify their policies in semantic environments such as Semantic Web. The policies specified by security authorities should be enforced in the required environment. Access control mechanisms are designed to enforce them.

In this paper, MA(DL)$^2$ was employed to control accesses for Semantic Web. Semantic Web was divided into some security domains, each contains a security authority and a security agent. The authority is to specify the policies of how the domain resources can be accessed, and the agent is to enforce the policies using the designed mechanism. The designed mechanism used semantic technologies which are fully consistent with the nature of Semantic Web.

The paper described the designed mechanism implemented within an access control framework. In order to demonstrate the applicability of the mechanism a case study and scenario of elections system was illustrated. The comparison with the other frameworks shows how the designed framework solves others shortcomings.

# 9. REFERENCES


[1] ITU-T recommendation x.509: Information technology - open systems interconnection - the directory: Public-key and attribute certificate frameworks. Technical report, 2001.

[2] M. Amini and R. Jalili. Specification and inference of authorization and obligation policies using deontic logic for semantic-aware environments. In *4th Iranian Society of Cryptology Conference (ISCC'07)*, pages 175–184, Tehran, Iran, 2007.

[3] M. Amini and R. Jalili. A calculus for composite authorities' policy derivation in shared domains of pervasive computing environments. In *IEEE International Workshop on Internet and Distributed Computing Systems (IDCS'08)*, Bangladesh, 2008.

[4] D. Clarke, J.-E. Elien, C. Ellison, M. Fredette, A. Morcos, and R. L. Rivest. Certificate chain discovery in *spki/sdsi*. *Journal of Computer Security*, 9(4):285–322, 2001.

[5] E. Damiani, S. D. C. d. Vimercati, C. Fugazza, and P. Samarati. Semantics-aware privacy and access control:motivation and preliminary results. In *1st Italian Semantic Web Workshop on Semantic Web Applications and Perspectives (SWAP'04)*, Ancona, Italy, 2004.

[6] G. Denker, S. Nguyen, and A. Ton. Owl-s semantics of security web services: a case study. In *1st European Semantic Web Symposium*, volume 3053/2004 of *Lecture Notes in Computer Science*, pages 240–253, Heraklion, Greece, 2004. Springer Berlin / Heidelberg.

[7] M. A. Ehsan, M. Amini, and R. Jalili. Using semantic annotation to design security mechanisms based on the semantic-aware access control $MA(DL)^2$. In *5th International Isc Coneference On Inforamtion Security and Cryptology (ISCISC'08)*, pages 3–14, Tehran, Iran, 2008.

[8] M. A. Ehsan, M. Amini, and R. Jalili. Handling context in a semantic-based access control framework. In *5th International Symposium on Frontiers of Information Systems and Network Applications (FINA'09)*, Bradford, UK, 2009.

[9] S. Javanmardi, M. Amini, and R. Jalili. An access control model for protecting semantic web resources. In *2nd International Semantic Web Policy Workshop (SWPW'06)*, pages 32–46, Athens, GA, USA, 2006.

[10] S. Javanmardi, M. Amini, R. Jalili, and Y. GanjiSaffar. Sbac: A semantic based access control model. In *11th Nordic Workshop on Secure IT-systems (NordSec'06)*, Linkping, Sweden, 2006.

[11] L. Kagal, T. Berners-Lee, D. Connolly, and D. Weitzner. Promoting interoperability between heterogeneous policy domains. Technical report, W3, 2006.

[12] L. Kagal, T. Berners-Lee, D. Connolly, and D. Weitzner. Using semantic web technologies for policy management on the web. In *21st National Conference on Artificial Intelligence (AAAI'06)*, Boston, Massachusetts, USA, 2006.

[13] L. Kagal, T. Finin, and A. Joshi. A policy language for a pervasive computing environment. In *4th IEEE International Workshop on Policies for Distributed Systems and Networks*, pages 63–74, Lake Como, Italy, 2003.

[14] T. Moses. Xacml 1.1 specification set. Technical report,

Oasis, 2003.

[15] T. Moses. Xacml 2.0 specification set. Technical report, Oasis, 2005.

[16] A. Naumenko. Semantics-based access control ontologies and feasibility study of policy enforcement function. In *3rd International Conference on Web Information Systems and Technologies (WEBIST'07)*, volume Internet Technologies, Barcelona, Spain, 2007. INSTICC Press.

[17] A. Naumenko, A. Katasonov, and V. Terziyan. A security framework for smart ubiquitous industrial resources. In *3rd International Conference on Interoperability for Enterprise Software and Applications (IESA'07)*, pages 183–194, Madeira Island, Portugal, 2007. Springer.

[18] A. Noorollahi, M. Amini, and R. Jalili. A semantic aware access control model with real time constraints on history of accesses. In *International Workshop on Secure Information Systems (SIS'08)*, Wisla, Poland, 2008.

[19] A. Noorollahi, M. Amini, and R. Jalili. A temporal semantic-based access control model. In *13th International CSI Computer Science Conference(CSICC'08)*, Kish Island, Iran, 2008. Springer-Verlag.

[20] A. Noorollahi, M. Amini, R. Jalili, and J. H. Jafarian. A semantic aware history based access control model using logical time approach. In *IEEE International Workshop on Internet and Distributed Computing Systems (IDCS'08)*, Khulna, Bangladesh, 2008.

[21] M. I. Yague, M.-d.-M. Gallardo, and A. Mana. Semantic access control model: A formal specification. In *European Symposium On Research In Computer Security(ESORICS)*, volume LNCS 3679, pages 24–43, Milano, Italy, 2005. Springer-Verlag.

[22] M. I. Yague, A. Mana, and J. Lopez. A metadata-based access control model for web services. *Internet Research*, 15(1):99–116, 2005.

[23] M. I. Yague, A. Mana, J. Lopez, and J. M. Troya. Applying the semantic web layers to access control. In *the DEXA 2003 Workshop on Web Semantics (WebS'03)*, pages 622–62, Prague, Czech Republic, 2003. IEEE Computer Society.