Composing Access Control Policies for Semantic Web Services

Sudhir Agarwal
Institute of Applied Informatics and
Formal Description Methods (AIFB),
University of Karlsruhe, Germany
agarwal@aifb.uni-karlsruhe.de

Barbara Sprick and Sandra Wortmann
Information Systems and Security,
Department of Computer Science,
University of Dortmund, Germany
barbara.sprick@udo.edu

***Abstract*— Semantic web services promise a lot of new features like automatic discovery, composition, simulation and verfication to name a few. However, several security related issues have to be resolved before semantic web services can be employed in typical business scenarios. In this paper, we present an approach to enable access control for semantic web services. Our approach builds on the idea of autonomous granting of access rights, decision making based on independent trust structures and respects privacy requirements of the users. Our framework allows the specification and computation of complex access control policies in a manageable and efficient way. Therefore, our approach is useful not only in web services based applications (typically client-server architecture) but also in peer to peer and agent based applications.**

I. Introduction

With the advent of the Semantic Web [1], [2], [3], Web services have gained even more importance [4]. Semantic Web techniques, especially ontologies, allow to describe Web services with machine understandable semantics, thus enabling new features like automatic composition, simulation and discovery of Web services [4], [5]. The vision of the Semantic Web is to make the current Web more like an information system. In such an information system Web services play the role of operations available to the users. However, the use of Web services is not restricted to access information, but also in many other areas, for example electronic business and enterprise application integration.

Because of the vast heterogeneity of the available information, information providers and users, security becomes extremely important. Security related aspects are mostly classified in three categories, namely confidentiality, integrity and availability [6], [7], [8]. Access control, which means the users must fulfill certain conditions in order to access certain functionality plays an important role in all three fields. For example, a student must show her library card to borrow a book from the university library. In context of confidentiality, it means that a student has access to the information relevant to only her own library account and thus can not know which other students have borrowed which books. In context of integrity, it means that a student may not change or cause a change in information relevant to the library account of another student. In context of availability, access control helps to prevent denial of service attacks that can take place if the access is uncontrolled.

In this paper, we present a Semantic Web compatible approach for specifying and composing access control policies of Semantic Web Services. First we identify a number of requirements for access control policies for semantic Web service. We then combine two well founded techniques, namely DAML-S [4] for describing Web services and the policy algebra for composing access control polices introduced by Bonatti et al. in [9]. The process model of DAML-S allows to specify semantic Web services that are composed of other semantic Web services by operations like *sequence, choice, parallel, iteration*, etc. We show, how access control policies for composite Web services can be computed from the access control policies of the component Web services and from contracts that the provider of an composite service may have with providers of the component Web services. We also allow the use of partial access control policies for component services if, for example, only partial behaviour of the component Web service is employed for the composite Web service.

The paper is structured as follows: in section II, we motivate a few important requirements for access control of semantic web services. In sections III and IV we give short introductions to DAML-S [4] and the policy algebra introduced by Bonatti et al. in [9] respectively. In section V we present our main contribution by introducing an approach for specifying access control for semantic web services. In section VI , we show how our approach can be implemented with SPKI/SDSI. In section VII, we discuss some related work and finally we conclude in section VIII.

II. Access Control Requirements for Semantic Web Services

Current access control is mostly based on authentication which often requires proof of identity. Since this is not compatible with the obvious privacy requirements of users, we identify the following requirement.

Requirement 1: **Access control should be based on capabilities rather than on identities.**

Often access control requires central control, for example registration or for specification and verification of the access rights. Central control retards the spontaneity and reuse of access control policies and is thus inappropriate for highly distributed systems like the semantic Web. We thus identify the following requirement.

Requirement 2: **Access control mechanism may not require central control.**

In a dynamic and distributed environemt like the semantic web, web service providers act spontaneously and independently of each other. Hence, we identify the following requirement.

Requirement 3: **Each Web service provider must be able to specify the access control policy of her Web service autonomously**.

The access control policy of a Web service is specified by the provider of the Web service description (mostly identical with the provider of the Web service). An end user knowing some web services may combine few of them in some way to solve a certain task at hand. Prior to executing such a combination or plan she may want to know whether she can fulfill the access control policy of the combination (plan). Hence we identify the following requirement for specifying access control for Semantic Web services.

Requirement 4: **The framework must allow an end user to *check* and *prove* her eligibility for a Web service**.

Consider a Web service that offers electricity contracts and requires that the customer is at least 18 years of age. This requirement can be specified as access control policy of the Web service rather easily. However, the access control policies of most of the Web services are not so simple. For example, it is quite realistic that an electricity company offering such a Web service requires that the customer is at least 18 years of age as well as lives in a particular geographical region. The access control policy becomes even more complex when the access control requires not only that a user must have certain properties but also that a user may *not* have certain properties. For example, the customer may not have any outstanding accounts with the electricity company. We identify further requirements for specifying access control for Semantic Web services.

Requirement 5: **The framework must support the specification of complex access control requirements**.

Now consider that the electricity selling Web service has two input parameters, namely `deliveryAddress` and `noticePeriod`. The ”functional” precondition for the `deliveryAddress` is that it must be a valid address in Germany and for `noticePeriod` is that it must be either 1 month or 3 months. Further, the Web service's access control policy requires that contracts with one month notice period and delivery address outside a particular geographical region are closed only with users who can prove their Greenpeace membership. Hence, we see that the access control requirements of a Web service may depend on the requested functionality (controlled by the values of the input parameters) and that the provided functionality may depend on the access control conditions fulfilled by the requester.

Functional and access control related aspects are not always separable from each other. A composite Web service offers multiple functionalities (e.g. if it contains “choice”). These functionalities can have different access contol requirements. That is, it is not realistic to say that a composite Web service has one global access control policy and if a user fulfills the access control policy she has access to all the functionalities offered by the Web service and to none, if she does not.

Thus we identify that access control and functional aspects are not always independent of each other and consequently following requirement.

Requirement 6: **The framework must support the interplay of the access control and functional aspects of the Web services**.

Web services are typically distinguished in *atomic* and *composite* Web services. As the terms suggest, an atomic Web service is one that can not be further broken into parts, whereas a composite Web service is one that is decomposable into atomic and composite Web services, which are often referred to as component Web services. In addition to the set of component Web services, a composite Web service has a control flow and a data flow graph that contain information about how the component Web services are connected and how the data flows from one component Web service to another respectively.

Consider the following two Web services: (1) a Web service w_1 that offers Greenpeace membership and (2) our previous electricity selling Web service w_2 which requires Greenpeace membership for contracts with one month notice period for delivery addresses outside a particular geographical region. Now consider a composite Web service w_3 that first executes Web service w_1 and then Web service w_2, that is, it closes a Greenpeace membership before closing an electricity contract. Obviously, the access requirement ”Greenpeace membership” of Web service w_2 is fulfilled after the execution of Web service w_1 and hence Greenpeace membership is not required to access the composite Web service w_3 although it is required by its component Web service w_2.

While the access control policy of an atomic Web service can be specified directly, the access control policy of a composite Web service depends on those of its component Web services and thus must be computed by the provider of the composite Web service. Hence, we identify the following requirements for specifying access control for Semantic Web services.

Requirement 7: **The framework must support a Web service provider in *computing* the access control policy of a composite Web service**.

During the execution of a composite Web service, a component Web service can certify that the requester of the component Web service has certain attributes. To do so, the component Web service issues appropriate credentials by which the requester can prove the aforementioned attributes. If these attributes are required for the access of any of the subsequent component Web services, the requester does not need to be able to prove them at the beginning of the execution of the composite Web service. Rather, the composite Web service can pass the newly issued credentials that prove the required attributes on to the other Web service.

Requirement 8: **The framework must be able to deal with the capabilities that are certified to the requester *on the fly*, that is, during the execution of a composite Web service**.

Consider two Web services A and B and that A uses B. That is, A is composite Web service and B one of its component Web services. Assuming that the providers of A and B specify and manage the access control policies of their respective Web services autonomously (cf. requirement 3), we identify the problem, that the provider of A cannot know at the time of specification of the access control policy of A, that the access control policy of B will not be change in the future. Hence the provider of A may not wish to embed the access control policy of B hard-coded in that of A but rather in a more dynamic fashion that is compatible with the later changes in the access control policy of B. Consider the Web services w_1, w_2 and w_3 from our previous example. Suppose, that after the Web service w_3 has specified its access control policy the Web services Web service w_2 changes its access control policy by adding a further requirement that the user must be at least 18 years of age. If the access control policy of w_3 has embedded the access control policy of w_2 in a hard-coded fashion, it becomes inconsistent because it does not reflect the current requirements.

Requirement 9: **The specification of access control policies of composite Web services must be immune to the changes in the access control policies of its component Web services**.

Web services, especially composite Web services may offer different functionalities depending on the values of the input parameters. A provider of a composite Web service may wish to embed another Web service only partially, that is she may be interested in only a subset of its functionalities. Consider our previous example in the motivation of the requirement 6. If there is a composite Web service that offers water only to those who have a electricty contract with 3 months notice period, then the water selling composite Web service may wish to embed the electricty selling Web service so that the users can close an electricity contract with 3 months notice period, if the do not have it already. Obviously, the water selling Web service is not interested in electricity contracts with 1 month notice period and hence wishes to embed only the part of the electricity selling web service that sells contracts with 3 months notice period.

Requirement 10: **The framework must allow to identify the required partial behaviour of a composite Web service and to compute and integrate the access control requirements of the partial behaviour of a component Web service**.

A provider of a Web service A may have a contract with an other Web service B that has an impact on the access control policy of Web service B. If Web service A now embeds service B as a component service, then the contract between the service providers of Web services A and B, respectively, may lead to a substitution of a part of the access control policy of Web service B by a partial policy agreed on in the respective contract.

Requirement 11: **The framework must be able to deal with contracts that have impact on the access control policy of a component Web service**.

Consider a Web service a for paying the electricity bill and another Web service b that allows access to the contents of a magazine. Suppose, that everytime a user pays her electricity bill, she gets some points for every KWh she saves. The magazine Web services offers access to the magazine contents only in return for such points. When a user has collected enough such points she can show them to b and gain access to the magazine contents. However, if the shown points are not "consumed" by b, a user can show the "same" points again and again and gain access to the magazine contents, which is certainly not in the interest of b. Therefore we identify the following requirement

Requirement 12: **The framework must be able to specify consumable credentials. It must be possible to specify in an access control policy that a web service consumes some credentials**.

Often, the credentials that a user possesses have certain validity. A component Web service of a composite Web service verifies the credentials and makes an access decision right before it starts executing. This leads to the problem that if the previous component Web services consumes too much time then the credentials that were valid at the time of checking whether the user fulfills the access control policy may not be valid at the time of actual execution of one of its component Web services.

Requirement 13: **The framework must be able to specify the validity and reason about the execution time of a Web services**.

When a composite Web service acts as a mediator between a user and a component Web service, the output delivered by the component web service may contain (sensitive) information, that can be misused by the the composite Web service or it may not be compatible with the privacy requirements of the user. Consequently, the user may not want that the composite Web service experiences the contents of the outputs.

Requirement 14: **The framework must make sure that any information delivered by a component Web service is not misused by the composite web service**.

When a composite Web service acts as mediator between a user and a component Web service, and passes on credentials of the requester to the component Web service, the composite Web service acts on behalf of the requester. The requester as well as the component Web service want to ensure, that the transmitted credentials are not misused by the composite Web service, for example by using them multiple times without notifying the requester.

Requirement 15: **The framework must make sure that the credentials shown by a requester are not misused by a composite Web service**.

III. Introduction to DAML-S

DAML-S is a DAML+OIL ontology for describing Web services with the objective of making Web services computerinterpretable and hence enabling tasks like discovery, composition, simulation, interoperation and execution monitoring of Web services. DAML-S complements the various industrial efforts that are low-level, by providing Web service descriptions at application level [4], [5], [10]. DAML-S has three main parts, namely `ServiceProfile`, `ServiceModel` and `ServiceGrounding`. `ServiceProfile` contains properties related to the functionality a service offers and answers the question *what does a service do?*, `ServiceModel` contains properties related to the operation of a Web service and answers the question *How does a service work?* and `ServiceGrounding` contains properties related to the access to a Web service and answers the question *How can a service be accessed?*

A service profile provides a high-level description of a service and its provider. It is used to request or advertise services with discovery services and capability registries. Service profiles consist of three types of information: a *description* of the service and the service provider; the *functional behavior* of the service and several *functional attributes* tailored for automated service selection.

The operation of a Web service is described in terms of a process model, which details both the control structure and data flow structure of the service. Two main components of the process model are the *process ontology*, which describes a service in terms of its inputs, output, preconditions, effects and where appropriate, its component subprocesses; and the *process control ontology* which describes each process in terms of its state, including initial activation, execution and completion. The primary kind of entity in the process ontology is *process*. DAML-S distinguishes between *atomic*, *simple* and *composite* processes. Atomic processes are directly invocable and execute in a single step. Simple processes, on the other hand, are not directly invocable and are not associated with a grounded. They are rather used as elements of abstraction. Composite processes are decomposable into other (non-composite or composite) processes. Their decompositions are specified by control constructs *sequence*, *split*, *split+join*, *if-then-else*, *choice*, *while*, *repeat-until* etc. For details regarding operational semantics of the control constructs refer to [10], [11]

Conditions play a central role in service profile as well as service model. Currently, a condition in DAML-S is not further specified but used at various places, for example in the *if-then-else* construct. Further, DAML-S allows web services to have conditional outputs. Conditional outputs are parameters with a condition property and are not delivered always but only when the condition is true.

A grounding can be thought of as a mapping from an abstract to concrete specification of those service description elements that are required for interacting with a service. The grounding of a service has mainly to do with the protocol and message formats, serialization, transport and addressing. For more detailed information on DAML-S, refer to [4], [5], [10], [11].

IV. Introduction to policy-algebra

In this section we introduce the algebra for composing access control policies as it is described in [9].

A *ground authorization term* is a triple of the form (s, o, a) where s denotes a subject, o denotes a Web service and a denotes of conditional output of the Web service o. The tuple $<o, a>$ is called an interface. The expressed relationship is considered as a permission for s to use the interface $<o, a>$. An (access control) *policy* is a set of ground authorization terms.

The algebra allows policies to be restricted by posing constraints on their authorizations. For this purpose, Bonatti et al. make their algebra parametric with respect to a constraint language $\mathcal{L}_{acon}$. Policy expressions are syntactically built from policy identifiers and algebra operators as follows:

$$\begin{aligned} \text{E} ::= \;& \mathbf{id} \mid E + E \mid E \;\&\; E \mid E^C \mid o(E, E, E) \mid \\ & E * R \mid T(E) \mid (E) \\ \text{T} ::= \;& \tau \mathbf{id}.E \end{aligned}$$

Here, **id** is the token type of policy identifiers, E is the nonterminal describing *policy expressions*, T is a *template*, that represents partially specified policies, $C \in \mathcal{L}_{acon}$ is a restriction on policies. Note, that the templates are not policy expressions, only templates with actual parameters are.

The semantics of the described policy algebra is described as a function that maps each policy expression onto a set of ground authorization terms, and each template onto a function over policies. Here, the addition operator $(+)$ is interpreted as set-theoretic union. Similarly, the conjunction operator $(\&)$, and the subtraction operator $(-)$ are interpreted as set-theoretic intersection and difference, respectively. The scoping restriction (C), where the constraint "$C \subseteq \mathbf{Con}$" is a set of interfaces, is interpreted as selection of those authorization terms that satisfy C. Overriding $(o(E, E_1, E_2))$ is interpreted as the policy that results from policy E when a part of E, which is specified by means of a third policy E_3 is overridden by policy E_2. This operator is a derived operator: $o(E, E_1, E_2) = (E - E_2) + (E_1 \& E_2)$.

Policy identifiers are interpreted by an environment that maps policy identifiers to policies. The semantics ofthe policy algebra is a function e that maps each policy expression to a policy, inductively extending an environment by using the pertinent interpretation of the operators.

V. Specification of Access Control for Semantic Web Services

In this section we show how the described policy algebra can be integrated in DAML-S and how it can be used to compute access control policies of composite Web services.

A. Integration of access control with DAML-S

In this section, we show how the policy algebra introduced in section IV can be integrated with DAML-S to enable access control for semantic web services.

In figure 1, the concept `GroundAuthorizationTerm` with properties `subject`, `object` and `authorization` represents a triple of the form (s, o, a). Setting the range of the property `subject` to `Capabilities` allows to specify users of a Web service based on their properties (e.g. public key) and not on their identities. The `object` is the Web service itself. Hence, we set the range of `object` to `Process`. We set the range of `authorization` to `ConditionalOutput` since a conditional output corresponds to a functionality offered by a Web service.

Concept `AccessControlPolicy` represents an access control policy. The property `goundAuthorizationTerm` with range `GroundAuthorization` represents the set of ground authorization terms, the access control policy consists of. We view an access control policy of a Web service as a condition that a user has to fulfill to get access to the Web service. Hence, we model a concept `AccessControlCondition` as subclass of `Condition`. `AccessControlCondition` has properties `accessControlPolicy` of type `AccessControlPolicy` and `inputParameter`. A precondition of type `AccessControlCondition` means that the input parameter referred to by the property `inputParameter` should prove the satisfiability of the access control policy referred to by the property `accessControlPolicy`. The policy algebra operators and constructs can be modeled rather straightforward and are not shown in figure 1.

B. Computation of access control policies for compsite Web services

In this section we show how the introduced policy algebra can be used to compute access control policies of (composite) Web services.

We distinguish between atomic Web services and composite Web services as described in section III. Composite Web services are composed from component Web services by operations *Sequence (;), Choice (+), Parallel (||), Iteration (∗)*.[1]

The provider of an atomic Web service can specify the access control policy of her service autonomously and independently. As described in section IV, the access control policy $\Pi(w)$ of a Web service w is defined as a set of ground authorization terms of the form (s, o, a).

The provider of a composite Web service computes the access control policy of the composite Web service from the access control policies of the component Web services. As stated in requirement 7, the framework should support the Web service provider in computing the access control policies.

We will now show, how this can be done in the described framework. For the computation, the access control policies of the desired functionalities of the component Web services, the type of their composition and possible contracts between the provider of the composite Web service and the component Web services need to be taken into account.

[1]DAML-S support more control constructs, for example *if-then-else,repeat-until, unordered, split*, which we do not consider any further in this paper.

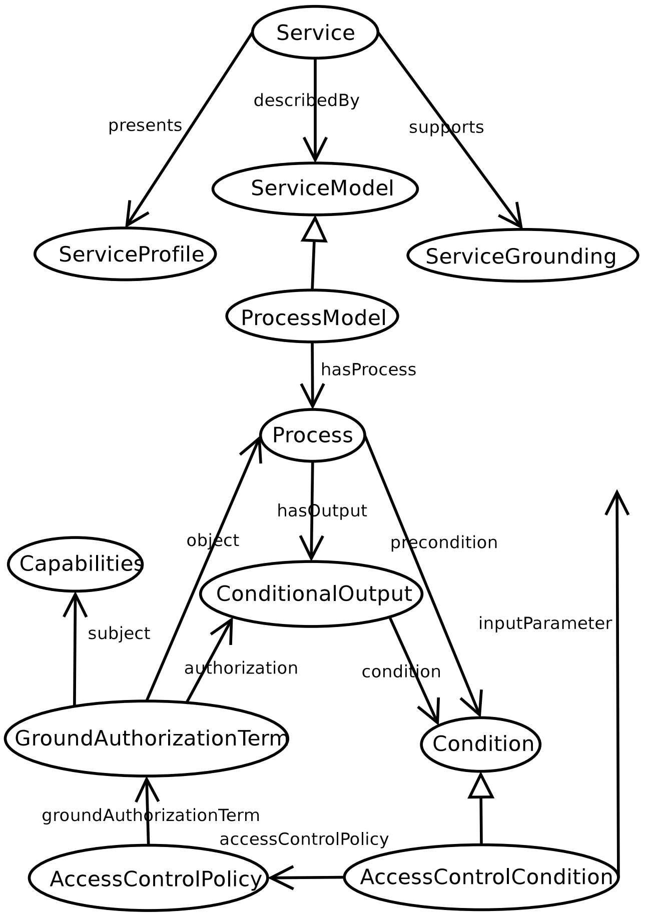

Fig. 1. Integration of DAML-S and Policy Algebra

1) Support of partial functionality of a Web service: According to requirement 10, a composite Web service does not always need the full functionality of a component Web service, but rather a subset of the functionalities (conditional outputs). Each functionality of a Web service is specified as an interface $<o, a>$. The access control policy of the component Web services will contain authorization terms for several interfaces, i.e. for several conditional outputs. In section IV we introduced the policy algebra parameterised with a constraint language $\mathcal{L}_{acon}$. We use the scoping operator for restricting the access control policies of a Web service to the access control policy of the *desired functionality*, i.e. the conditional outputs of the Web service. Therefore, we instantiate the parameter $\mathcal{L}_{acon}$ as follows: Let $Con \subseteq \mathcal{P}(O \times A)^2$ be a set of sets of interfaces. We then define $\mathcal{L}_{acon} := Con$, i.e. each set of interfaces $C \in Con$ is a constraint and thereby a possible scoping restriction for a policy Π. We say, a policy Π satisfies a restriction C iff the following holds: If $(s, o, a) \in \Pi$ then $<o, a> \in C$.

Consider, for example, a component Web service w with conditional outputs $\{co_1, co_2\}$. The access control policy $\Pi(w)$ of Web service w will contain authorizations for interface

[2]$\mathcal{P}(A)$ denotes the powerset of A

$<w, co_1>$ as well as for interface $<w, co_2>$. If composite Web service w' only requires conditional output co_1 of Web service w, the access control policy of Web service w' should contain authorization terms only for interface $<w, co_1>$ and not for interface $<w, co_2>$ too. The restriction of access control policy $\Pi(w)$ to the interface $<w, co_1>$ is then defined as $\Pi(w)^{\{<w,co_1>\}}$.

2) Support for independently specified access control policies: According to requirement 3, a Web service provider must be able to specify and modify the access control policies of his Web services autonomously and independently. This implies that a provider of a composite Web service does not know at design time how the access control policy of a component Web service will look like at the instantiation time. This requirement can be fulfilled by using the template operator of the introduced policy algebra. The template operator allows to specify that access control policy of a composite Web service contains the access control policy of a component Web service without actually inserting the current access control policy of the component Web service at the design time. The templates will be instantiated when the access control policy needs to be computed at the time of instantiation of the composite Web service.

Consider for example a Web service w that is specified as a sequence of component Web services w_1 and w_2. Obviously, the access control policies of the Web services w_1 and w_2 must be contained in the specification of the access control policy of the the Web service w. However, since the providers of Web services w_1 and w_2 can specify and modify access control policies of their respective Web service without notifying the provider of w, it is more appropriate if the provider of the composite Web service w uses template $\tau.X_1, X_2.(X_1 \& X_2)$.

3) Support for Contracts between Web Services: According to requirement 11, the framework should support contracts between Web service providers. These contracts may affect the access control policies of the Web services. In such a case, the provider of the composite Web service does not employ the full access control policy of the component Web service, but may want to replace a part of it with a policy the providers agreed upon in a contract. We consider following types of contracts.

1) The simplest case is to simply remove a part of the access control policy of the component Web service. Consider for example a composite Web service w that is composed of the component Web service w_1 and some other Web services. The access control policy of the Web service w_1 requires users to have a credit card. The providers of Web services w and w_1 may have a contract that says that the provider of Web service w will pay a monthly lumpsum to the provider of Web service w_1 and because of that, users who user Web service w_1 via w need not show their credit card. Let $\Pi(CreditCard)$ be an access control policy that requires users of w_1 to show their credit card. Let $\Pi(w_1)$ to the access control policy of w_1. Then the access control policy of Web service w, $\Pi(w)$, is calculated from the access control policy of the other involved Web services and from $\Pi(w_1)$ and $\Pi(CreditCard)$
2) In the second scenario a part of the access control policy of the component Web service w is supposed to substituted by a new policy. Consider for example the case when the contract specifies that user who use component web sercice w_1 via composite Web service w may give the bank account details instead of showing a credit card. The access control policy of Web service w is then calculated by the access control policies of the other involved Web services and $(\Pi(w_1) - \Pi(CreditCard))\&(\Pi(CreditCard) + \Pi(BankAccount))$.
3) In a third scenario, users who are principally entitled for a web serb service w_1 but are listed on a revocation list might still be allowed to user Web service w_1 by the Web service w if they fulfill some constraint π. In this case, the overriding operator of the algebra can be used to specify their modifications of the access control policy causes by the contract. Let $\Pi(w_1)$ denote the access control policy of Web service w_1. Let CRL denote the mentioned revocation list and let π denote the additional requirements for users that are on the revocation list to be allowed for w_1 via Web service w. Then $o(\Pi(w), \pi, CRL) = (\Pi(w_1) - CRL) + (\pi \& CRL)$ denotes the access control policy that allows users that either fulfill $\Pi(w_1)$ and are not on the revocation list CRL or are on the revocation list CRL but fulfill the condition π.

4) Computation of Access Control Policy: According to requirement 7, the framework must support the provider of a composite Web service in computing the access control policy from the access control policies of the component Web services. In this section, we show how this can be done.

In DAML-S description of a Web service, a Web service can be composed by using the control constructs *Sequence (;), Choice (+), Parallel (||), Iteration (*)*[3]. The table in Figure 2 shows for each control construct how a preliminary access control policy of the considered part of the composite Web service is computed from the component access control policies[4].

$w_1; w_2$	$\tau.X_1, X_2.(X_1 \& X_2)(\Pi(w_1), \Pi(w_2))$
$w_1 + w_2$	$\tau.X_1, X_2.(X_1 + X_2)(\Pi(w_1), \Pi(w_2))$
$w_1; w_2$	$\tau.X_1, X_2.(X_1 \& X_2)(\Pi(w_1), \Pi(w_2))$
w^*	$\tau.X_1.(X_1)(\Pi(w))$

Fig. 2. Computation of composed access control policies

In case, Web services are composed sequentially or in parallel, the access control policy of the composite services

[3]DAML-S allows other types of composition operations like *if-then-else, repeat-until, unordered, split* which we do not consider any further in this paper.

[4]Note that we assume that the component Web services w_i are already relevant parts of the web services $\bar{w}_i$ already identified by scoping, overriding, contracts etc.

is simply the conjunction of the component services, the requester of the composite Web service needs to fulfill the access control policies of both component services. In case, the Web service is composed as a choice between two component Web services, the requester needs to fulfill at least one the of the component services. In case, the composite Web service is an iteration of the component Web service, the requester needs to fulfill the access control policy of the component service. Note, that we assumed a capability based access control system where capabilities do not get revoked by using them.

The problem with the preliminary access control policy is that the interfaces of its ground authorization terms grant access to the components of the composite Web services but not to the composite Web service itself. We thus calculate the composite access control policy from the considered conditional output by substituting all interfaces of the component Web services by the interfaces for the considered condtional output. if Π^{co}_{prelim} is the preliminary access control policy for the conditional output co, and w is the name of the composite Web service, then the access control policy Π^{co} is computed as follows:

$$\Pi^{co}(w) := \{(s, w, co) \mid \text{there exists } (s, w', co') \in \Pi^{co}_{prelim}\}$$

Finally, we calculate the access control policy of the composite Web service as the union of the access control policiesfor the conditional outputs:

$$\Pi(w) := \bigcup \Pi^{co}(w)$$

VI. Implementation

In requirement 1 we suggested that access control should be based on capabilities rather than on identities. Thus, the subject s in a ground authorization term specifies a capability a user must have to get access to the interface $<o, a>$. The interface $<o, a>$ of a ground authorization term specifies a Web service o and a conditional output a of Web service o. Hence, the ground authorization term (s, o, a) specifies, that users having capability s have access to the functionality (conditional output) a of web service o.

We implement the policy algebra introduced in section IV with an extension of SPKI/SDSI as proposed in [12].

SPKI/SDSI is a credential based public key infrastructure resulted by merging SDSI (Simple Distributed Security Infrastructure) and SPKI (Simple Public Key Infrastructure). The main advantage of SPKI/SDSI compared to other credential based systems is that it does not require central control and allows users, e.g., Web service providers to specify their own trust structures independent of each other.

SPKI/SDSI supports two kinds of credentials, namely *name certificates* to bind principals to names and *authorization certificates* to bind authorizations to names. Besides name certificates and authorization certificates, SPKI/SDSI also provides access control lists (ACL) for specifying access control policies for some interface [13], [14], [15], [16].

We distinguish between atomic Web services and composite Web services. A provider of an atomic Web service specifies

- the functionality of her service w_1 as an interface (w_1, f) and
- the access control policy of her service autonomously by defining a set of authorization terms (p, w_1, f).

To specify the access control policy the provider defines an access control list acl_1 with respect to her Web service w_1. The access control list acl_1 consists of several entries. Each entry defines the authorised principals for a single functionality (w_1, f) and is implemented as an unsigned authorization certificate of the form

<`Self`,*Subject*,*Authorization*,*Delegation*,*Validity*>

1) `Self` represents the issuing principal, i.e. the provider of the Web service.
2) *Subject* denotes authorized principals.
3) *Authorization* specifies the granted permission, i.e. the functionality f of the Web service w_1.
4) *Delegation* is a boolean flag that specifies whether the authorised principals are allowed to forward the granted permission.
5) *Validity* denotes the validity of the certificate.

Authorised principals are defined by proven capabilities. In the simplest case the provider of the Web Sevice demands one capability. According to the extension of SPKI/SDSI, *Subject* may provide algebra expressions that allow the provider of the Web service to denote combined capabilities. Therefore the provider may use some operators, namely Addition $(+)$, Conjunction $(\&)$ and Subtraction $(-)$. During verification, an algebra expression is evaluated to a set of authorized principals. Furthermore the provider of the Web service gives trusted participants whom she trusts to state demanded capabilities.

A provider of a composite Web service w_{comp} composes her service from functionalities of atomic (component) Web services. Afterwards she computes the access control policy of w_{comp} from those of its components. The first step is adaption of the interface that results in a "new" interface (w_{comp}, f_{comp}).

To specify the access control policy the provider defines an access control list acl_{comp} with respect to her Web service w_{comp}. The access control list acl_{comp} consists of several entries for each interface (w_{comp}, f_{comp}). Again each entry is implemented as an unsigned authorization certificates of the form

<`Self`,*Subject*,*Authorization*,*Delegation*,*Validity*>

1) `Self` represents the issuing principal, i.e. the provider of the Web service.
2) *Subject* denotes the authorized principals, which are specified as shown in Figure 2.
3) *Authorization* specifies the granted permission, i.e. the interface (w_{comp}, f_{comp}).
4) *Delegation* is a boolean flag that specifies whether the authorised principals are allowed to forward the granted permission.
5) *Validity* denotes the validity of the certificate.

During verification, the specification given in *Subject* must be evaluated to a set of authorized principals. Therefore, the Web service provider instantiates each occurence of a Template operator by inserting the current access control policy of the component Web Services. To do so, the Web service provider needs to communicate with the component Web service providers. We have not considered secure communication of access control policies between Web service providers yet. Afterwards *Subject* is an algebra expression as introduced and is evaluated as already mentioned.

VII. Related Work

In our knowledge, the first work that addressed the issue of security and DAML-S is [17]. In the mentioned work authors focus on developing security-related ontologies and give several security-related ontologies that are designed to represent well-known security concepts. They introduce a reasoning engine for a two step matchmaking. In [18], authors introduce a policy language that allows policies to be described in terms deontic concepts and models speech acts, which allows the dynamic modification of existing policies, decentralized security control and less exhaustive policies. However, it is not yet clear, how this policy language can be integrated and used with a Web service description language e.g., DAML-S. In [19], authors propose adding privacy and authentication annotations, for example, cryptographic type, to input and output parameters to aid in selection of semantic web services.

VIII. Conclusion

In this paper, we have presented an approach for specifying access control policies for semantic web services. We have identified some important requirements for an access control enabled semantic web services framework. We integrated a policy algebra with the semantic web service description language DAML-S at the specification level and show how access control policies of composite web services can be computed from the structure of the composite web service and the access control policies of its component web services. Then we presented how such specifications can be implemented by using SPKI/SDSI.

References

[1] T. Berners-Lee, J. Hendler, and O. Lassila, "The semantic web," *Scientific American*, May 2001.

[2] D. Fensel, J. Hendler, H. Lieberman, and W. Wahlster, Eds., *Spinning the Semantic Web*. MIT Press, 2002.

[3] P. Patel-Schneider and D. Fensel, "Layering the semantic web: Problems and directions," in *ISWC2002: Ist International Semantic Web Conference, Sardinia, Italy*, ser. Lecture Notes in Computer Science. Springer, June 2002, pp. 16–29.

[4] A. Ankolekar, M. H. Burstein, J. R. Hobbs, O. Lassila, D. Martin, D. V. McDermott, S. A. McIlraith, S. Narayanan, M. Paolucci, T. R. Payne, and K. Sycara, "DAML-S: Web Service Description for the Semantic Web," in *ISWC2002: Ist International Semantic Web Conference, Sardinia, Italy*, ser. Lecture Notes in Computer Science. Springer, June 2002, pp. 348–363.

[5] M. Burstein, G. Denker, J. Hobbs, L. Kagal, O. Lassila, D. Martin, D. McDermott, S. McIlraith, S. Narayanan, M. Paolucci, B. Parsia, T. Payne, E. Sirin, N. Srinivasan, and K. Sycara, "Daml-s: Semantic markup for web services, version 0.9," DAML-S Services Coalition, Tech. Rep., 2003.

[6] M. Bishop, *Computer Security – Art and Science*. Addison Wesley, 2003.

[7] P. Samarati and S. Capitani di Vimercati, "Access control: policies, models, and mechanisms," in *Foundations of Security Analysis and Design (FOSAD)*, ser. Lecture Notes in Computer Science, R. Focardi and R. Gorrieri, Eds., vol. 2171, FOSAD 2000, Bertinoro, Italy. Springer Verlag, Berlin, October 2001, pp. 137–196.

[8] D. E. Denning, *Cryptography and Data Security*. Addison Wesley, 1982.

[9] P. Bonatti, S. de Capitani di Vimercati, and P. Samarati, "An algebra for composing access control policies," *ACM Transactions on Information and System Security (TISSEC)*, vol. 5, no. 1, pp. 1–35, February 2002.

[10] K. Sycara, M. Paolucci, A. Ankolekar, and N. Srinivasan, "Automated discovery, interaction and composition of semantic web services," *Journal of Web Semantics*, vol. 1, no. 1, pp. 27–46, 2003.

[11] S. Narayanan and S. A. McIlraith, "Simulation, verfication and automated composition of semantic web services," in *Eleventh World Wide Web Conference, WWW02*, May 2002.

[12] J. Biskup and S. Wortmann, "Towards a credential-based implementation of compound access control policies," University of Dortmund, Tech. Rep., 2003, http://ls6-www.cs.uni-dortmund.de/issi/publications.

[13] C. M. Ellison, B. Frantz, B. Lampson, R. L. Rivest, B. M. Thomas, and T. Ylonen, "Simple public key certificate," http://world.std.com/~cme/html/spki.html, July 1999.

[14] ——, "SPKI certificate theory," Internet RFC 2693, September 1999.

[15] R. L. Rivest and B. Lampson, "SDSI - a simple distributed security infrastructure," http://theory.lcs.mit.edu/~cis/sdsi.html, April 1996.

[16] S. Agarwal, B. Sprick, and S. Wortmann, "Credential based access control for semantic web services," in *AAAI Spring Symposium 2004 - Semantic Web Services (To appear)*, March 2004.

[17] G. Denker, L. Kagal, T. Finin, K. Sycara, and M. Paolucci, "Security for daml web services: Annotation and matchmaking," in *Second International Semantic Web Conference*, ser. Lecture Notes in Computer Science, vol. 2870. Springer, October 2003.

[18] L. Kagal, T. Finin, and A. Joshi, "A policy based approach to security for the semantic web," in *2nd International Semantic Web Conference (ISWC2003)*, ser. Lecture Notes in Computer Science, vol. 2870. Springer, October 2003.

[19] G. Denker, L. Kagal, T. Finin, M. Paolucci, N. Srinivasan, and K. Sycara, "Sowl: A security infrastructure for owl-s - an approach to confidentiality and integrity," in *AAAI Spring Symposium 2004 - Semantic Web Services (To appear)*, March 2004.