Ontology-based Secure XML Content Distribution

Mohammad Ashiqur Rahaman, Yves Roudier, Philip Miseldine and Andreas Schaad

Abstract This paper presents an ontology-driven secure XML content distribution scheme. This scheme first relies on a semantic access control model for XML documents that achieves three objectives: (1) representing flexible and evolvable policies, (2) providing a high-level mapping and interoperable interface to documents, and (3) automating the granting of fine-grained access rights by inferring on content semantics. A novel XML document parsing mechanism is defined to delegate document access control enforcement to a third party without leaking the document XML schema to it. The *Encrypted Breadth First Order Labels (EBOL)* encoding is used to bind semantic concepts with XML document nodes and to check the integrity of a document.

1 Introduction

The increasing standardization of XML processing (e.g. XML Schema, DTD, XSL) makes it possible for peer organizations to cooperate and to integrate their information systems through XML document production and exchanges. Documents are structured and modeled through XML schemas in peer organizations. Schemas may contain valuable and confidential information about resources, strategies, services, or information system structure closely tied to business processes which organizations do not want to expose. The data model may evolve due to changes in the organization, for instance after a merger; existing data exchanges with peers should however be maintained. We claim that, although data models may differ from one organization to another or vary with time, the semantics of document data units like subtrees or nodes might constitute a more stable and interoperable interface between organizations. Semantic Web languages like RDF [3] and OWL [2] make it possible to share an ontology describing a conceptual data model, independently from XML data structure yet that can be mapped to instances of XML schemas. We also claim that access control can be defined at the semantic level notably to achieve a simpler expression of policies with complex organization rules and constraints. First, expressing access rights over a single con-

Mohammad Ashiqur Rahaman, Yves Roudier, Philipe Miseldine and Andreas Schaad. SAP Research, EURECOM. {mohammad.ashiqur.rahaman, philip.miseldine, andreas.schaad}@sap.com, {yves.roudier, mohammad.rahaman}@eurecom.fr

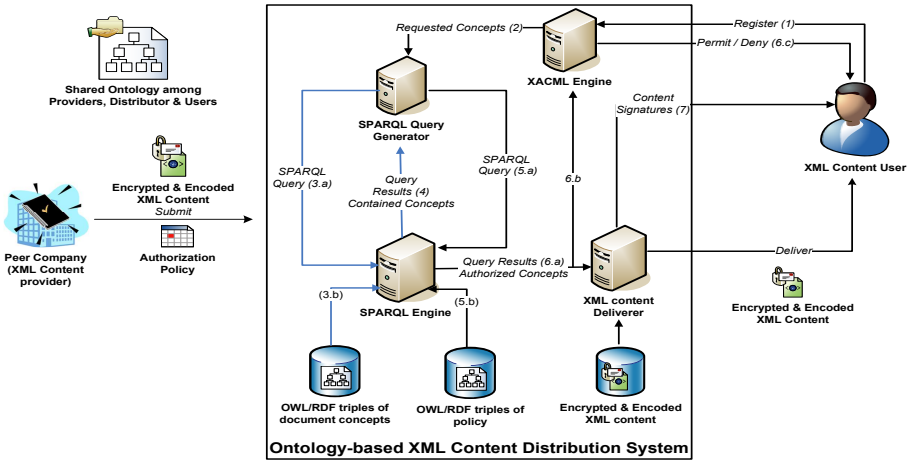


Fig. 1 The Ontology-based XML Content Distribution System. The numbered lines depict the sequence of operations upon a registration request.

cept might result into granting authorizations to multiple XML documents or portions thereof. Second, and more importantly, authorizations on concepts might be automatically inferred from the expression of the right to access a related concept. Third and finally, as shown in related work like Rei [13], ontologies can formally describe an access control model by representing policy concepts as first-class objects. We contend that this feature is particularly suitable to inter-organizational document exchange systems, by making it possible to store incremental versions of access control policies, possibly timestamped in the same fashion as documents to which they apply, thereby easing user revocation.

This paper describes access control mechanisms addressing all three objectives: our solution integrates ontologies for describing and reasoning over documents and authorization policies, which we implement using SPARQL [4] together with XACML [11]. We assume a large scale system where documents have to be distributed to many users: scalability is an essential issue here, and the content providers can not serve content to a large number of users nor to authenticate each of them. Documents may be updated, even after they are initially released by their provider. We assume a third party which we term a distributor, takes care of the transient storage and of the distribution of documents. Its role is important since users may not be online when a document is sent around. Message oriented middleware (MOM) or publish/subscribe paradigms provide examples of middleware adapted to such tasks. Content providers and users may pertain to different organizations and even be competitors: not every document should thus be readable by any user (Fig 1).

2 Solution Overview

Semantic data model. A domain-specific ontology provides the common language to communicate the contents of an XML document. Fig 2 shows a semantic graph of concepts as it may be defined through such an ontology (e.g. work order, production and quality-inspection). It also illustrates how these concepts may be mapped to XML documents in a manufacturing production environment scenario. Two document providers

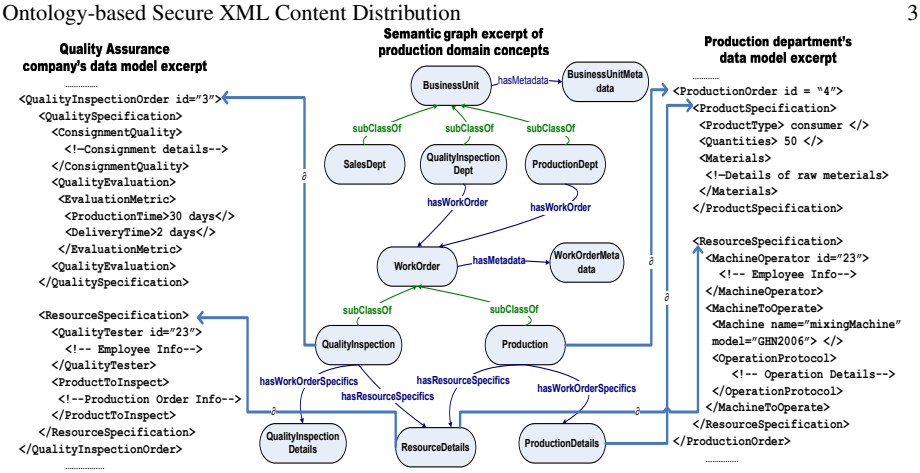
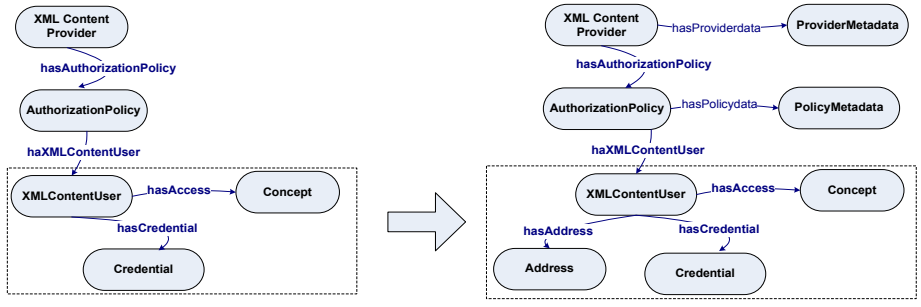


Fig. 2 A semantic graph of work order document concepts in a production domain. The 'Production' work order, 'QualityInspection' work order and 'ResourceDetails ' concepts are mapped to the corresponding XML data model excerpts using a mapping relation, ∂.

are considered here, the production department and the quality inspection company. Conceptually, the metadata (e.g. ID, priority level (urgent, normal, escalated), issue date) for all work orders (production order, quality-inspection order) would be the same for all work orders. However, each work order contains specific details that will be taken into account by a specific business unit. For example, the quality inspection order would carry information regarding the specification of the product quality and the metrics to measure them.

Document encoding. Users and providers will not share all existing XML schemas, since these describe the provider's information system organization. Authorizations will be given to users to access contents related with particular semantics, as described through the concepts of an ontology. We assume the distributor is trusted by providers to host and to selectively deliver their contents to authorized users only. The distributor has access to the semantics of every node he receives from the provider. While it can decide whether to forward that node to a user, it should not know the structure of complete documents. On the other hand, an authorized user should clearly be able to read some content he receives. Such a secure exchange of documents can be achieved through the separate encryption of each document node with a secret that the provider and the consumer share. At the middleware level, a concept and the document portions to which it maps are encoded together by a content provider. The concepts described in that encoded document will be accessible by distributors. The document encoding will however hide the structure of the schema underlying the document and protect the content through encryption and integrity protection measures. Providers will define explicit access control rules and will also likely issue inference rules describing how to generate new access control rules. For instance, additional access rights might be granted on a subclass of a granted concept. Some inference rules might also describe constraints and prevent a single user from being granted two exclusive authorizations. The distributor enforces the authorization policy defined by the provider. XACML uses the notion of subjects, resources, and actions, to describe access control rules. In our setting, a user would be modeled as a subject, and ontological concepts as resources.



(a)

XML provider	user_credentials	C_i	O
P_1	Cred1	WorkOrder	O_1
P_2	Cred2	QualityInspection	O_2

(b)

XML provider	U_{add}	user_credentials	C_i	O
P_1	www.one.com	Cred1	WorkOrder	O_1
P_2	www.two.com	Cred2	QualityInspection	O_2

Fig. 3 The table represents the policy specification by the XML content providers. Policy ontology is maintained by the distributor. (a) Initial policy ontology.(b) Updated policy ontology.

Actions would largely consist in read, delete, and write, to describe the usage governing the mapped XML content.

Interaction Phases. We consider five basic phases in our document distribution system. In the first phase the provider sends encoded and encrypted XML content to the distributor using the *EBOL* technique detailed in Section 4 (Fig 1). Associated authorization policies might also be sent to the distributor which will enforce them on behalf of the provider. In a second phase, the user registers for some concepts with the distributor. The user has to provide valid credentials to access XML content mapped to the requested concept as discussed in Section 3. Credentials might for instance consist of certificates issued by some authority. Depending on the applicable authorization policy, the distributor then sends a set of *content signatures* (cf. Section 4) to the authorized users. The *content signature* describes the encoding, and serves as a mean to verify the XML content subsequently distributed. In a third phase, the distributor performs a selective delivery of relevant XML contents to registered users. It determines and extracts the authorized content out of the documents sent by one or multiple providers. This process is performed over the encoded and encrypted XML content. The user verifies the received XML content, both semantically and structurally, in a fourth phase using the *content signatures*. The fifth phase is the unregistration of a user. It may occur at user's request, or be forced by the distributor if the user credentials expired or if the provider's policy is changed. This final operation is outside the scope of this paper.

3 Authorization Policy

3.1 Ontology-based Data Model

This section describes the ontology-based data model used to express flexible authorization policies. A concept C_i is an abstraction that can be communicated among peers. An ontology is a shared set of concepts in a domain. The ontology is defined primarily by the notions of *class*, *subclass*, and *properties* representing concepts and their relationships using OWL [2].

Definition 1 *Concept Containment: Let $\mathscr{C}$ be the collection of all concepts and $C_i, C_j \in \mathscr{C}$. If there is a subclass hierarchy from C_i to C_j denoted as $C_i \Rightarrow, ..., \Rightarrow C_j$ then C_i contains C_j and noted as $C_i \preceq C_j$.*

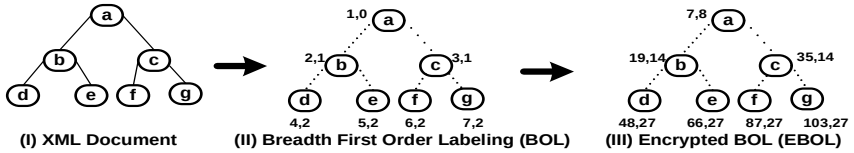


Fig. 4 (I) XML document tree. (II) BOL labeling. (III) Encrypted BOL labeling. Solid and dotted lines respectively depict explicit (I) and implicit (II,III) hierarchy representations and storage.

Example: Fig 2 shows a collection of concepts $\mathscr{C} = \{BusinessUnit, BusinessUnitMetadata, \text{etc.}\}$ for a production hall. *WorkOrder* contains *QualityInspection* and *Production*, i.e. $WorkOrder \preceq QualityInspection, WorkOrder \preceq Production$. □

3.2 Ontology-based Authorization Policy

We describe an ontology-based authorization policy as a set of explicit rules constructed as follows ($[x+]$ is used to denote a non-empty set of elements of type x):

1. Rules take the general form $[user\_credentials, [C_i]+, \mathscr{O}]+$ stating that access over one or more concepts C_i is allowed to the user holding *user_credentials* provided $\mathscr{O}$ is true.
2. Expression $\mathscr{O}$ characterizes relationships and constraints verified by browsing the semantic graph (such as of Fig 2). This expression enables a provider to restrict eligible concepts of the ontology, and may be parameterized by *user_credentials* or elements of $[C_i]+$, as described in Section 5.

Fig 3(a) shows an example of a policy specified by two XML content providers P_1(i.e. Production department) and P_2 (i.e. Quality assurance company) of the Fig 2. $\mathscr{O}_1$ for the user with credential $Cred1$ is: if a user is allowed to access the concept *WorkOrder* then he is also allowed to access to all the contained concepts of *WorkOrder*. $\mathscr{O}_2$ for the user with credential $Cred2$ is: he is allowed to access the concept *QualityInspection* if he has access to the concept *ResourceDetails*. The distributor describes such policies of the providers and generates policy instances as an OWL triple (see Fig 1 and 3) using SPARQL. Any change in the policy such as adding an address parameter for request filtering or adding metadata about provider and policy (shown in Fig 3(b)) would introduce additional concepts and relationships among them. For example, P_2 may add a constraint, $\mathscr{O}_2$, expressing that the user is allowed to access *QualityInspection* if any other provider allows the user the same access right.

4 XML Parsing, Encoding and Encryption

Encoding requires parsing the XML document: we use a breadth-first order technique to parse the XML nodes level by level from root to the leaves and to encode structure and conceptual information on the fly (Fig 4). This section describes the mapping of concepts to XML data units and the parsing, encoding, and encryption method in detail.

4.1 Ontology Mapping to XML Structure

An XML document, d, identified by doc_{id} (e.g. URI, RDF) is a collection of parsed XML nodes and a document portion d_i is a subtree rooted at node i of d. A mapping defines relations (∂) from a concept and its sub-class hierarchical path to document portions d_i which is used to determine the XML content associated to concepts. Such a mapping is illustrated by the following example.

Example: In Fig 2, the concepts *ProductionDetails* and *ResourceDetails*, identified by the paths over the semantic graph *BusinessUnit.ProductionDept.hasWorkOrder.Workorder.Production.ProductionDetails* and *...Production.ResourceDetails* are mapped to the document portions rooted at `<ProductSpecification>` and `<ResourceSpecification>` of the production department's XML data model. In the quality assurance company's data model, the concepts *ResourceDetails* and *QualityInspection*, identified by the path expressions *...QualityInspection. ResourceDetails* and *...QualityInspection* are mapped to the document portions rooted at `<ResourceSpecification>` and `<QualityInspectionOrder>` respectively. □

4.2 Encrypted Breadth-First Order Labels for XML Parsing

Once the mapping is done the provider parses the XML documents as follows: sibling nodes are stored into a FIFO queue and associated a BOL (an integer pair as defined below) capturing various structural relationships of the parsed XML node (i.e. parent-child, siblings, left/right child) with a minimal memory footprint.

Breadth First Order Labels (BOL): A *BOL* is a pair of integers associated to an XML node as it is parsed in breadth first order . The first integer in the pair is the order associated with a node whose left siblings and ancestors have already been parsed and thus have associated BOLs. The second integer is the depth of the node in the document which is increased by one as new depth level is reached. The BOL starts with (1,0) as illustrated in Fig. 4 (the example given is a binary tree, but BOLs can be defined on any type of tree)

Let a be the parent of two nodes $b, c \in d_i$. We denote its BOL as B_a. Let f_{order} and f_{level} be two functions operating on a BOL respectively returning the BOL order (first attribute of the BOL pair) and BOL depth (second attribute). Let us assume that b is the last child of a parsed and that c is to be parsed next. c will be associated a BOL with $f_{order}(B_c) = f_{order}(B_b) + 1$. $f_{level}(B_a)$ uniquely identifies the depth level of the node a in d. The order of the BOL exhibits the following structural properties:

1. $f_{order}(B_a)$ uniquely identifies node a in document d and the subtree d_a rooted at a.
2. Let $B^a_{Highest}$ be the largest BOL order of a parsed node in document portion d_a; then $B^a_{Highest} > f_{order}(B_z) > f_{order}(B_a)$, where $z \in d_a$.
3. $f_{order}(B_c) > f_{order}(B_b) > f_{order}(B_a)$.

The first property is used to identify and extract a specific document portion from a document. Combined with the depth level of a node, that property ensures that any unexpected move, copy or replace activity in the document is detected. The second property imposes an upper bound on the BOL of any queried node parsed in a document. In effect, it detects if a node is added or deleted and which one it is. The third property permits detecting any unintended swapping among the children in a received document portion (subtree).

A BOL is by definition plain text and thus may reveal important structure specific information (i.e. information leaking), such as number of nodes and thus the size of the document and even hierarchical relationship among the nodes to an adversary. Encryption over such BOL numbers protects this undesired information from leaking.

Encrypted BOL (EBOL): Let B_a be the BOL of an XML node a. Let f_e be an order preserving encryption function [5]. The EBOL of a, denoted as E_a is a pair of

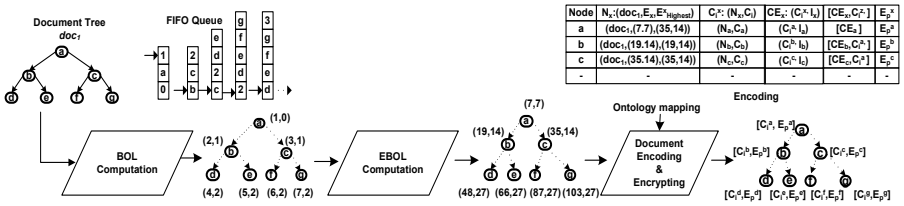


Fig. 5 Execution steps of the XML processing by document providers.

integers defined as : $(f_e(f_{order}(B_a)), f_e(f_{level}(B_a)))$. While $f_e(f_{order}(B_a))$ is performed for each node a, $f_e(f_{level}(B_a))$ is performed if a is the first node in a level.

The EBOL preserves exactly the same properties of BOL (see Fig 4). The EBOL order value hides the actual node number and its depth level as opposed to the BOL attributes and thus prevents information leaking.

4.3 Encoding Method

In the following, encoding elements are introduced to describe concepts that are mapped to data units (i.e. subtrees or nodes) as well as the properties of these data units and their encryption.

Node Identifier: Let x be a node in d_i. The node identifier of x denoted as N_x is a tuple formed by three elements $(doc_{id}, E_x, E^x_{Highest})$, where doc_{id} is the document identifier of d_i, E_x is the EBOL of x, $E^x_{Highest}$ is the highest EBOL in the document portion rooted at x. A node identifier is unique for all documents in the system. The depth included in E_x uniquely determines the node's level. E_x and $E^x_{Highest}$ together determine the parsed document portion. Finally, doc_{id} resolves appropriate XML nodes of the associated document with respect to the same concept.

Node Integrity: The node content consists of attributes, their values and text content inside the tag but not any descendants of the node. The node integrity code is a hash computed out of the concatenation of a node identifier and content, denoted as $I_x = H(N_x, Ct_x)$, where N_x is the node identifier, Ct_x is the content of x, and H is a one way collision resistant hash function.

Content Signature: Let C_i and x be a concept and an XML node respectively. The content signature, denoted as C^x_i, is a pair (N_x, C_i), where N_x is the node identifier of x and C_i is a concept mapped to x. The *content signature* incorporates semantic information such as conceptual and structural information attached to an XML nodes.

Content Encoding: An encoding information CE_x of a node x is $CE_x = (C^x_i, I_x)$, where C^x_i is the *content signature* and I_x is the node integrity respectively. Each XML node x is encoded as a pair $[CE_x, C^z_i]$, where CE_x is the encoding information of node x and C^z_i is the *content signature* of the parent node z of x. For the root node of a document the encoded node is $[CE_x]$.

Document Encryption: Each encoded node is encrypted using a key shared between the content provider and the content user. After encryption, an XML node x is represented as $[C^x_i, E^x_p]$, where C^x_i is the *content signature* of x and E^x_p is the encrypted value of the content encoding pair $[CE_x, C^z_i]$ of the node x.

Fig 5 depicts the encoding and encryption processing of XML nodes using EBOL described above.

5 Access Control Enforcement and Distribution

Semantic Access Control. The distributor maintains the shared OWL ontology describing the document concepts (Fig 2). It also maintains an OWL ontology describing the providers' authorization policies (Fig 1) so as to enforce access control through selective data distribution. Deciding on eligible concepts for a user as well as finding which access control rules apply requires reasoning on these ontologies. We suggest the use of SPARQL [4] as a way to implement such inference rules. A SPARQL query can be crafted to find concepts which a user can be implicitly granted access to starting from one concept to which the user is explicitly granted access. The result to such a query would for instance consist in a set of concepts related through a subclass relationship and that should equally be granted access according to the provider policy or to some domain-specific knowledge. SPARQL queries over the document concepts allow us to reason about the semantic graph patterns. SPARQL queries over the policy ontology can also be used to reason and evaluate the policies by dynamically computing the aggregated authorized concepts for a user. To this effect, the distributor would have to host an engine like Joseki [1] to interpret queries.

The distributor must host a XACML engine to evaluate a registration request for concepts and return a response (i.e. Permit/Deny) to the user. In case of a "Permit" it responds by sending the *content signatures* of the accessible concepts. (see Fig 1) Upon the receipt of a XACML request for a set of concepts (1), the service determines all the contained concepts of the requested concepts (by concept containment) to get all the candidate accessible concepts. The XACML engine forwards such a request to the SPARQL generator (2) to convert it into SPARQL queries (3.a) using the requested concepts over the shared ontology represented as OWL triples (3.b). For instance, a registration request for the concept *WorkOrder* from a user with credential *Cred*1 is converted into the following SPARQL by the query generator:

```
PREFIX  po: <http://www.owl-ontologies.com/Ontology1223675912.owl#>
SELECT ?subClasses
WHERE { ?subClasses rdfs:subClassOf po:WorkOrder. }
```

The above SPARQL query returns all the subclass concepts of *WorkOrder* (4), i.e. *QualityInspection*, *Production*. If any of these result concepts also has subclass concepts then similar queries are performed recursively. To this end, multiple candidate concepts are determined while the initial request might only be for one concept. In case the user does not request for specific concepts then all the concepts in the ontology are candidate concepts to be evaluated further. In particular, a similar query should be performed starting from the most general concepts to determine all the concepts in the domain. In order to determine the authorized concepts for the requested user, the above query result (i.e. *QualityInspection*, *Production*) is then used into a further SPARQL query (5.a) which evaluates associated policy triples from all providers (5.b). The result of this query is the maximal set of aggregated concepts (possibly empty if none is permitted) that are accessible to the requester (6.a). The rule $\mathcal{O}_1$ of provider P_1 described at Section 3.2 allows the requester to access the subclass concepts. The following query is used to evaluate this rule:

```
PREFIX po: <http://www.owl-ontologies.com/Ontology1224765032.owl#>
SELECT  ?concept
WHERE{{?user po:hasCredential po:Cred1}{?user po:hasAccess ?concept.}}
```

The first WHERE clause determines the users with credential $Cred1$ and the second clause determines the accessible concepts for those users. If the result set contains the $QualityInspection$ and $Production$ concepts then the XACML engine returns a "Permit" response to the user (6.b,6.c). The XML content distributor in the system then extracts the *content signature*s of the authorized concepts by manipulating only the encrypted and encoded content for the requested user and sends those as a response to a successful registration (7). Otherwise, none of these concepts is accessible to the requester and the XACML engine simply denies access (6.c).

Selective XML Content Distribution. The XML content distributor sends the encrypted and encoded XML content to authorized users after identifying the appropriate XML content. This can be handled by two functions $auth\_list(U)$ and $distribution\_list(D)$. The former returns a maximal set of authorized concepts for a user U. The latter returns the set of concepts for which the mapped XML nodes are currently distributed by the distributor D. An encrypted XML content (i.e. $[C_i^x, E_p^x]$) for an authorized user contains node N_x, its content under encoding CE_x, and concept C_i in the *content signature*, i.e. C_i^x by definitions of Section 4.3. The selective delivery of XML content to an authorized user U proceeds as follows:

1. Separate allowed concepts: find all $C_i \in auth\_list(U) \in distribution\_list(D)$.
2. Determine allowed nodes: match concepts of $auth\_list(U)$ with encoded concepts in C_i^x
3. Extract associated encrypted and encoded XML nodes (i.e. $[C_i^x, E_p^x]$).
4. Finally, send user U the encoded and encrypted XML nodes extracted in step 2.

6 XML Content Verification

Upon receipt of encrypted and encoded XML nodes, an end user is able to perform a semantic verification followed by an EBOL-based verification. In the following, we use A_U to denote the list of *content signatures* and R_U the set of encrypted and encoded nodes received by the user U during registration and after delivery respectively. $\mathcal{N}_A$ and $\mathcal{N}_R$ denote the set of node identifiers in A_U and R_U respectively.

6.1 Semantic Verification

In order to detect any semantics-related authorization violation, the following verification steps must be performed.

1. (C-I) have all concepts been received?
2. (C-II) have all XML nodes from different documents been received?
3. (C-III) do the document nodes correspond to nodes mapped with a desired concept?

The user U verifies whether all the concepts of A_U it has access to are contained in R_U. The verification is as follows: $(\forall c \in A_U \exists r \in R_U \ni (N_R, C_i) = (N_A^{'}, C_i^{'}))$, where $(N_A^{'}, C_i^{'})$ is the *content signature*, if there is a concept in R_U with an identical concept, then all the authorized concepts have been received by *U (C-I verified)*.

U then verifies whether it has received all XML nodes from different documents. It checks a *belong-to* relation between all the document identifiers doc_{id} in the authorized node identifiers of A_U and the document identifiers $doc_{id}^{'}$ of the received node identifiers of R_U. This check is as follows: $(\forall n \in \mathcal{N}_A \exists r \in \mathcal{N}_R | (doc_{id} = doc_{id}^{'}))$; i.e. for each

node in $\mathcal{N}_A$, if there is an identical document identifier in $\mathcal{N}_R$, then all the nodes have been received by *U (C-II verified)*.

(C-III) can be verified by *C-I*. Let C_r be a received concept then a user verifies whether C_r belongs to A_U, that is $C_r \in A_U$. If this verification fails then the received concept C_r is not a desired one.

6.2 EBOL-based Verification

After a successful semantic verification, a user U can verify the following EBOL-based integrity violations:

1. (S-I) has the node content been changed?
2. (S-II) has some XML nodes not been received?
3. (S-III) have some nodes been moved?
4. (S-IV) has the node order been changed?

U decrypts the received XML nodes in $\mathcal{N}_R$ and traverses each document portion rooted at $r \in \mathcal{N}_R$ in breadth first order. Let x be the current visiting node. After decrypting an encoded node x gives the following encoded node:

$$[x, C_i^x, [CE_x, C_i^z]] = [x, < N_x, C_i >, [[< N_x, C_i >, I_x], C_i^z]]$$

U takes N_x from the outer C_i^x and x's content, Ct_x, and then computes the local hash of x as $I_x^{'} = H(N_x, Ct_x)$ which is then compared with I_x. If any mismatch is found, the node content has been changed *(S-I verified)*.

U further checks the *belong-to* relation between all node identifiers of A_U and the received node identifiers of R_U. This check is as follows: $(\forall a \in \mathcal{N}_A \; \exists r \in \mathcal{N}_R | (E_r, E^r_{Highest}) = (E_a^{'}, E^{'a}_{Highest}))$; i.e. for each node in $\mathcal{N}_A$, if there is an identical node identifier in $\mathcal{N}_R$, then all the nodes have been received by *U (S-II verified)*.

The verification process continues as the value of the node identifier N_x in the outer C_i^x must match with the inner node identifier N_x in CE_x. If not, then an integrity violation is detected and the node x can be discarded immediately without knowing the precise violation. To be precise, the elements of outer N_x are compared with the corresponding elements of the inner one. (a) if $f_{order}(E_x) \neq f_{order}(E_x^{'})$, where $E_x^{'}$ is in the inner N_x this means an order change is detected. (b) if $f_{level}(E_x) = f_{level}(E_x^{'})$ then the depth level of x in outer N_x is compared with the depth level of the received node in the inner C_i^x. If they do not match then the node x is moved to another depth level *(S-III semi verified)*.

The success of previous element wise matching does not guarantee a full integrity check. The depth level of the outer N_x must be compared with the depth level of the parent z of x in the inner C_i^z. If the latter is not less than the former then the node x is moved *(S-III fully verified)*.

During the breadth-first order traversal for a current node x, an order of EBOL E_x smaller than that of any previously visited node detects to an integrity violation. No such detection ensures that no order change was performed in a set of received XML nodes *(S-IV verified)*.

7 Related Work

There has been remarkable progress in recent years regarding access control to XML data structures in a client/server paradigm [6, 7, 8, 9, 16, 17, 18]. In these approaches,

the server enforces access control policies on a per request basis. Instead, our work focuses on delegating third parties the selective delivery of semantically equivalent content to authorized users independently of providers. The work of [14, 15] focuses on the delivery of encrypted XML data: authorization policies are specified based on the XML hierarchical structure yet document parsing is in post order. Our approach is fundamentally different as policy specification is assumed to be on domain concepts and selective delivery is performed based on the semantics captured in concepts, not document structure. Moreover, the EBOL computation can be performed on the fly while parsing documents. Our previous work [20] focuses on enabling authorized users to exchange document portions using a group key based approach that allows users with similar interests to be independent of a central authority, although it does not address document semantics.

[12] and [19] propose an ontology based access control for XML documents having variant schemas and semantically related documents respectively. However, none of them considers issues related to dissemination of semantically related data or document integrity and confidentiality. [10] discusses two ways ontologies can make it possible to describe access control models, but in that case focusing on different features of RBAC models. Although this work aims at modelling access control in a generic manner using Semantic Web methods in much the same way as our work, it does not specifically address the protection of XML schemas defining the resources accessed nor the practical implementation of enforcement. [21] introduces a formal model for semantic access control and associated algorithms which can be used in conjunction with our mechanism to detect if two providers defined conflicting access control policies on documents they distribute.

8 Conclusion and Future Work

This paper described an ontology-based XML content distribution system. Our solution protects the confidentiality of the document content and structure to protect the information system structure from other organizations and its content from unauthorized users. Document nodes are tagged with their semantic description and also incorporate integrity protection measures. Access control enforcement relies on a middleware that makes use of the semantic tagging of each document node which our EBOL scheme renders readable even for parties which cannot decrypt nodes. Semantic tagging can be efficiently analyzed, even for large documents, because of the breadth-first order parsing scheme adopted.

Our solution also illustrates in what respect semantic access control makes document exchanges feasible across organizational boundaries while protecting the layout of an organization's information system. We described how access control enforcement might be implemented by combining a XACML engine with a SPARQL engine. The use of ontologies allows us not only to reason about document authorizations, but also on the access control model: alternative paradigms, like the separation of concerns, might be introduced as inference rules on the access control ontology. Future work will investigate the implementation of such evolvable policies with Semantic Web methods.

Acknowledgment: This work is partly sponsored by EU IST-2004-026650 project R4eGov. Thanks to Henrik, Lim, Smriti and Slim for their valuable comments.

References

1. Joseki - A SPARQL Server for Jena, http://www.joseki.org/.
2. OWL Web Ontology Language Overview, http://www.w3.org/tr/owl-features/.
3. Resource Description Framework (RDF), http://www.w3.org/rdf/.
4. SPARQL Query Language for RDF, http://www.w3.org/tr/rdf-sparql-query/.
5. R. Agrawal, J. Kiernan, R. Srikant, and Y. Xu. Order preserving encryption for numeric data. In *SIGMOD '04: Proceedings of the 2004 ACM SIGMOD international conference on Management of data*, pages 563–574, New York, NY, USA, 2004. ACM.
6. W.-C. L. Bo Luo, Dongwon Lee and P. Liu. *A Flexible Framework for Architecting XML Access Control Enforcement Mechanisms*, volume Volume 3178/2004 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg, December 2004.
7. E. Damiani, S. D. C. di Vimercati, S. Paraboschi, and P. Samarati. Fine Grained Access Control for Soap E-services. In *WWW '01: Proceedings of the 10th international conference on World Wide Web*, pages 504–513, New York, NY, USA, 2001. ACM.
8. E. Damiani, S. D. C. di Vimercati, S. Paraboschi, and P. Samarati. A Fine-grained Access Control System for XML Documents. *ACM Trans. Inf. Syst. Secur.*, 5(2):169–202, 2002.
9. W. Fan, C.-Y. Chan, and M. Garofalakis. Secure XML Querying With Security Views. In *SIGMOD '04: Proceedings of the 2004 ACM SIGMOD international conference on Management of data*, pages 587–598, New York, NY, USA, 2004. ACM Press.
10. T. Finin, A. Joshi, L. Kagal, J. Niu, R. Sandhu, W. Winsborough, and B. Thuraisingham. Rowlbac: Representing Role Based Access Control in OWL. In N. ACM, New York, editor, *Proceedings of the 13th ACM Symposium on Access Control Models and Technologies (SACMAT '08), Estes Park, CO, USA.*, pages 73–82, June 2008.
11. S. Godik and T. Moses. eXtensible Access Control Markup Language (XACML), version 1.0, OASIS Standard, 2003.
12. A. Jain, D. Wijesekera, A. Singhal, and B. Thuraisingham. Semantic-Aware Data Protection in Web Services, Proceedings of IEEE Workshop on Web Services Security held in Berkeley, CA, May 2006, 2006.
13. L. Kagal, M. Paolucci, N. Srinivasan, G. Denker, T. Finin, and K. Sycara. Authorization and privacy for semantic web services. *IEEE Intelligent Systems*, 19(4):50–56, 2004.
14. A. Kundu and E. Bertino. A new model for secure dissemination of xml content. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 38(3):292–301, May 2008.
15. A. Kundu and B. Elisa. Secure Dissemination of XML Content Using Structure-based Routing. In *EDOC '06: Proceedings of the 10th IEEE International Enterprise Distributed Object Computing Conference*, pages 153–164, Washington, DC, USA, 2006. IEEE Computer Society.
16. G. Kuper, F. Massacci, and N. Rassadko. Generalized XML Security Views. In *SACMAT '05: Proceedings of the tenth ACM symposium on Access control models and technologies*, pages 77–84, New York, NY, USA, 2005. ACM Press.
17. G. Miklau and D. Suciu. Controlling Access to Published Data Using Cryptography. In *VLDB*, pages 898–909, 2003.
18. M. Murata, A. Tozawa, M. Kudo, and S. Hada. XML Access Control Using Static Analysis. In *CCS '03: Proceedings of the 10th ACM conference on Computer and communications security*, pages 73–84, New York, NY, USA, 2003. ACM Press.
19. V. Parmar, H. Shi, and S.-S. Chen. XML Access Control for Semantically Related XML Documents. *System Sciences, 2003. Proceedings of the 36th Annual Hawaii International Conference on*, pages 10 pp.–, Jan. 2003.
20. M. A. Rahaman, Y. Roudier, and A. Schaad. Distributed Access Control for XML Document Centric Collaborations, The 12th IEEE International EDOC Conference. 2008.
21. M. I. Yage, M. del-mar Gallardo, and A. Maa. Semantic access control model: A formal specification, european symposium on research in computer security, esorics 2005 pp.24-43, lncs 3679, springer-verlag, milano, italy,. 2005.