International Journal of Computer Information Systems and Industrial Management Applications.
ISSN 2150-7988 Volume 6 (2014) pp. 560–570

Access Control Policies for Traceability Information Systems

Miguel L. Pardal[1], Mark Harrison[2], Sanjay Sarma[3], and José Alves Marques[1]

[1]Department of Computer Science and Engineering
Instituto Superior Técnico, Technical University of Lisbon, Portugal
miguel.pardal@ist.utl.pt, *jose.marques@link.pt*

[2]Auto-ID Labs, Institute for Manufacturing,
University of Cambridge, UK
mark.harrison@cantab.net

[3]Auto-ID Labs, Massachusetts Institute of Technology, USA
sesarma@mit.edu

Abstract: **Traceability information systems need to collect and process data from multiple companies across the supply chain and many of the business partners are not known in advance. This open-ended security is, in principle, a good match for a Service-Oriented Architecture (SOA) design and for the use of Web Services (WS) technologies because they implement flexible and inter-operable systems based on services. However there is a gap between the visibility restrictions and the way to express them using standard WS technologies.**

This paper describes Supply Chain Authorization (SCAz), an interface developed to define and enforce visibility restrictions – *access control policies* – for supply chain systems. Several implementations are presented and the trade-offs are discussed. The performance of SCAz is assessed in the setting of an externalized security architecture by comparing raw authorization implementations with their equivalents translated to the standard language eXtensible Access Control Markup Language (XACML).

The SCAz Chain-of-Trust Assertions (CTA) implementation is found to have similar performance to other approaches while allowing extensions such as delegated trust, transitive trust, conditional trust, and bulk trust.

Keywords: Web Services; Authorization; XACML; Performance; Supply Chain Traceability

I. Introduction

Service-Oriented Architecture (SOA) [1] and Web Services (WS) [2] were designed to allow easier inter-operation of heterogeneous systems and to satisfy the non-functional requirements needed by real business systems.

WS standards are defined by IETF, W3C and OASIS to address non-functional concerns such as security, transactions, and reliable messaging. In this paper, the WS acronym is used to refer to most variants of distributed communication using eXtensible Markup Language (XML) message formats and protocols, including SOAP and REST.

The complexity of WS standards raises performance concerns. A common criticism of WS technologies is that they are slow and difficult to use [3]. The performance overheads have been measured in practice by Juric at al. [4] and they are significant. For example, SOAP messages are, on average, 4.3 times larger than equivalent binary messages and response times are 9 times slower. The overheads are even more significant when using security mechanisms: messages are 27 times larger, and response times are 15 times slower.

Since WS imply performance overheads, they are better suited to handle use cases where there is an actual need for flexibility and inter-operability, like in *Supply Chain Management* (SCM) and *Traceability* systems. A specific example of a useful traceability system is a Pharmaceuticals Pedigree control system that is meant to prevent drug shortages and detect counterfeit drugs. Dynamic authorizations are needed in supply chain applications because there are multiple businesses involved, distributed across the world, and many of them without prior knowledge of the others.

In this paper, we investigate the performance of authorizations for traceability information systems. We describe the Supply Chain Authorization (SCAz) interface and its three implementations:

- Enumerated Access Control (EAC);
- Chain-of-Communication Tokens (CCT);
- Chain-of-Trust Assertions (CTA).

EAC and CCT follow more traditional approaches to access control – access-control lists and capabilities [5] – whereas CTA uses open-ended representation and is intended to be extensible.

The performance is assessed using measurements collected from prototype implementations assuming an externalized security architecture. To compare the overheads of WS authorizations, the raw implementations are compared with their equivalents translated to eXtensible Access Control Markup Language (XACML) [6]

After the assessment, the CTA implementation is further developed with more sophisticated sharing conditions.

A. Outline

In the next section, the traceability system architecture for the supply chain domain is described and the externalized security architecture is presented with focus on authorization. A short introduction of Linked Data is also provided. Next, the supply chain authorization implementations are presented along with their conversion procedures to XACML format. The performance assessment study results are presented and CTA implementation extensions are defined in detail. The paper ends with the conclusions and future work directions.

II. Traceability System

SCM solutions focus on planning and execution, integrated with companies' Enterprise Resources Planning (ERP) systems. Their goal is to improve the flows of physical objects so that goods travel, from producer to consumer, in the least amount of time, and at the lowest cost [7]. In this context, information is collected to allow traceability queries [8], such as: Track (*Where is a specific item?*); and Trace (*Where has a specific item been?*).

Goods are tagged with unique serial numbers encoded using printed two-dimensional bar-codes and/or Radio Frequency Identification (RFID tags). A widely used scheme to define the item-level identifiers is called the Electronic Product Code (EPC). Each EPC contains a serial number that uniquely identifies a physical object instance.

The EPC Network architecture [9] defines standard data capture and query interface for repositories of traceability data, called the EPC IS (Information Services) [10]. The architecture also proposes a service to facilitate data retrieval, called the EPC DS (Discovery Services) [11] [12].

Figure 1 depicts the EPC Network and how EPC data is captured. EPC IS store detailed events whereas EPC DS stores references to the locations of event data.

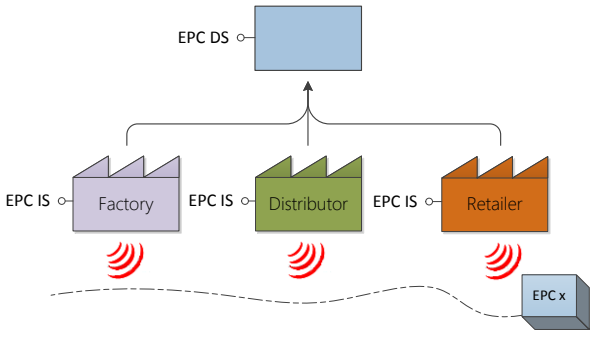

Figure. 1: EPC Network architecture with DS and IS.

The EPC Network is *centralized* and uses *meta-data* to index the detailed information. Currently, it is the most common traceability system approach [13].

Each supply chain participant can benefit from exchanging data with other companies, but information such as the levels of demand, inventory, and supplier identities should be kept to trusted business partners. Each company can define its own *trust circle*, like the one in Figure 2. In fact, there are studies that show that without clearly defined and enforced data access control, companies are not willing to share traceability data [14].

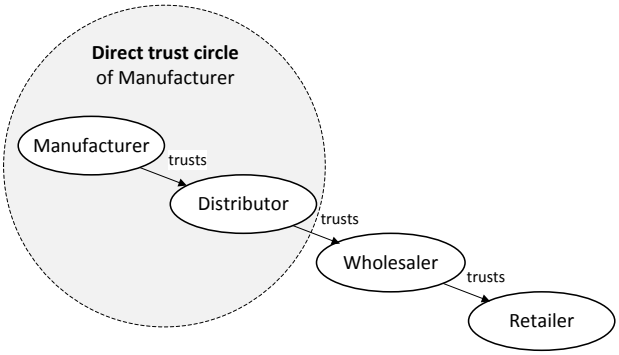

Figure. 2: Direct trust circle of a Manufacturer.

III. Externalized Security

The goal of the externalized security architecture is to unify the security management of different applications so that business rules can be changed dynamically. This contrasts with rigid security measures embedded in each application.

Externalized security encompasses user management, authentication, authorization, logging and auditing [15]. The security measures are summarized in Figure 3: authentication protects the *subject*; authorization and non-repudiation protect the *action*; finally, availability, integrity and confidentiality measures protect the *resource*.

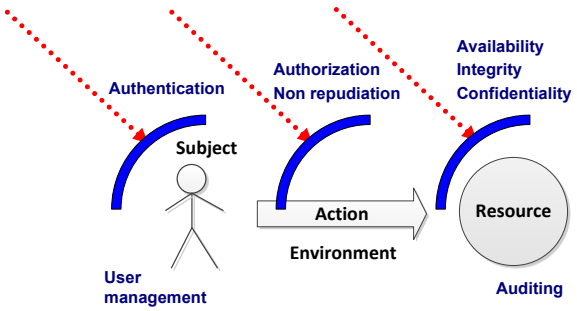

Figure. 3: Security measures.

The authentication of subjects can be achieved with identity providers [16] and the exchange of Security Assertion Markup Language (SAML) assertions [17].

The authorization of actions can be expressed with the eXtensible Access Control Markup Language (XACML) [6] that allows the combination of multiple, possibly conflicting, policies to define fine-grained access control.

A. eXtensible Access Control Markup Language

XACML [6] is an XML vocabulary to represent authorization policies and requests. In formal terms, an authorization is the verification of a subject's right to execute an action on a resource.

The standard defines a *policy* format and a *processing* model.

1) Policy

Figure 4 presents a simplified XACML Policy structure with target and rules. The *target* defines a set of 'match' conditions regarding the subject, action, resource and environmen-

t, that determine if the policy is relevant for the request. The *rules* are conditions that evaluate to 'Permit', 'Deny', 'NotApplicable' (when no target matches), or 'Indeterminate' (when an internal errors occur).

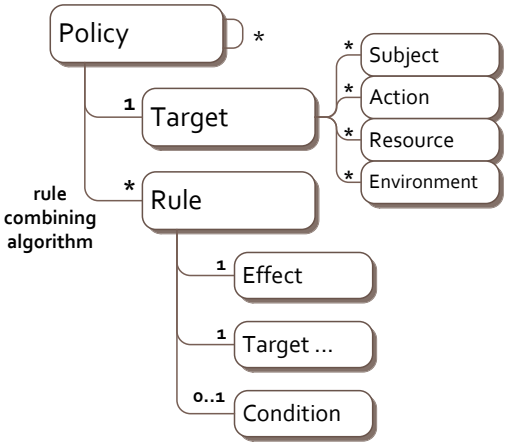

Figure. 4: XACML simplified Policy model.

XACML has a set of predefined functions that are used to transform and compare data attributes. The policy language is declarative and new function definitions are not allowed to keep the the policy processing simple i.e. with low algorithmic complexity [18]. Additional functions can be provided by XACML library implementations but, since they are not standard, they may not be available in other implementations.

2) Processing

The authorization architecture that XACML assumes is defined by RFC 2753 [19] and 2904 [20] and is represented in Figure 5.
The Policy Administration Point (PAP) is used to author and manage policies. The Policy Enforcement Point (PEP) is the component that intercepts requests to a resource, and can also perform 'before' or 'after' actions, called *obligations*. The PEP checks with the Policy Decision Point (PDP), that *evaluates* the request with respect to the applicable policies. If necessary, the Policy Information Point (PIP) retrieves additional attribute values for the PDP that are not contained in the request.

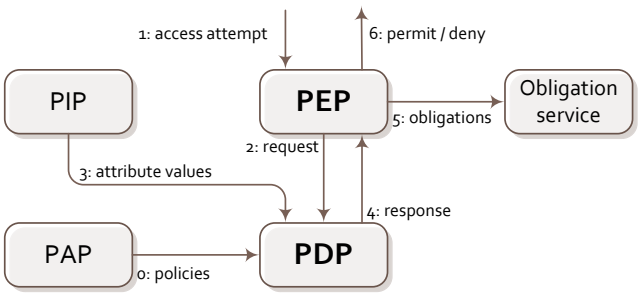

Figure. 5: XACML request processing.

3) Implementations

XACML is currently in version 3.0, with many deployments and industry support. The XACML open-source implementations were surveyed, as presented in Table 1.
The Sun Microsystems XACML library can be considered the reference implementation, but it has not been updated since 2004. Since the project was made publicly available, several "branches" were created by different developers. There is a 2.0 version under development but its sponsorship is not clear and it breaks source code compatibility.
PicketBox is supported by JBoss but the library is insufficiently documented at the present time.
HERAS-AF [21] is a well documented library that was developed in academia but is currently also used by industry. HERAS-AF stands for Holistic Enterprise-Ready Application Security Architecture Framework. The open-source version is restricted to in-memory policy repository and the project has been inactive since 2010.
'enterprise-java-xacml' has the best reported performance [22] but has little documentation.
XEngine [23] also claims to have the best performance but there are few code examples available and it is insufficiently documented and currently inactive.

Impl.	Version	Last update	Sponsors
sunxacml	1.2	Jul 2004	Sun Microsystems (currently Oracle)
Heras-AF	1.0.0-M2	Sep 2010	U. Applied Sciences Rapperswil, Switzerland
enterprise -java-xacml	r258	Jan 2009	Zian Wang
PicketBox XACML	2.0.9.Final	June 2013	JBoss, Red Hat
xEngine	beta 0.2	Aug 2010	Michigan State U., North Carolina State U.

Table 1: Open-source XACML implementations.

IV. Related work

A. Traceability

WS technologies are already widely used in traceability system implementations. The most prominent example is Fosstrak [24] that provides a SOAP endpoint to the EPC IS event repository and an alternative REST-based interface [25].

B. Linked Data

Linked Data can be used to represent data access policies. It is the underlying distributed data model of the Semantic Web.
The Semantic Web [26] is intended to be an organized worldwide system where information flows in a smooth but orderly way. The **AAA** slogan – **A**nyone can say **A**nything about **A**ny topic – is intrinsic to the design of the Semantic Web. This means that, in practice, there can be many contradictions and inconsistencies in the data that make it hard to use effectively. As a result, different sources, organizations, and styles of information need to co-exist, and semantic modeling tools are required to build models that make data usable and useful in this context.
At the core of the distributed data model is the Resource Description Framework (RDF) [27] that is a universal data

model that represents data structures as triples. Each triple – subject, predicate, and object – can be represented in graph format. For instance, the triple ':company0 cta:publishes :record0' is represented in Figure 6.

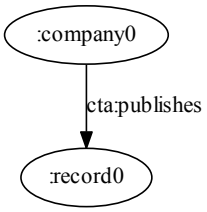

Figure. 6: RDF triple represented as a graph.

The SPARQL Protocol and RDF Query Language (SPARQL) [28] is a query language that partly resembles the Structured Query Language (SQL) widely used in relational databases. SPARQL is dedicated to find matches for RDF statements that may contain variables, rather than extracting values from specific columns of specific tables for records that match particular criteria.
The typical Linked Data application architecture, depicted in Figure 7, extends a database application architecture. The database – RDF store – merges the information and applies models to infer new data and validate conditions. Data input converters transform data from other formats – web pages, spreadsheets, tables, databases – to RDF. The query engine is implemented using SPARQL. The application interface uses the content of an RDF store in interactions with end-users.

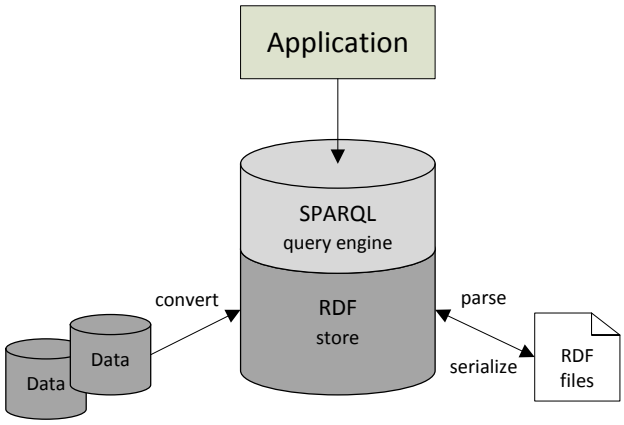

Figure. 7: Linked Data application architecture, adapted from [26].

Linked Data technologies – RDF, OWL and SPARQL – provide open-ended data representation and querying, and there is significant work where they are used for security.
Cadenhead et al. [29] describes a general-purpose, scalable RDF policy engine that includes support for a diverse set of security policies. The policy engine was evaluated as being highly available and scalable.
Finin et al. [30] show different ways to support the NIST standard Role-Based Access Control (RBAC) model in Web Ontology Language (OWL) and then discuss how the OWL constructions can be extended to model attribute-based RBAC or more generally attribute-based access control.
Ferrini et al. [31] start from XACML that does not natively support RBAC and introduce XACML+OWL, a framework that integrates OWL ontologies and XACML policies for supporting RBAC. It decouples the design by modeling the role hierarchy and the constraints with an OWL ontology and the authorization policies with XACML.
Papakonstantinou et al. [32] use 'quadruples' to encode access labels for RDF triples, representing information such as time, trust, and provenance. The authors make use of the SPARQL language to determine the triples that define these labels.
The OWL versus SPARQL inference approaches bring back the discussion of declarative versus procedural knowledge representation [33]. Declarative knowledge is "knowing that" and is expressed by OWL statements whereas procedural knowledge is "knowing how" and is expressed by SPARQL procedures.

V. Supply Chain Authorization

Access control for supply chains is different from traditional authorization because, in most cases, the path that each physical object path will follow is not known in advance. This fact prevents the prior knowledge about who should be authorized to access the data [34]. Also, since there are a large number of items, it is inconvenient to define policies for each one.
The Supply Chain Authorization (SCAz) Application Programming Interface (API) has been proposed recently [35] [36] to allow companies participating in a supply chain to express their authorization rules using business concepts such as item and company.
A *traceability data set* contains events owned by a company about a specific item. Each policy protects a data set.
There are three alternative implementations for SCAz. Figure 8 shows how the classes relate to the interfaces: each class implements a mechanism-specific API and both implement the SCAz interface, allowing the same authorization needs to be mapped transparently to distinct implementations.

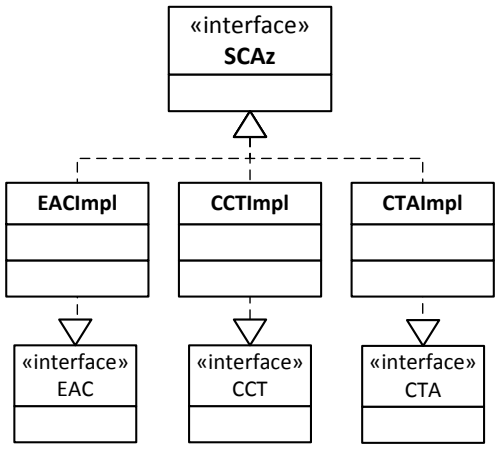

Figure. 8: SCAz, EAC, CCT, and CTA interfaces and classes.

A. Enumerated Access Control

Enumerated Access Control (EAC) is based on Access-Control Lists (ACLs) [5]. Each ACL (see Figure 9) has an owner and several permissions that define authorized user-action pairs that keep the access rights indexed by the object identifier.

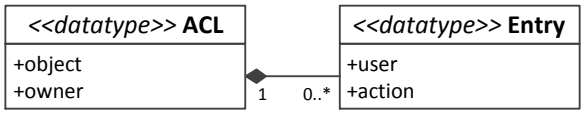

Figure. 9: ACL data structure.

Each traceability data set is protected by an ACL. The master ACL is maintained at the DS. A local copy is also maintained at each IS. The data owner contacts DS to add new partners to the ACL and the changes are propagated to the IS.
EAC uses the ACL implementation available in the Java virtual machine (package `sun.security.acl`) and Figure 10 shows the available ACL operations.

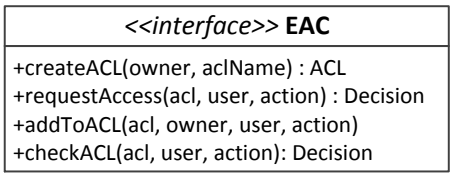

Figure. 10: EAC interface operations.

B. Chain-of-Communication Tokens

Chain-of-Communication Tokens (CCT) is an adapted Capability [5] mechanism. CCT represents access rights within object references called tokens. Figure 11 presents the contents of a token: there is a public identifier and a secret that is used as a 'password'. The remaining data – owner, resource, and action – identify the purpose of the token: "the *owner* grants the right to perform the *action* on the *resource* to the token holder".

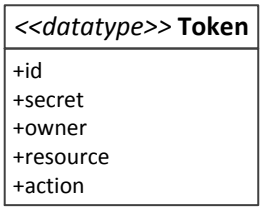

Figure. 11: Authorization Token data structure.

A token protects a traceability data set. Figure 12 shows the token operations. New visibility scopes can be created by issuing a new token for the object. The data owner sends the token directly or via DS to its partners to authorize them.

C. Chain-of-Trust Assertions

Chain-of-Trust Assertions (CTA) access rights are expressed using logical statements, called *assertions*, issued by the multiple parties. Figure 13 shows the assertion operations.

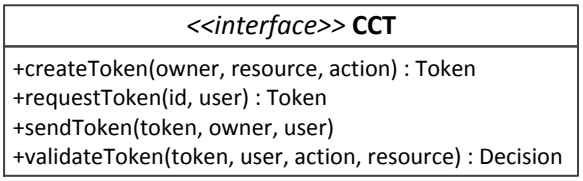

Figure. 12: CCT interface operations.

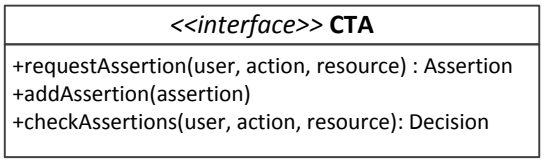

Figure. 13: CTA interface operations.

CTA is different from EAC and CCT because its semantics can be extended. CTA is implemented using *Linked Data* technologies. RDF and SPARQL are applied to construct fine-grained access control policies and evaluate them, respectively.
CTA policies are represented as RDF statements and stored in the RDF store. Figures 14 and 15 show a simple CTA policy stated in RDF classes and properties, expressed as subject-predicate-object tuples.

```
:company0  a  cta:Organization .
:company1  a  cta:Organization .

:item0     a  cta:Identifier .
:record0   a  cta:Record .
:policy0   a  cta:Policy .

:company0  cta:publishes   :record0 .
:record0   cta:about       :item0 .

:company0  cta:creates     :policy0 .
:policy0   cta:protects    :item0 .
:policy0   cta:grantsRead  :company0 .
:policy0   cta:grantsRead  :company1 .
```

Figure. 14: CTA Policy in RDF Turtle format.

In the example, 'policy0' created by 'company0' – the data owner – grants read access to 'record0' about 'item0' to 'company0' and 'company1' – the partner. The extensibility can be achieved by adding new properties e.g. `cta:grantsWrite` to grant *write* access.
The assertions are sent to the policy evaluation authority. Given a request, access is granted if there is an unbroken chain of trust assertions leading back to the owner of the data. Assertions can later be revoked, if necessary.
CTA was implemented using the the Apache Jena RDF library (available at `http://jena.apache.org/`) and HERAS-AF. Both were selected because of the available documentation and open access to the source code.

D. Conversion to XACML

SCAz represents data visibility permissions using supply chain business concepts. The policies are then translated to XACML so they can be deployed and enforced in a stan-

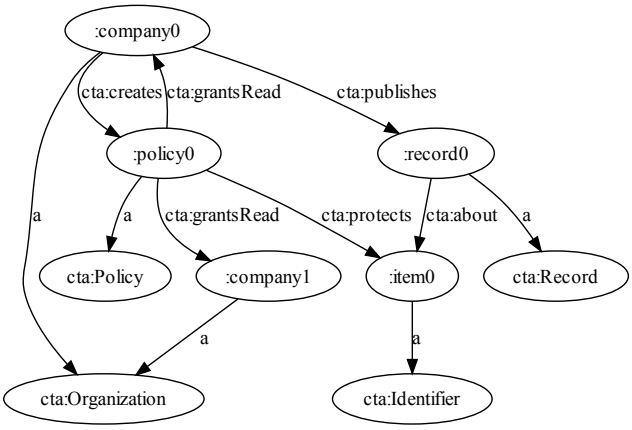

Figure. 15: CTA Policy graph.

dard security infrastructure, leveraging existing libraries and tools. This conversion approach is based on the policy conversions by Karjoth et al. [37].
Figure 16 depicts the EPC Network including the access control policies. There is one PEP and PDP for each EPC IS and DS. The policies should be authored by the data owner and then distributed to both DS and IS, to be used for enforcement.

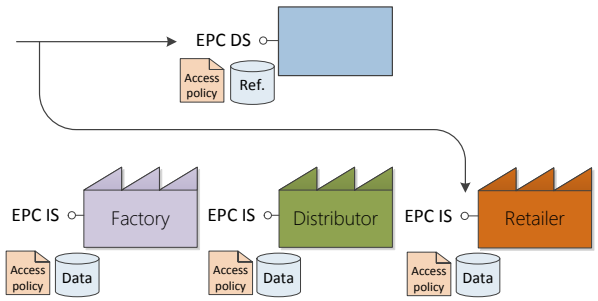

Figure. 16: EPC Network architecture with DS and IS and security policies.

For the authorizations in the traceability system, XACML is used as the canonical format for representing data access policies, both at the EPC IS and DS levels, as depicted in Figure 16.

1) Conversion from EAC

Each item data set has a single ACL, and that ACL corresponds to a single XACML policy. The item identifier is used as policy name and as target. The rule combining algorithm is 'first-applicable' meaning that the outcome of the first matched rule will be the access decision. For each entry in the source ACL, a 'Permit' rule is generated for per subject-action pair. Also a 'Deny' rule is generated. Lastly, there is a "Deny all" rule for all other attempts.

2) From CCT

Each token corresponds to one XACML policy that expresses its permissions. The token identifier is used as policy name and as target. For each capability encoded in the token there is a 'Permit' rule that checks if the action-resource pair and the secret are correct. In the end there is a "catch" rule to deny access for all other attempts using the token.

3) From CTA

To convert a CTA policy to a XACML policy the RDF statements are navigated as follows: first the objects of "*pol* protects *id*" statements are found. For each item identifier, a XACML policy is created. The policy target matches the item identifier. A permit rule is created to grant each access right – e.g. read – to the objects from "*pol* grantsRead *org*" statements. A final catch rule is created so that all other requests regarding the item are also denied. The rule combining algorithm is 'first-applicable' so the outcome of the matched first rule is the access decision.

VI. Performance assessment

The aim of the performance assessment was to evaluate if the performance of the solution is suitable for large supply chains.
The SCAz assessment tool [35] was developed to test both the raw EAC, CCT and CTA mechanisms and from their XACML-equivalent forms.
The test machine was a Quad-core CPU[1] at 2.50 GHz, with 3.25 GB of usable RAM, and 1 TiB hard disk; running 32-bit Windows 7 (version 6.1.7601), and Java 1.7.0_04.
The policies were defined using the common SCAz API to allow the same business needs to be represented internally by each implementation. The policies were then converted to XACML format and tested using the HERAS-AF XACML implementation to assess the correctness and the performance.

A. Evaluation

The data sharing policies were correctly translated and enforced. The presented performance figures are the result of the average of repeated runs of the same experiments.
The number of items considered in the experiments were based on information collected by Ilic et al. [38], ranging from 10^2 to 10^4.

1) Raw

Figure 17 presents a plot of the request evaluation time for increasing number of policies protecting traceability data sets. EAC and CTA handle the loads much better with results below 0.1 ms. Performance of CCT implemented with custom code is clearly the worse because CCT needs to perform searches in token collections for each request while the other two receive all the values as arguments.

2) XACML

Figure 18 presents a plot of the 'request evaluation' time, again for increasing number of policies converted to XACML format.
The performance of CCT and CTA are the best, EAC is worse, but on the same order of magnitude.

[1] Intel Core 2 Quad Central Processing Unit Q8300

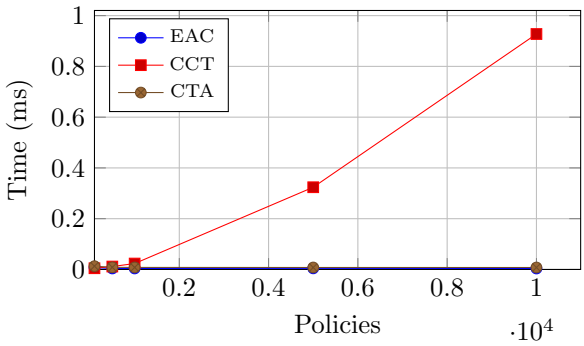

Figure. 17: Raw evaluation time with increasing number of policies.

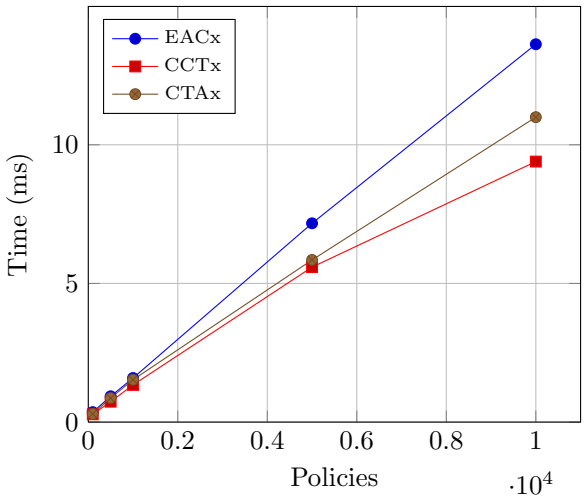

Figure. 18: XACML evaluation time with increasing number of policies.

3) Raw vs. XACML

Comparing the y-axis of Figures 17 and 18 it is visible that the XACML performance overheads are very significant. Table 2 presents each XACML time divided by the corresponding raw time for chains handling increasing numbers of items protected by policies. An average 400-fold overhead can be observed. Also, the performance overhead increases with the number of deployed policies.

Nr. Policies	EAC	CCT	CTA
$0.01 \cdot 10^4$	49.9	58.9	22.7
$0.05 \cdot 10^4$	240.0	63.9	103.5
$0.1 \cdot 10^4$	406.7	57.5	191.4
$0.5 \cdot 10^4$	1670.1	17.3	752.9
$1 \cdot 10^4$	3684.4	10.1	1429.2

Table 2: XACML overhead with increasing number of item policies.

VII. Policy Building Blocks

The SCAz Chain-of-Trust Assertions (CTA) implementation was found to be the most expressive and with similar performance to the "classic" approaches based on access-control lists and capabilities [5].
The representation using Linked Data technologies allows new statements to be added. Instead of issuing plain trust assertions, parties can issue *conditional* assertions, like reciprocal trust ("I trust you if you trust me"). Additionally, special predicates can be designed to express *dynamic* chain upstream/downstream conditions, allowing data sharing or, at least, initial data discovery between parties that did not have previous interactions. It can also support the *delegation* of administrative rights from one organization to another. These constructs can also be combined to express rich data sharing conditions.

A. Delegated Trust

For parties whom the creator of the data record does know directly, a possible solution is for the creator of the data record to grant explicit read access rights to parties that are known and trusted and to give *delegation rights* to some of these parties, to allow them to grant further access rights to additional parties that they know and trust.
The extension to add support for the *delegation* of administrative rights from one organization to another is shown in Figures 19 and 20: 'company0' delegates the ability to grant access to 'company1', and 'company1' grants read access to 'company2'. 'company0' does not have to know 'company2'. Access is granted if there is an explicit unbroken chain of trust assertions leading back to the owner of the data.
To verify the delegated trust, two SPARQL *construct* statements are used. The first statement is used to compute the delegation path. It checks if the 'cta:delegates' predicates chain forward. The second statement verifies if the read access is granted by anyone in the trust chain.

1) Sharing downstream

Consider the supply chain in Figure 2. The Manufacturer creates a data record about a specific product instance but only

```
:company0  cta:publishes   :record0 .
:record0   cta:about       :item0 .

:company0  cta:creates     :policy0 .
:policy0   cta:protects    :item0 .
:policy0   cta:delegates   :company1 .
:policy0   cta:grantsRead  :company2 .
```

Figure. 19: CTA delegation extension (type definition predicates omitted).

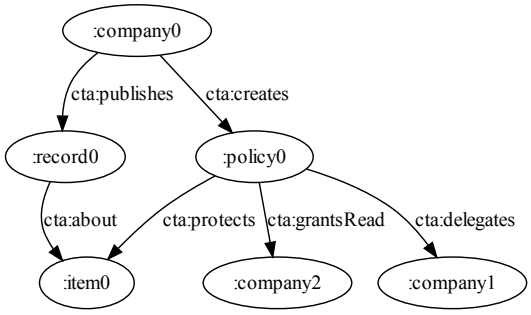

Figure. 20: CTA delegation extension graph.

wants to share it with the companies who were on the specific chain of custody (unique supply chain path) that the specific product instance took. The problem is that the Manufacturer only knows which Distributor the product is shipped to, and has no visibility beyond that point. By creating a policy that delegates rights to the Distributor to create additional policies, the Manufacturer can choose to trust the Distributor to pass on the appropriate access control privileges to the appropriate downstream parties, eventually reaching the Retailer who actually sold that specific product instance corresponding to the data in record.

2) Sharing upstream

The Retailer wants to share some information with upstream parties. It publishes a record and creates a policy that delegates rights to an upstream party instead of a downstream party. The process continues upstream, until the policy created by Distributor grants read access to the Manufacturer.
Parties in the middle of the supply chain, such as distributors and wholesalers might also publish records and create policies that delegate rights to parties further upstream (their suppliers) or further downstream (their customers).

B. Transitive trust

Trust for data regarding a specific item sometimes needs to express dynamic chain upstream/downstream conditions, allowing data sharing between parties that did not have previous interactions.
The *transitive* predicates are represented in Figure 21. By issuing the 'cta:trustChain' predicate, 'company0' allows 'company1' to access 'record0' about 'item0' because it published 'record1' about the same item.

```
:company0  cta:publishes   :record0 .
:record0   cta:about       :item0 .

:company0  cta:creates     :policy0 .
:policy0   cta:protects    :item0 .
:policy0   cta:trustChain  :item0.

:company1  cta:publishes   :record1 .
:record1   cta:about       :item0 .
```

Figure. 21: CTA chain trust transitivity extension (type definitions omitted).

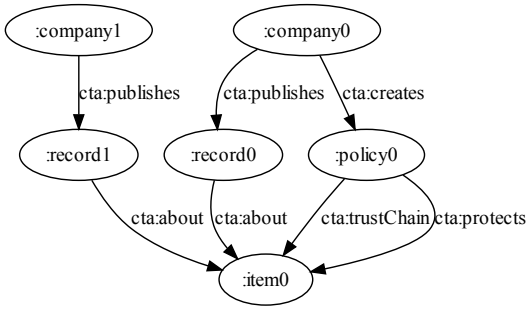

Figure. 22: CTA chain trust transitivity extension graph.

C. Conditional Trust

Trading partners can issue *conditional* assertions, like reciprocal trust: "I trust you if you trust me". The reciprocal trust predicates are represented in Figures 23 and 24. The 'cta:grantsReadRecipr' issued by 'company0' is only effective if a similar predicate is issued granting conditional access to records about the same item.

```
:company0  cta:publishes          :record0 .
:record0   cta:about              :item0 .

:company0  cta:creates            :policy0 .
:policy0   cta:protects           :item0 .
:policy0   cta:grantsReadRecipr   :company1.

:company1  cta:creates            :policy1 .
:policy1   cta:protects           :item0 .
:policy1   cta:grantsReadRecipr   :company0.
```

Figure. 23: CTA reciprocal trust extension (type definitions omitted).

D. Bulk trust

So far the data sharing policies have addressed individual items and individual companies. However, considerable efficiencies can be obtained by representing object groupings (lots) and company sets (groups).
There are three ways of modelling relationships with cardinality greater than one in RDF. The first, and simplest, is to define multiple values for a predicate. The second uses 'head' and 'rest' predicates to create a linked list and is intended for closed, ordered collections. The third uses types

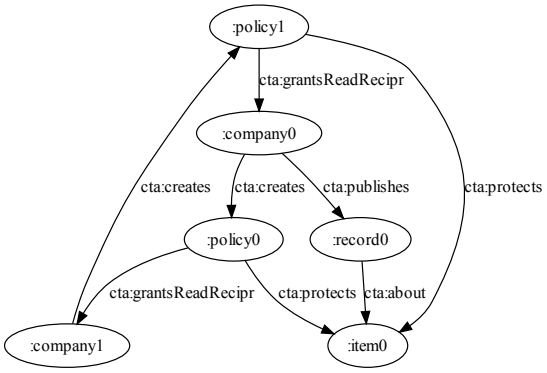

Figure. 24: CTA reciprocal trust extension graph.

and special ordinal predicates to define the items that belong to the collection. There are ordered and unordered collections, called *Sequence* and *Bag*, respectively.
Figures 25 and 26 represent 'lot0' that contains three items and 'group0' that contains two companies. For the *product lot*, multiple predicate values were used because it is a simpler, less verbose approach that is suitable for relationships without attributes. The lot object can be further characterized, with predicates for it. For the *trading partner group*, a 'Bag' was used because it provides an identity to the collection and allows further characterization of the relationship.
Also the cardinality of the relationship is expected to be much smaller than the lots that can reach thousands of items.

```
:lot0   cta:inLot  :item0.
:lot0   cta:inLot  :item1.
:lot0   cta:inLot  :item2.

:group0 cta:group [
    a       rdf:Bag;
    rdf:_1  :company0;
    rdf:_2  :company1
].
```

Figure. 25: CTA bulk trust. Product lot and company group.

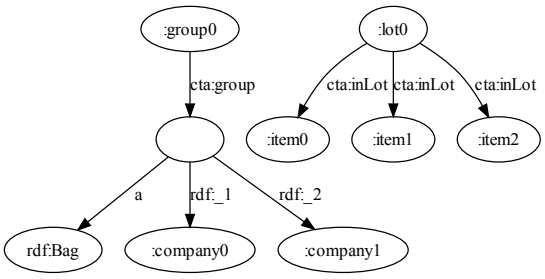

Figure. 26: CTA bulk trust graph.

VIII. Conclusion

Our results for access control agree with previous results [4] [39] that show that WS technologies overheads are very significant. The performance of the raw implementations compared with the XACML-converted policies shows that XACML overheads are very significant: 400 times on average, and over 1000 times in the worst case.
There is a clear need to simplify and streamline the standards and there is room for performance improvements in the implementations. However, despite these drawbacks, WS technologies are widely used, meaning that their advantages like inter-operability, flexibility, and tool support; add value for developers and make up for the additional cost. That is also the case for using XACML to protect supply chain data. XACML allows the policies to be exchanged in a standard format that can be properly interpreted in all the policy enforcement points and allows the use of other tools that comply with the standard (e.g. auditing tools).
The contribution of this work is the detailed description of the SCAz API and its implementations: EAC based on ACLs, CCT based on capabilities, and CTA based on Linked Data technologies; and the *performance assessment* of the enforcement of supply chain data visibility policies, with an externalized security architecture. CTA has extensible semantics while EAC and CCT have predefined semantics and, because of this, CTA was expected to be slower. However, in the assessment, all three mechanisms express exactly the same visibility restrictions and there are no significant performance differences for evaluation of requests, and CTA is even better than EAC. Considering that the performance is similar and that it is extensible, in future work CTA should be the preferred choice for expressing supply chain authorizations.

A. Future work

CTA will be used to represent advanced conditions that are required for specific business cases – Pharmaceuticals pedigrees, for instance – and the new policies will be converted to XACML and enforced in the context of the EPC Network.
There are indications that the performance overheads reported using the HERAS-AF library could be significantly lowered with more optimized alternatives [40] [23]. This proposition needs to be measured in practice before more general conclusions can be drawn.
The evaluation job execution will be improved to measure the performance of XACML with more than 10^4 items [38].
The authorization challenges of externalized security in the supply chain traceability system illustrated how WS can assist in complex and dynamic business environments involving multiple organizations. Much more research can be done in SCM and Traceability systems, to explore the advanced capabilities of Web Services.

Acknowledgment

Miguel L. Pardal is supported by a PhD fellowship from the Portuguese Foundation for Science and Technology FCT (SFRH/BD/45289/2008).

References

[1] T. Erl, *SOA Principles of Service Design*. Prentice Hall, July 2007.

[2] G. Alonso, F. Casati, H. Kuno, and V. Machiraju, *Web Services: Concepts, Architectures and Applications*. Springer Verlag, 2004.

[3] J. Newmarch, "A critique of web services," *IADIS E-Commerce*, 2004.

[4] M. B. Juric, I. Rozman, B. Brumen, M. Colnaric, and M. Hericko, "Comparison of performance of Web Services, WS-Security, RMI, and RMI-SSL," *Journal of Systems and Software*, vol. 79, no. 5, pp. 689–700, 2006.

[5] R. Sandhu and P. Samarati, "Access control: principle and practice," *IEEE Comm. Magazine*, vol. 32, no. 9, pp. 40–48, September 1994.

[6] B. Parducci, H. Lockhart, and E. Rissanen, *eXtensible Access Control Markup Language (XACML) Version 3.0*, OASIS Std., January 2013.

[7] K. Laudon and J. Laudon, *Management Information Systems - 12th edition*. Prentice Hall, January 2011.

[8] R. Agrawal, A. Cheung, K. Kailing, and S. Schonauer, "Towards Traceability across Sovereign, Distributed RFID Databases," in *Int'l Database Engineering and Applications Symp. (IDEAS)*, 2006.

[9] F. Thiesse, C. Floerkemeier, M. Harrison, F. Michahelles, and C. Roduner, "Technology, Standards, and Real-World Deployments of the EPC Network," *IEEE Internet Computing*, vol. 13, no. 2, pp. 36–43, 2009.

[10] EPCglobal, *EPC Information Services (EPCIS) 1.0.1 Specification*, GS1 Std., September 2007.

[11] J. J. Cantero, M. A. Guijarro, A. Plaza, G. Arrebola, and J. Baos, "A Design for Secure Discovery Services in the EPCglobal Architecture," in *Unique Radio Innovation for the 21st Century*, D. C. C. Ranasinghe, Q. Z. Z. Sheng, and S. Zeadally, Eds. Springer Berlin Heidelberg, 2010, pp. 183–201.

[12] C. Kürschner, C. Condea, O. Kasten, and F. Thiesse, "Discovery Service Design in the EPCglobal Network, Towards Full Supply Chain Visibility," *Internet of Things*, pp. 19–34, 2008.

[13] M. L. Pardal and J. A. Marques, "Cost Model for RFID-based Traceability Information Systems," in *IEEE Int'l Conf. on RFID Technology and Applications*, September 2011.

[14] M. Eurich, N. Oertel, and R. Boutellier, "The impact of perceived privacy risks on organizations' willingness to share item-level event data across the supply chain," *Journal of Electronic Commerce Research*, vol. 10, no. 3-4, pp. 423–440, December 2010.

[15] R. N. Hebig, C. Meinel, M. Menzel, I. Thomas, and R. Warschofsky, "A Web Service Architecture for Decentralised Identity- and Attribute-Based Access Control," in *Proc. IEEE Int'l Conf. Web Services (ICWS)*, 2009, pp. 551–558.

[16] D. Baier, V. Bertocci, K. Brown, E. Pace, and M. Woloski, *A Guide to Claims-Based Identity and Access Control*. Microsoft, 2010.

[17] S. Cantor, J. Kemp, R. Philpott, and E. Maler, *Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0*, OASIS Std., March 2005.

[18] C. Alm and R. Illig, "Translating High-Level Authorization Constraints to XACML," in *Proc. 6th World Congress Services (SERVICES-1)*, 2010, pp. 629–636.

[19] R. Yavatkar, D. Pendarakis, and R. Guerin, *RFC 2753 – A Framework for Policy-based Admission Control*, IEFT, Internet Engineering Task Force Std., January 2000. [Online]. Available: http://www.ietf.org/rfc/rfc2753.txt

[20] J. Vollbrecht, P. Calhoun, S. Farrell, L. Gommans, G. Gross, B. de Bruijn, C. de Laat, M. Holdrege, and D. Spence, *RFC 2904 – AAA Authorization Framework*, IEFT, Internet Engineering Task Force Std., August 2000. [Online]. Available: http://www.ietf.org/rfc/rfc2904.txt

[21] F. Huonder, "Conflict Detection and Resolution of X-ACML Policies," Master's thesis, University of Applied Sciences Rapperswil, July 2010.

[22] F. Turkmen and B. Crispo, "Performance evaluation of XACML PDP implementations," in *Proc. of the 2008 ACM Workshop on Secure Web Services*, ser. SWS. New York, NY, USA: ACM, 2008, pp. 37–44.

[23] A. Liu, F. Chen, J. Hwang, and T. Xie, "Designing Fast and Scalable XACML Policy Evaluation Engines," *IEEE Transactions on Computers*, no. 99, 2010.

[24] C. Floerkemeier, C. Roduner, and M. Lampe, "RFID Application Development with the Accada Middleware Platform," *IEEE Systems Journal, Special Issue on RFID Technology*, December 2007.

[25] D. Guinard, C. Floerkemeier, and S. Sarma, "Cloud computing, REST and Mashups to simplify RFID application development and deployment," in *Proc. of the 2nd Int'l Workshop on Web of Things (WoT)*. ACM, 2011, pp. 9:1–9:6.

[26] D. Allemang and J. Hendler, *Semantic Web for the Working Ontologist, Second Edition: Effective Modeling in RDFS and OWL*, 2nd ed. Morgan Kaufmann, June 2011.

[27] F. Manola and E. Miller, *RDF Primer*, W3C Std. [Online]. Available: http://www.w3.org/TR/2004/REC-rdf-primer-20040210/

[28] S. Harris and A. Seaborne, *SPARQL 1.1 Query Language*, W3C Std. [Online]. Available: http://www.w3.org/TR/2012/PR-sparql11-query-20121108/

[29] T. Cadenhead, V. Khadilkar, M. Kantarcioglu, and B. Thuraisingham, "A cloud-based rdf policy engine for assured information sharing," in *Proceedings of the 17th ACM symposium on Access Control Models and Technologies*, ser. SACMAT '12. New York, NY, USA: ACM, 2012, pp. 113–116. [Online]. Available: http://doi.acm.org/10.1145/2295136.2295157

[30] T. Finin, A. Joshi, L. Kagal, J. Niu, R. Sandhu, W. Winsborough, and B. Thuraisingham, "Rowlbac: representing role based access control in owl," in *Proceedings of the 13th ACM symposium on Access control models and technologies*, ser. SACMAT '08. New York, NY, USA: ACM, 2008, pp. 73–82. [Online]. Available: http://doi.acm.org/10.1145/1377836.1377849

[31] R. Ferrini and E. Bertino, "Supporting rbac with xacml+owl," in *Proceedings of the 14th ACM symposium on Access control models and technologies*, ser. SACMAT '09. New York, NY, USA: ACM, 2009, pp. 145–154. [Online]. Available: http://doi.acm.org/10.1145/1542207.1542231

[32] V. Papakonstantinou, M. Michou, I. Fundulaki, G. Flouris, and G. Antoniou, "Access control for rdf graphs using abstract models," in *Proceedings of the 17th ACM symposium on Access Control Models and Technologies*, ser. SACMAT '12. New York, NY, USA: ACM, 2012, pp. 103–112. [Online]. Available: http://doi.acm.org/10.1145/2295136.2295155

[33] R. Brachman and H. Levesque, *Knowledge Representation and Reasoning*. Morgan Kaufman, 2004.

[34] M. L. Pardal, M. Harrison, and J. A. Marques, "Assessment of Visibility Restriction Mechanisms for RFID Data Discovery Services," in *IEEE Int'l Conf. on RFID*, April 2012, p. 7.

[35] M. L. Pardal, M. Harrison, S. Sarma, and J. A. Marques, "Enforcing RFID Data Visibility Restrictions Using XACML Security Policies," in *IEEE Int'l Conf. on RFID Technology and Applications*, November 2012.

[36] ——, "Performance Assessment of XACML Authorizations for Supply Chain Traceability Web Services," in *8th Int'l Conf. on Next Generation Web Services Practices (NWeSP)*, November 2012.

[37] G. Karjoth, A. Schade, and E. V. Herreweghen, "Implementing ACL-Based Policies in XACML," in *Annual Computer Security Applications Conf. (ACSAC)*, December 2008, pp. 183–192.

[38] A. Ilic, A. Grssbauer, F. Michahelles, and E. Fleisch, "Understanding data volume problems of RFID-enabled supply chains," *Business Process Management Journal*, vol. 16, no. 6, 2011.

[39] R. J. Bayardo, D. Gruhl, V. Josifovski, and J. Myllymaki, "An evaluation of binary XML encoding optimizations for fast stream based XML processing," in *Proc. of the 13th Int'l Conf. on World Wide Web*, ser. WWW. New York, NY, USA: ACM, 2004, pp. 345–354.

[40] B. Butler, B. Jennings, and D. Botvich, "XACML policy performance evaluation using a flexible load testing framework," in *Proc. of the 17th ACM Conf. on Computer and Communications Security*, ser. CCS. New York, NY, USA: ACM, 2010, pp. 648–650.