Enforcing RFID Data Visibility Restrictions Using XACML Security Policies

Miguel L. Pardal[†], Mark Harrison[‡], Sanjay Sarma[§], José Alves Marques[†]

[†]Department of Computer Science and Engineering
Instituto Superior Técnico, Technical University of Lisbon, Portugal

[‡]Auto-ID Labs, Institute for Manufacturing,
University of Cambridge, UK

[§]Auto-ID Labs, Massachusetts Institute of Technology,
Cambridge, Massachusetts, USA

Email: miguel.pardal@ist.utl.pt, mark.harrison@cantab.net, sesarma@mit.edu, jose.marques@link.pt

***Abstract*—Radio Frequency Identification (RFID) technology allows automatic data capture from tagged objects moving in a supply chain. This data can be very useful if it is used to answer traceability queries, however it is distributed across many different repositories, owned by different companies.**

Discovery Services (DS) are designed to assist in retrieving the RFID data relevant for traceability queries while enforcing *sharing policies* that are defined and required by participating companies to prevent sensitive data from being exposed.

In this paper we define an interface for Supply Chain Authorization (SC-Az) and describe the implementation of two visibility restriction mechanisms based on Access Control Lists (ACLs) and Capabilities. Both approaches were converted to the standard eXtensible Access Control Markup Language (XACML) and their correctness and performance was evaluated for supply chains with increasing size.

I. Introduction

The world around us interwines thousands of supply chains that provide goods from the points of production to the points of consumption. Supply Chain Management (SCM) solutions focus on supply chain planning and execution, integrated with companies' Enterprise Resources Planning (ERP) systems. Their ultimate goal is to get the right amount of products from production to consumption in the least amount of time and at the lowest cost. The main obstacle to their effectiveness is *inaccurate* or *untimely* information [1].

Radio Frequency Identification (RFID) technologies can be used to significantly improve the quality of supply chain information [2], especially when used with the EPCglobal architecture [3] that provides global and unique identifiers for physical objects along with hardware and software standards.

RFID and EPC standards can improve SCM and ERP systems, by providing an interface to answer highly granular traceability queries, like *Tracking - Where is the object?* - and *Tracing - Where has the object been?* [4] However, the data remains fragmented across the supply chain.

A. Problem

Traceability data is captured by different organizations at diverse geographical locations, and then it is stored within independent data silos, as depicted in Figure 1. These silos use the EPC Information Services (EPC IS) [5] standard and provide means to capture and exchange consolidated data.

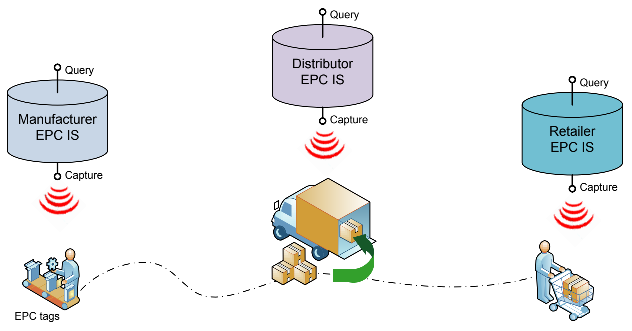

Fig. 1. RFID data is captured and stored across the supply chain.

This disjunction is not simply a data partition problem that can be solved technically with more network connections and bigger databases. It is a security problem of paramount importance for the participants of the supply chain. Although companies can benefit from sharing data, data about the flow of physical goods can expose sensitive business information (e.g. levels of demand and inventory, supplier relationships, etc.) and companies do not want to risk this being leaked to competitors and to other unauthorized parties. So, the definition and enforcement of *data sharing policies* is indispensable, if companies are ever going to participate in a Business-to-Business (B2B) traceability system.

B. Proposal

EPC Discovery Services (EPC DS) [6] are an extension to the EPCglobal architecture that is intended to address the security and partition challenges. Figure 2 shows how Discovery Services (DS) assist in traceability queries: earlier, when EPC-identified objects are observed and data is captured, a DS is notified; when a query is issued, a DS is contacted to find the IS instances containing relevant data. However, DS will only make the data visible if its owners explicitly authorized it.

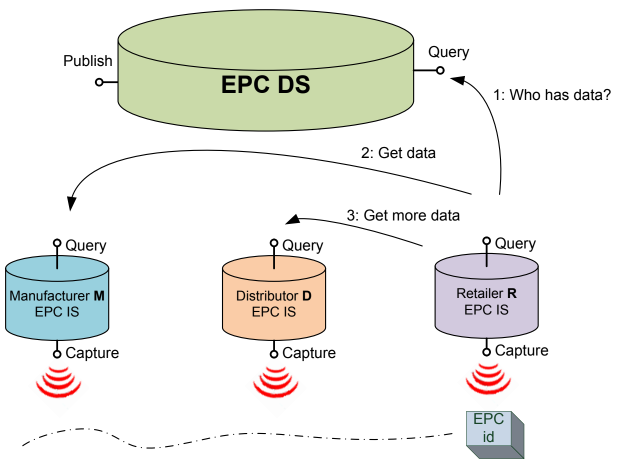

Fig. 2. Traceability information system using EPC IS and DS.

In this paper we define the Supply Chain Authorization (SC-Az) Application Programming Interface (API) to express authorizations with supply chain concepts, and we implement it using two "classical" authorization mechanisms [7]: Access Control Lists (ACLs) and Capabilities. We also translate both approaches to eXtensible Access Control Markup Language (XACML), a standard authorization language, and then compare their correctness and their performance as the supply chain size increases.

C. Overview

The rest of the paper is organized as follows. First we review related work. Then we define SC-Az and its implementations. Next we briefly introduce XACML and describe the assessment tool. We evaluate the performance and discuss the findings and conclude the paper with a summary of the contributions and future work.

II. Related work

A. Traceability architecture

The EPCglobal IS + DS traceability architecture is *semi-centralized* - DS instances are few compared to IS instances and play a "special" role - and traceability data is *referenced* (not copied). For these reasons, this architecture is classified by Do et al. [8] as Meta-Data Integration (MDI).

Previous work by Evdokimov et al. [9] qualitatively analyzed traceability architectures and compared functional requirements. Pardal and Alves Marques [10] surveyed over 20 traceability systems, summarized them in four categories, and quantitatively analyzed each one concluding that MDI it has the second-best performance estimates and it provides additional indirection levels that can be used to address other concerns. Namely, DS can be implemented as a *trusted third party* that can mediate (partial) trust between the supply chain participants.

B. Visibility restriction approaches

In subsequent work, Pardal, Harrison and Alves Marques [11] studied *visibility restriction mechanisms* for MDI and presented effort estimates. This compared the performance of two approaches with a limited semantic expressivity, Enumerated Access Control (EAC) and Chain-of-Communication Tokens (CCT), as well as a third option with extensible semantics, named Chain-of-Trust Assertions (CTA). The results showed that the expected performance is similar for all approaches, and that access control for supply chains is different from traditional authorization: there is a lack of prior knowledge about who should be authorized because of the way each individual object path emerges.

C. Authorization

Data sharing policies need to be expressed in ways that make sense both for the security and supply chain users while meeting performance requirements [12].

Shi et al. [13] implemented a secure DS and used an extended attribute-based access control to implement fine-grained access policies. This implementation could be extended to support XACML as a policy input format because it is a standard way of defining and enforcing policies, making auditing and validation easier.

Policy translation approaches are presented by several authors. Karjoth et al. [14] converted a vendor-specific policy language to XACML that included ACLs. Alm and Illig [15] translated complex policies such as 'Role-Based Access Control' and 'Separation of Duty'. Motivated by the verbosity of XACML, Cox [16] converted the policies to executable Java classes and achieved increased performance and better debugging capabilities.

Butler et al. [17] evaluated *correctness* by comparing responses from different XACML implementations.

Turkmen and Crispo [18] compared *performance* of the 'policy loading' and 'request evaluation' operations of XACML implementations, and they reported loading issues with more than 100 policies. Liu et al. [19] discussed performance optimization techniques and showed that several improvements are possible.

III. Supply Chain Authorization

We propose an API called Supply Chain Authorization (SC-Az) to allow companies participating in a supply chain to express their authorization concerns using concepts in that domain: item, company, etc. Figure 3 presents its operations: *init* for initializing, *share* for granting access, *request* to ask for access, and *enforce* to verify permission.

SC-Az is used to capture authorization requirements in a formal way that is understandable by supply chain users and

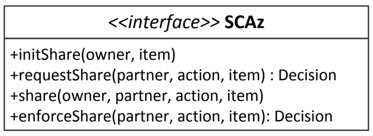
<<interface>> SCAz
+initShare(owner, item)
+requestShare(partner, action, item) : Decision
+share(owner, partner, action, item)
+enforceShare(partner, action, item): Decision

Fig. 3. SCAz interface operations.

then it is translated to XACML policy that can be deployed and enforced in a standard infrastructure.

We developed two alternative implementations of SC-Az. Each one is a different formulation meant to express *visibility restriction* approaches: Enumerated Access Control (EAC) and Chain-of-Communication Tokens (CCT).

Figure 4 shows how the classes relate to the interfaces: each class implements a mechanism-specific API and both implement the SC-Az API, allowing the same business needs to be mapped transparently to distinct implementations.

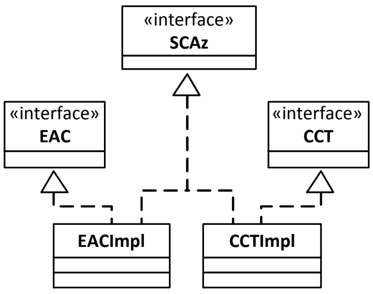

Fig. 4. SCAz, EAC, and CCT interfaces and classes.

A. Enumerated Access Control

EAC is based on ACLs [7], represented in Figure 5, that keep the access rights indexed by the object identifier. Each ACL has an owner and several permissions that define authorized user-action pairs.

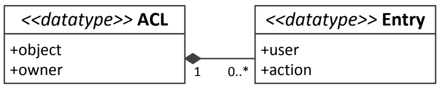

Fig. 5. Access Control List (ACL) data structure.

Each data set - *the events owned by a company C about an item i* - is protected by an ACL. Figure 6 shows the ACL-related operations. The master ACL is maintained at the DS, but a local copy is maintained at each IS to also protect its records. The data owner contacts DS to add new partners to the ACL.

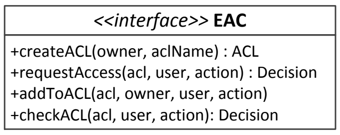
<<interface>> EAC
+createACL(owner, aclName) : ACL
+requestAccess(acl, user, action) : Decision
+addToACL(acl, owner, user, action)
+checkACL(acl, user, action): Decision

Fig. 6. EAC interface operations.

B. Chain-of-Communication Tokens

CCT is an adapted Capabilities [7] mechanism because the access rights are kept within the object structure. This means that when the token is shared, the access rights are automatically shared. Figure 7 presents the token's contents that are the authorized action-resource pairs, an Universally Unique Identifier (UUID) for the 'id' and the 'secret'.

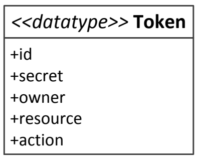

Fig. 7. Authorization Token data structure.

Each data set is protected by a token. Figure 8 shows the token-related operations. New visibility scopes can be created by issuing new tokens. The data owner sends the token to new partners directly or via DS to authorize them.

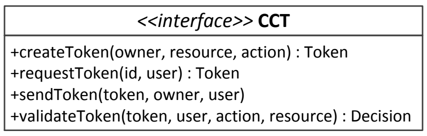
<<interface>> CCT
+createToken(owner, resource, action) : Token
+requestToken(id, user) : Token
+sendToken(token, owner, user)
+validateToken(token, user, action, resource) : Decision

Fig. 8. CCT interface operations.

IV. eXtensible Access Control Markup Language

XACML is a standard proposed by OASIS [20], currently in version 3.0. It is an eXtensible Markup Language (XML) vocabulary that represents authorization policies and requests. The authorization is the verification of a subject's right to execute an action on a resource.

The use cases for XACML are fine-grained access control and externalized security. Externalized security encompasses external user management, external authentication, external logging and auditing, and external authorization. The idea is to avoid hard-coded security and to unify the security management across applications so that business rules can be changed dynamically. Hebig et al. [21] demonstrated how several technologies can work together for this purpose. Using this approach, traceability data sharing policies can be authored by the data owner and then used both at EPC DS and IS level for improved enforcement consistency.

A. Policies

A XACML document has the structure represented in Figure 9. The document can contain a single policy or (nested) policy sets. A policy consists of a set of obligations, a target, a set of rules, and a rule-combining algorithm.

Obligations are actions that must be executed when a request is processed and are typically used to write audit logs.

The *target* defines a simplified set of conditions and attributes that help in determining whether the policy is relevant for

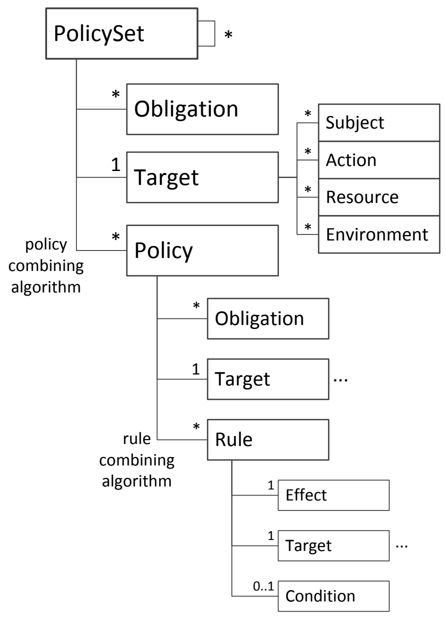

Fig. 9. XACML Policy structure.

the request. It is very important for performance, because it provides policy index keys.

A policy *rule* is composed of: effect, target, and conditions. The rule *effect* can be 'Permit', 'Deny', 'NotApplicable'[1], or 'Indeterminate'[2]. The rule *target* again determines if the rule is relevant for the request. The rule *conditions* are statements about attributes with arbitrary nesting of functions that, upon evaluation, return either 'True', 'False', or 'Indeterminate'. The rule effect is the intended consequence of the rule - 'Permit' or 'Deny' - when the condition returns 'True'. XACML has a fixed set of functions to keep policies declarative and with low algorithmic complexity [15]. However, *custom functions* can be defined by the Policy Decision Point (PDP).

Finally, a *policy/rule combining algorithm* is responsible for reconciling conflicts between policies/rules and to arrive at one outcome per policy per request using logical conjunction, disjunction or other algorithms.

B. *Workflow*

The authorization architecture [22] [23] and the request processing workflow are represented in Figure 10. The Policy Administration Point (PAP) is used to author and manage policies, and they are loaded before the requests happen (0). An access attempt is intercepted by Policy Enforcement Point (PEP) (1) and an access request is sent to the PDP (2). The Policy Information Point (PIP) provides attribute values, if necessary (3) and the PDP makes a decision (4). Any implied obligations are serviced (5) and the action is permitted or denied (6).

[1]'NotApplicable' is used when no policy was matched or when some required attribute was missing.

[2]'Indeterminate' signifies an error during evaluation of the policy.

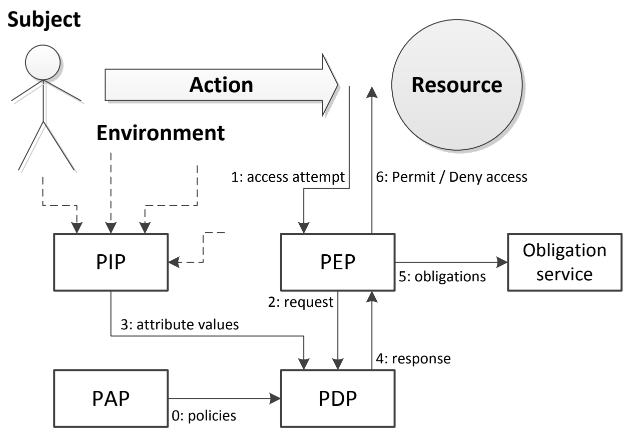

Fig. 10. XACML request processing.

V. Assessment tool

An assessment tool named *SC-Az tool* was developed to test the correctness and performance of the security implementations. The tool can generate supply chain scenarios, generate policies and requests, convert them to XACML, load and evaluate them, and measure the performance.

The HERAS-AF[3] [24] XACML engine was selected for use because it is open-source and well documented.

A. *Data flow*

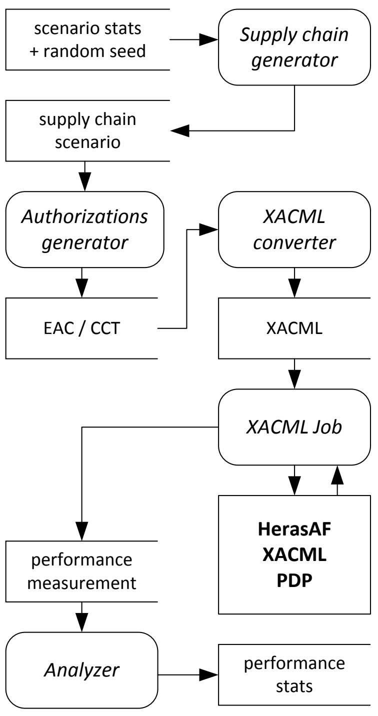

Fig. 11. SCAz tool data flow diagram.

Figure 11 represents the data flow associated with the tool. The initial input contains scenario statistics, such as the 'chain

[3]Holistic Enterprise-Ready Application Security Architecture Framework.

length' and the 'number of items', and the 'random seed', that are used to generate a supply chain scenario. Next, authorizations and requests are also generated with equal probability of 'Permit' or 'Deny'. The access decision is computed using either EAC or CCT to determine the expected outcome. Then the policies and requests are translated to XACML, and a job is sent to the PDP, where the policy files are unmarshalled[4] and deployed, the requests are unmarshalled and evaluated, and the response is marshalled[5]. The measurements are collected and final statistics are computed.

B. *Conversion to XACML*

1) From EAC: Each item data set has a single ACL, and that ACL corresponds to a single XACML policy [14].

The policy name and target is the item identifier. The rule combining algorithm is 'first-applicable' meaning that the outcome of the first rule that matched is the access decision. For each entry in the source ACL, a 'Permit' rule is generated for each 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 policy name and target is the token identifier. 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.

The secret is represented as a Base-64 binary literal in the policy[6] but it could have been retrieved from a secure store.

VI. Evaluation

Using the SC-Az tool we performed multiple experiments to evaluate the *correctness* and *performance* of XACML for supply chain authorizations.

A. *Correctness*

We generated the XACML policy from SC-Az using EAC or CCT. We compared the SC-Az decisions with the XACML decisions. If any inconsistent response is returned - e.g. the implementation says 'Deny' but XACML PDP says 'Permit' - then the discrepancy is detected and the executing job is cancelled to ensure that only correct responses are used.

B. *Performance*

To assess the performance we designed two experiments: 'companies' and 'items'. The test machine was a Quad-core CPU[7] 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 absolute values of the presented results will differ in a different server, but the relative performance should be similar. The experiments were repeated several times[8] to obtain statistically meaningful values.

[4]Unmarshal: convert from XML text file to Java objects in-memory.
[5]Marshal: convert from Java objects in-memory to XML text file.
[6]Base-64 is a standard way to represent binary data as text.
[7]Intel Core 2 Quad Central Processing Unit Q8300
[8]At least 30 times each, so that the sampling distribution can be considered 'normal' according to the Central Limit Theorem.

1) Companies: We considered three supply chains with different lengths: short (3), medium (6), and long (12).

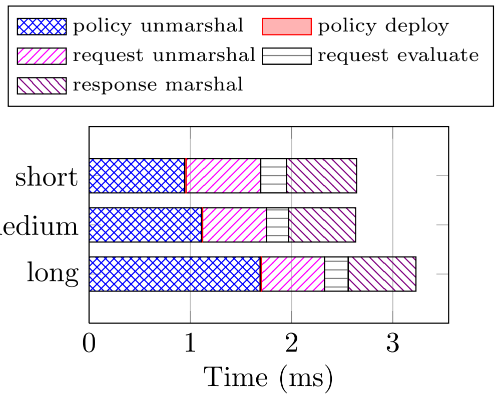

Fig. 12. EAC processing time breakdown for request evaluation.

Figure 12 shows the processing time for the three supply chains using EAC. The number of authorized partners in a policy increases with the item path length, making the generated policy size also increase. The average response time grows with the chain length, but below linear for the considered chains. For CCT (not shown), each XACML policy has constant size, because the token and its authorized actions have constant size. This improves the performance, making CCT faster than EAC. The job times are dominated by the persistent storage access times (91.5% of the overall time) whereas 'policy deployment' and 'request evaluation' take roughly just 0.5% and 8%, respectively.

2) Items: Figure 13 presents a plot of the 'request evaluation' time for increasing number of ACL and token policies.

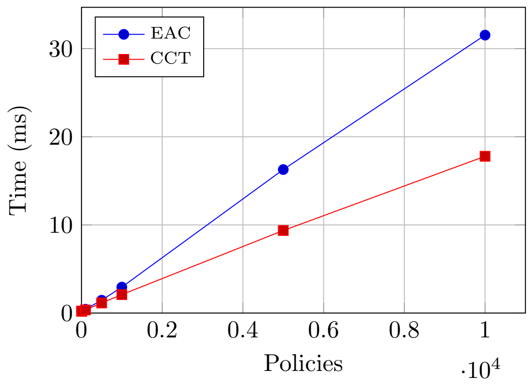

Fig. 13. EAC and CCT evaluation time with increasing item numbers.

CCT has better performance, roughly 80%, explained by the constant size of the token policies versus the increasing size of the ACL policies.

VII. Conclusion

This paper introduced the SC-Az API for expressing authorizations for data sharing in the supply chain domain. It also introduced two implementations: EAC and CCT. Both policy formats were converted to the standard XACML format, and the HERAS-AF implementation was used to assess the correctness and the performance. We verified that the data sharing policies could be translated and enforced correctly using a standard XACML infrastructure. We also verified the correctness of the visibility restrictions.

We conclude that XACML is a useful policy interchange and execution standard for supply chain authorizations whereas SC-Az can be much more intuitive to express the intended restrictions. Combining both, we have a standard policy infrastructure and a domain-specific interface.

Regarding performance, the results show that tokens (CCT) are clearly better than ACLs (EAC). However, ACLs can provide finer control over authorizations. We can leverage both advantages by starting with a quicker token verification - *Does the querying party hold the token?* - and then a more detailed one - *Is the querying party listed in the ACL?*

We found that the policy loading and unmarshalling is expensive and therefore it should be leveraged for several requests, corroborating the findings of Turkmen and Crispo [18]. The performance of EAC started to degrade after 10^4 policies and the performance of CCT after 10^5 policies, which might pose problems for large item volumes (e.g. 10^6). However, there is room for improvement by using a production-quality XACML policy store, better indexing of policies, etc [19].

A. *Future work*

We will improve the job execution to try to effectively measure the performance of XACML with more policies.

The performance impact of representing object groupings - *batches* - and company sets - *groups* - will also be assessed. It is expected that these formulations maintain the performance level currently measured but allow for larger business domains.

We will also compare EAC and CCT with Chain-of-Trust Assertions (CTA), a third implementation that uses logic and that has extensible semantics. We will try to convert CTA to XACML directly, but the predefined functions might not be enough to express some of the assertions. We will either add custom functions to the PDP or we will use the PIP to integrate with an external reasoning engine.

Finally, the overall latencies of traceability queries for single EPC, EPC enumerations, and EPC ranges will be measured because these are the use cases that will have the most direct impact on enterprise systems' performance.

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

[2] E. W. Schuster, S. J. Allen, and D. L. Brock, *Global RFID: The value of the EPCglobal network for supply chain management*. Springer, 2007.

[3] *The EPCglobal Architecture Framework 1.4*, Std.

[4] 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.

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

[6] T. Burbridge and M. Harrison, "Security Considerations in the Design and Peering of RFID Discovery Services," in *IEEE Int'l Conf. on RFID*, Orlando, USA, 2009, pp. 249–256.

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

[8] H.-H. Do, J. Anke, and G. Hackenbroich, "Architecture evaluation for distributed Auto-ID systems," in *Proc. 17th Int'l Workshop on Database and Expert Systems Applications (DEXA)*, 2006, pp. 30–34.

[9] S. Evdokimov, B. Fabian, S. Kunz, and N. Schoenemann, "Comparison of Discovery Service architectures for the Internet of Things," in *IEEE Int'l Conf. on Sensor Networks, Ubiquitous, and Trustworthy Computing (SUTC)*, 2010, pp. 237–244.

[10] 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.

[11] 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.

[12] BRIDGE, "Requirements document of serial level lookup service for various industries," University of Cambridge and AT4 wireless and BT Research and SAP Research and ETH Zurich and GS1 UK, Tech. Rep., August 2007.

[13] J. Shi, D. Sim, Y. Li, and R. Deng, "SecDS: a secure EPC discovery service system in EPCglobal network," in *Proc. of the 2nd ACM Conf. on Data and Application Security and Privacy*, ser. CODASPY. New York, NY, USA: ACM, 2012, pp. 267–274.

[14] 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.

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

[16] B. J. Cox, "Policy Based Access Control (PBAC) for Diverse DoD Security Domains," Technica Corporation, Tech. Rep., March 2011.

[17] 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.

[18] 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.

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

[20] B. Parducci, H. Lockhart, and E. Rissanen, *eXtensible Access Control Markup Language (XACML) Version 3.0*, OASIS Std., August 2011.

[21] 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.

[22] 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

[23] 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

[24] F. Huonder, "Conflict Detection and Resolution of XACML Policies," Master's thesis, University of Applied Sciences Rapperswil, July 2010.