# A policy-based framework for interoperable digital content management

Fernando Marques Figueira Filho and
João Porto de Albuquerque and
Paulo Lício de Geus
Institute of Computing
University of Campinas
13083-970 Campinas/SP Brazil
e-mail: (fernando, jporto, paulo)@las.ic.unicamp.br

Heiko Krumm
FB Informatik
Universität Dortmund
D-44221 Dortmund, Germany
e-mail: heiko.krumm@udo.edu

*Abstract*— **Through the past years, several digital rights management (DRM) solutions for controlled dissemination of digital information have been developed using cryptography and other technologies. Within so many different solutions, however, interoperability problems arise, which increase the interest on integrated design and management of these technologies. Pursuing these goals, this paper presents a framework which aims at promoting interoperability among DRM systems, using a service-oriented architecture (SOA) and a high-level policy modeling approach.**

## I. Introduction

Digital Rights Management is a collection of technologies that enables controlled dissemination of digital information. Today, the majority of DRM applications are used in copyrighted content distribution, such as movies and music, but it is expected that those technologies will also benefit, in a near future, corporate enterprises, small content producers, and individuals who intend to securely distribute their information.

Although there have been considerable advances in the area, DRM systems still do not interoperate, which affects a wide range of participants. A content producer may want to distribute over different DRM platforms to expand his consumer covering, while the consumer may want to use his content in more than one specific device. Moreover, the lack of interoperability can be used to stimulate the monopoly of proprietary software and devices by some vendors, which can be harmful for both users and content producers.

The main difficulties to achieve interoperability are based on differences on formats and protocols, but also on difficulties in trying to integrate management while simultaneously operating different DRM systems. There are basically two approaches to achieve interoperability: (a) modifying existing systems in order to support one another—which requires the establishment of standards and the modification of core parts in each DRM platform; or (b) the elaboration of a system that serves as a bridge between the platforms by using their own native protocols and formats. As opposed to the former, this approach does not require any modification on the existing DRM platforms, since interoperability is achieved through an external system which supports cross-platform DRM functionality. Considering that modifying existing DRM platforms deals with legal aspects and depends on industry motivations, the approach of developing an external system seams more feasible to achieve interoperability nowadays.

Following this approach, this paper presents an approach to promote interoperability among DRM platforms. The framework is based on the fact that in every platform, the lifetime of contents follows basically the same steps: firstly, it is packaged using cryptography, in order to protect it against unauthorized users. Then, at some moment during content distribution, it is licensed to a specific user or device. A license is a file containing the rights and conditions, described in a platform-specific format, which govern contents' usage by that particular user. Our framework relies upon the policy-based management approach [1], representing those rights and conditions by means of a single policy-based model, which is generic for every DRM platform.

Interoperability is promoted by specialized framework services which are able to interpret this policy model and generate licenses in different formats, each one compatible with a specific DRM platform. Content packaging is also done by the framework, following each platform specification. Services are distributed in a service-oriented architecture (SOA) designed to be loosely coupled, so as new DRM technologies emerge, new services can be easily attached to the framework. As such, they are implemented using Web Services to provide easier compatibility with most computer architectures and programming languages.

The policy model is conceptually based on two other existing models: the Role-Based Access Control (RBAC) [2] and its extension, the Generalized RBAC [3]. These models define policies in a higher level of abstraction relying upon roles. Policies at this level are defined by object manipulation in a graphical interface, similar to the one used by [4], [5].

The next section presents the conceptual models in which our approach is based. The system architecture is analyzed in Section III and Section IV illustrates our approach by presenting some example scenarios. Subsequently, our implementation platform is detailed (Section V) and we discuss some related work in Section VI. Lastly, we cast conclusions for this paper and expectations around future work (Section VII).

## II. Policy Model

In this paper, policies are based in an object-oriented model which can be divided conceptually into two levels of abstraction—in the sense of a policy hierarchy [6]—as depicted in Fig. 1. Policies in the abstract level are more stable and their construction is supported by a graphical tool called MoBaSec, which was also used in other policy-based management applications such as [4], [5]. The abstract level is based on the role-based access control (RBAC) concepts [2] and on one of its extensions, the GRBAC [3].

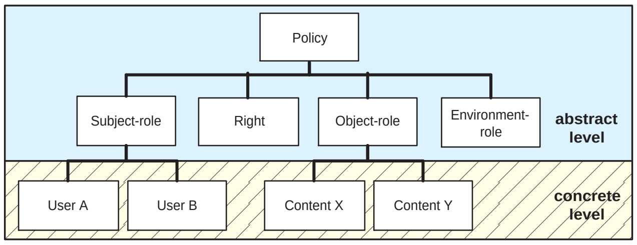
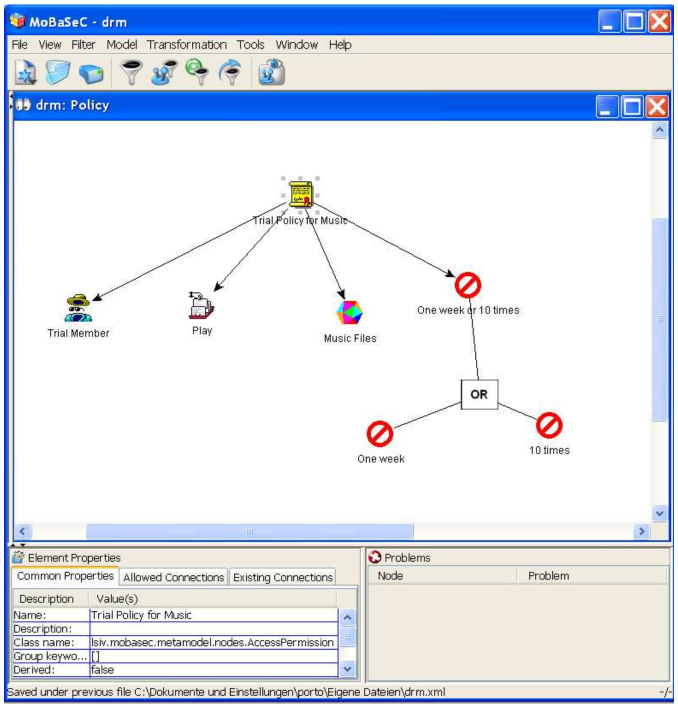


Fig. 1. Policy structure

RBAC has been used to simplify permission management, especially when users are hierarchically organized or when it is possible to identify common characteristics among them. Such scenario is found in various DRM business models (e.g. service subscription or purchasing, membership of a club or organization). Instead of associating rights to each user, rights are assigned to *subject-roles*, which are in turn associated to users. As such, in the DRM context, a *subject-role* is related to a *service package* that users may acquire, i.e. a group of rights over a number of digital contents that can be purchased.

Furthermore, DRM permissions commonly associate conditions and restrictions to a right (e.g. play, print), based on state information. This information is included in the license and used by a particular DRM platform to control, for example, the number of times a user exercises a right, the time interval during which a content can be used, among others. GRBAC extends RBAC through the introduction of *environment-roles*, which are applied to our policy model to incorporate those state-based conditions and restrictions.

As in [4], *environment-roles* are combined by means of logical operators to compose more complex conditions. Fig. 2 shows the following policy modeled in the graphical editor: "trial members can play music files until reaching 10 plays *or* during the period of one week counting from its first use". In the example, "trial members" is the *subject-role* and "play" is the policy *right*. "One week" and "10 times" are both *environment-roles* connected by the logical operator *OR* to compose a complex condition. GRBAC also defines *object-roles*, which are used here to group contents and build policies based on their common characteristics, such as type (audio, video etc.) and confidentiality level. In Fig. 2, "music files" is the *object-role*.



| Description | Value(s) |
|---|---|
| Name: | Trial Policy for Music |
| Description: | |
| Class name: | lsiv.mobasec.metamodel.nodes.AccessPermission |
| Group keywo... | [] |
| Derived: | false |

Saved under previous file C:\Dokumente und Einstellungen\porto\Eigene Dateien\drm.xml

Fig. 2. MoBaSec graphical interface

The second abstraction level models offers a more concrete view of the entities of a DRM system (e.g. users, contents) and has a much more dynamic behavior. While the upmost level is updated by human intervention in the graphical tool, the second level is updated by services of our framework according to the external DRM system activity. The architecture that comprehends these services and its functioning are covered in the next section.

## III. Framework Architecture

The framework proposed in this work employs a service-driven architecture which can be divided in platform-dependent services and core services, as depicted in Fig. 3. Platform-dependent services have different implementations of the same common interface, each of them capable of generating information in a platform-specific format. The other services interact with the *policy database* and perform the core functions of the framework.

All service interactions take into account that a previous abstract modeling step was done interactively by the policy modeler using the MoBaSec graphical tool (shown in Fig. 3). This modeling consists of creating all the *subject* and *object-role* types, as well as building the abstract level policies, as explained in Section II. At the end of this modeling step, all abstract level policies are stored in the *policy database* for further usage.

Each DRM platform has devices that interact with a Content Distribution System (CDS), from which it is possible to acquire content by subscribing to a service package. Such systems are found over the Internet and can offer different digital content types, as for instance movie rental portals and

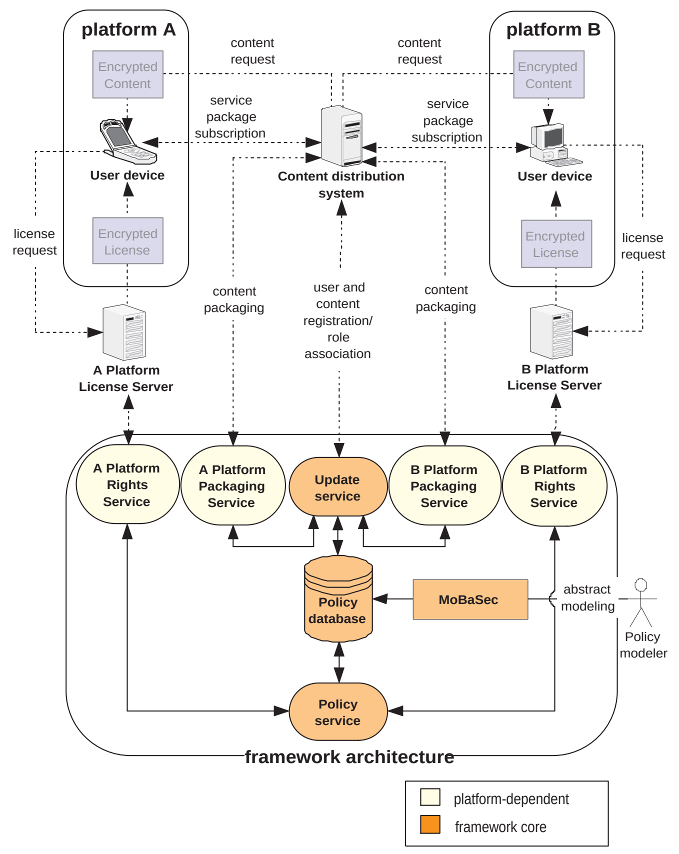


Fig. 3. Framework architecture

online music stores. In Fig. 3, the CDS delivers content to platforms A and B. For this purpose, however, it executes some tasks not addressed by our framework: (a) it provides user device authentication and keeps track of the correlation between users and their devices, e.g. a user has a device of Platform A and another of Platform B, and wishes to use the same content on both; (b) it distributes content in the correct format, depending on the requestor's platform.

On the other hand, our framework concentrates on appropriate content packaging and management of usage rights. As such, the *packaging services* of to the framework prepare the content for delivery to different DRM platforms. In Fig. 3, the framework includes a specialized *packaging service* for each DRM platform with which the framework operates.

As shown in Fig. 3, a plain content file is sent by the CDS to a specific *packaging service* relying upon a secure communication channel. The *packaging service* then appends the necessary headers and encrypts the content file using a pre-generated encryption key, which is associated with a global content identifier and stored in the *policy database* through a secure requisition to the *update service*. The resulting content package is then returned to the Content Distribution System and published. After that, users can acquire the content directly from the CDS (as in Fig. 3) or from another user—which is called superdistribution.

The CDS stores an association between each offered service package and a *subject-role*. On the other hand, the framework stores the associations between each user and the subject-roles in which he/she is allowed to act. When a user subscribes to a service package, the CDS contacts the framework's *update service* to associate to the user a new *subject-role*, which depends on the recently acquired service package. As a consequence of the subscription, the user automatically acquires all permissions associated with his new role.

Similarly, the framework stores the associations between each content and the *object-roles* in which it is classified. Before being published, the CDS uses the *update service* to register the content in the framework and its associated *object-roles*.

The packaged content has to be licensed before being used by a user device. For this purpose, a license is obtained from a License Server outside the framework (Fig. 3), which is specific for each DRM platform. A License Server works as a gateway between a DRM device and the framework. On the one side, it communicates with user devices by using native protocols through which the device is authenticated and requests licenses. On the other side, the License Server obtains licenses from the framework through a corresponding *rights service*.

A *rights service* is responsible for translating a policy set to a platform-specific format, commonly specified by a particular Rights Expression Language (REL). Whenever a query is made to the framework, the *policy service* searches the *policy database* for all policies related to the user who is requesting a license and a given content set. It then sends the resulting policy set to the *rights service*, which translates the rights information to a specific REL, packages this information in the corresponding format, and encrypts it into a license file. The license is then transferred to and stored in the License Server. Lastly, the license is obtained by the user device. The next section gives more details about service interfaces and presents some example scenarios.

## IV. Example Scenarios

This section presents two example scenarios which illustrate our approach. Framework services are detailed by showing the invocation of their main operations in sequence diagrams. Fig. 4 shows the sequence diagram for a content distribution scenario, from the deployment of a new content to the download of this content by the user device.

In order to distribute a new content to a specific DRM platform, the Content Distribution System must firstly register it in the framework by invoking the *registerContent()* operation, then load the content using the corresponding *packaging service* (*loadContent()*). Finally, the CDS retrieves the content packaged in the specific format using the *getPackagedContent()* operation, which receives the content identification and, among other information, the network location of the License Server from which the content's license can be obtained in the future. The *packaging service* also generates an encryption key and stores it by invoking the *updateEncKey()* operation in the *update service*.

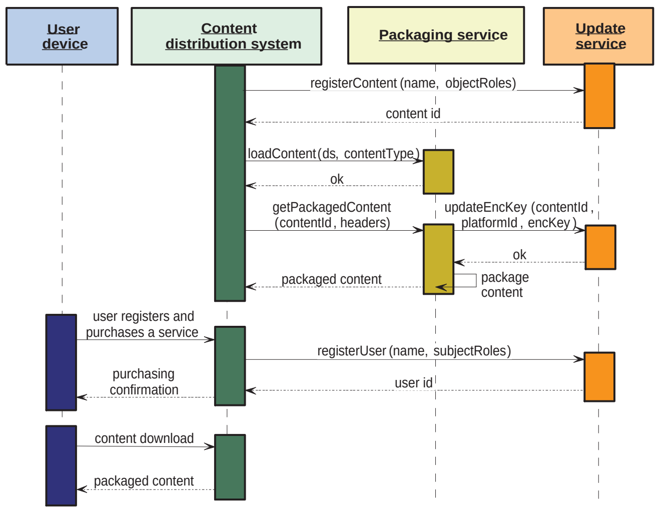

User device
Content distribution system
Packaging service
Update service
registerContent (name, objectRoles)
content id
loadContent(ds, contentType)
ok
getPackagedContent (contentId, headers)
updateEncKey (contentId, platformId, encKey)
ok
packaged content
package content
user registers and purchases a service
registerUser (name, subjectRoles)
user id
purchasing confirmation
content download
packaged content

Fig. 4. Content distribution scenario

The CDS interacts with the user providing a way (e.g. a web page) by which the user registers himself and subscribes to different service packages. At user subscription, the *subject role* associated with the chosen service package is added to the user's current subject role set. In Fig. 4, this is done by the *registerUser()* invocation.

Note that content and user registration is required just once in the system. After registration, the user can subscribe or unsubscribe service packages, which immediately modifies his current subject role set and consequently his permissions. Similarly, a content can be packaged into different formats after registration.

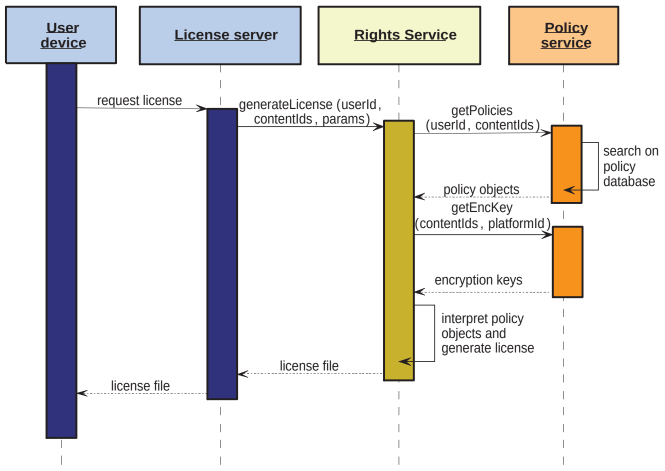

User device
License server
Rights Service
Policy service
request license
generateLicense (userId, contentIds, params)
getPolicies (userId, contentIds)
search on policy database
policy objects
getEncKey (contentIds, platformId)
encryption keys
interpret policy objects and generate license
license file
license file

Fig. 5. Licensing scenario

Fig 5 depicts a licensing scenario, which relies upon two other framework services (namely, *rights service* and *policy service*), a user device, and the corresponding License Server. The user device and the License Server communicate using a native protocol, by which the user device requests a license to a specific content set[1].

If the required license was previously generated and is already stored in the License Server, it is returned to the user device. Otherwise, the License Server has a client application which contacts the framework's *rights service*, passing the user and content identifications, as well as parameters with platform-dependent information. When receiving a license request, the *rights service* invokes the *policy service* by calling the *getPolicies()* operation, which searches the database for all policies related to that particular user and content set. The encryption keys used to package the content set are also retrieved by the *getEncKey()* operation. The keys are thus included in the license together with the rights information that were translated to a native format from the selected policies. Finally, the license file is returned to and stored in the License Server.

## V. Implementation Prototype

The framework is currently being developed in Java and integrated with the Nokia's S40 DRM platform [7]. This platform has a SDK composed by an emulator of the S40 Series cell phone and follows the Open Mobile Alliance (OMA) 1.0 specification [8]. OMA is an open standard widely adopted by mobile devices manufacturers and defines the rights expression language (REL), and the content package format, among other information used to implement the platform-dependent services of the framework.

A simple test application runs within a Java servlet container. It provides a web page by means of which a user might subscribe to the different service packages previously modeled with the graphical editor. Each subscription triggers invocations of the framework's *update service* relying upon the HTTP and SOAP protocols. Furthermore, a content can be downloaded from this web page to the emulator, which must activate the content before using it by acquiring a license from the License Server.

The License Server in turn consists of a WAP gateway developed to interact with the cell phone emulator. This gateway serves as a bridge between the framework's *rights service* and the emulator, distributing licenses to this by pushing them over WAP messages.

## VI. Related Work

This paper is an extension of a recently accepted work [9].

Some other recent work analyzes interoperability issues, as in Sun's project called DReaM [10], which also employs a service-oriented architecture. However, the "play once" license is assumed to be the common denominator for each DRM system, which limits the use of more complex usage rules.

Coral Consortium approaches the interoperability problem by developing a framework which attempts to fulfill the same requirements addressed in this paper. They provide the concept of *rights token*, which is a platform-agnostic representation of permissions and usage rights similar to a policy in our model,

[1]Under some DRM specifications, a single license can reference more than one content, e.g a multipart content composed by an image and a ringtone.

but without any tool assistance to design policies. The work group has recently published a whitepaper [11] describing the framework integration with Microsoft Windows Media DRM.

In a wider context, our work is related to the Model-based Management (MBM) [12], [13]. This approach employs an object-oriented layered model that aims at providing a smooth transition from an abstract view of the system to be managed and the policies that apply to it down to reaching a detailed system representation at the most inferior layer. It was already applied to the management of different security mechanism types, such as Virtual Private Networks [12], and to the integrated management of a number of network security mechanisms in large-scale, complex network environments [13], [5]. Furthermore, the SIRENA project [4] shows that the MBM approach can be profitably used with the GRBAC to address requirements of dynamic environment conditions. This is the inspiring direction taken by this work.

## VII. Conclusion

This work has presented a framework that offers a policy-based approach to the management of users' rights over digital content. The framework is specially conceived to achieve interoperability among the several existing DRM systems, relying upon a SOA architecture. A graphical policy editor is provided to edit an abstract set of high-level business policies that consist of *service packages*, i.e. groups of permissions over classes of digital objects. A number of Web Services is responsible for the establishment of the more dynamic model relationships such as the subscription of users to service packages, and the deployment of new content. As such, this separation between a more stable set of abstract policies and on-demand lower-level associations affords a more concise and intuitive graphical view of the business policies, improving their manageability while still coping with the highly dynamic nature of DRM relationships.

The core of the framework consists of two platform-independent Web Services that query and update the policy repository. Another set of platform-specific Web Services interface between existing DRM systems and the core services of the framework. The effort to make an existing DRM system to use our framework thus consists of the implementation of specific applications to serve as gateways between the existing DRM system and the framework services. In the DRM systems we have analyzed—including Nokia's S40 platform used in our implementation prototype—this has turned out to be a task of relatively low complexity. In this manner, the most different flavors of DRM systems are able to work according to the same set of policies, which are centrally managed by our framework.

We plan to include support to other DRM systems in our prototype, so that their integration can be tested and evaluated. An aspect to be developed in future work is the policy translation procedure to be applied when using several DRM platforms. Currently, we are capable to translate a policy into the rights expression syntax that follows the OMA DRM specification or to any other with similar representation power. However, if we take into account the simultaneous use of a less powerful rights expression language—in which there is no sufficient directives to represent all policy variations of OMA REL—then a translation may not produce an equivalent result, causing inconsistency. One solution to this problem is achieved by translating a non-mappable policy to a more restrictive set of rights, or even considering a smarter policy search algorithm which takes into account the DRM platform when selecting the policy set to be translated. As such, we would ensure that a user do not receive unintended rights over any content, but rather is less empowered that it could be (if using a more capable platform) due to technological restrictions.

## References

[1] M. Sloman, "Policy driven management for distributed systems," *Journal of Network and Systems Management*, vol. 2, no. 4, pp. 333–360, 1994.

[2] R. S. Sandhu, E. J. Coyne, H. L. Feinstein, and C. E. Youman, "Role-based access control models," *IEEE Computer*, vol. 29, no. 2, pp. 38–47, 1996.

[3] M. J. Covington, M. J. Moyer, and M. Ahamad, "Generalized role-based access control for securing future applications," in *23rd National Information Systems Security Conference Proceedings*, 2000.

[4] S. Illner, H. Krumm, I. Lück, A. Pohl, A. Bobek, H. Bohn, and F. Golatowski, "Model-based management of embedded service systems - an applied approach," in *Proc. 20th Int. IEEE Conf. on Advanced Information Networking and Applications (AINA2006)*, vol. 2. Vienna: IEEE Computer Society Press, April 2006, pp. 519–523.

[5] J. Porto de Albuquerque, H. Isenberg, H. Krumm, and P. L. de Geus, "Improving the configuration management of large network security systems," in *Ambient Networks: 16th IFIP/IEEE International Workshop on Distributed Systems: Operations and Management, DSOM 2005, Proceedings*, ser. Lecture Notes in Computer Science, vol. 3775. Berlin Heidelberg, Germany: Springer-Verlag, October 2005, pp. 36–47.

[6] M. Sloman, "Policy hierarchies for distributed systems management," in *IEEE Journal on Selected Areas in Communications*, Dec 1993, pp. 1404–1414.

[7] Nokia. (2006) Forum Nokia. [Online]. Available: http://forum.nokia.com/

[8] *OMA DRM V. 1.0*, Open Mobile Alliance, 2004.

[9] F. M. F. Filho, J. Porto de Albuquerque, and P. L. de Geus, "A service-oriented framework to promote interoperability among drm systems," in *9th IFIP/IEEE International Conference on Management of Multimedia and Mobile Networks and Services, MMNS 2006, Proceedings*, ser. Lecture Notes in Computer Science. Springer Verlag, October 2006.

[10] G. Fernando, T. Jacobs, and V. Swaminathan. (2005) Project DReaM - An Architectural Overview. White Paper. Open Media Commons. [Online]. Available: http://www.openmediacommons.org/

[11] C. Consortium. (2006) Providing Interoperability with Windows Media DRM. White Paper. Coral Consortium. [Online]. Available: http://www.coral-interop.org/

[12] I. Lück, S. Vögel, and H. Krumm, "Model-based configuration of VPNs," in *Proc. 8th IEEE/IFIP Network Operations and Management Symposium NOMS 2002*, R. Stadtler and M. Ulema, Eds. Florence, Italy: IEEE, 2002, pp. 589–602.

[13] J. Porto de Albuquerque, H. Krumm, and P. L. de Geus, "On scalability and modularisation in the modelling of security systems," in *Computer Security - ESORICS 2005, 10th European Symposium on Research in Computer Security, Milan, Italy, September 12-14, 2005, Proceedings*, ser. Lecture Notes in Computer Science, vol. 3679. Berlin Heidelberg, Germany: Springer-Verlag, September 2005, pp. 287–304.