Security Patterns for Capturing Encryption-Based Access Control to Sensor Data

Angel Cuevas*†, Paul El Khoury*‡, Laurent Gomez*, Annett Laube*
Email: acrumin@it.uc3m.es, paul.el.khoury@sap.com, laurent.gomez@sap.com, annett.laube@sap.com
*SAP Research, SAP Labs France, Sophia Antipolis
†Department of Telematic Engineering University Carlos III of Madrid
‡LIRIS University of Claude Bernard Lyon I

***Abstract*—A security pattern describes a particular recurring security problem that arises in specific contexts, and presents a well-proven generic solution for it [1]. This paper describes a generic solution that ensures end-to-end access control for data generated by wireless sensors and consumed by business applications, based on a new approach for encryption-based access control. The existing security mechanism is captured as Serenity ("System Engineering for Security and Dependability") security patterns that describe a security problem and its solution in an abstract way. The structured description makes the security solution better understandable for non-security experts and helps to disseminate the security knowledge among application developers.**

I. Introduction

Several standardized methodologies were developed to support the development of secure systems. Most of these standards provide comprehensive methodologies for specifying, implementing and evaluating security of IT products. Unfortunately, security experts are the interpreters of such standards. The description of these standards in natural language limits their ability in passing knowledge to novice security users. The fundamental added value of adopting the security patterns approach is providing security for non-security experts [1].

Application developers stand at the front line transforming clients' requirement specifications into business applications. The adoption of these business applications are tightly dependent on their compliance to security regulations. Unfortunately, most of application developers are not experts in security and thus are incapable of satisfying such exigence.

Security patterns capture security expertise in abstract and concrete methodologies. This approach fits well as the candidate link between security experts and application developers to encompass business applications with a security shield.

In the WASP project [2], Wireless Sensor Networks (WSNs) [3] are being integrated with business applications. This integration raises several security challenges such as confidentiality of sensor data. The WSN integration has then to be provided together with security solutions. Application developers in such scenario are not security experts. Therefore, security patterns help providing security solutions for WSN applications.

In addition, security solutions related to sensors for non-experts in this domain define a major challenge for the adoption of this futuristic vision related to security patterns.

This paper focuses on capturing a security solution that provides access control to sensor data based on light-weight encryption and grant provision [4] as security patterns, following the Serenity methodology [5].

Even though there are several ways for implementing the encryption-based access control to sensor data, our purpose is to capture the properties and functionalities that are common to all implementations using security patterns. Therefore, any particular solution could be derived from the defined pattern. Furthermore, capturing expertise as a pattern makes security solutions for a given problem more general. It is easier for non-security experts to find a suitable solution for a particular problem by searching into the patterns' properties.

The remainder of this paper is organized as follows. In section II, we give an overview Security Patterns. Section III proposes two scenarios used as example for security pattern's approach on WSN applications. We present an overview of the proposed security solution in section IV. Section V defines the abstract model used to capture this security solution as a combination of patterns. Next, section VI describes the security patterns and integration scheme of the proposed solution in more detail. Related works are discussed in section VII and finally in section VIII we conclude and explain future work.

II. Security Patterns Overview

To accomplish the security patterns' 'mission', a list of objectives summarized in four fundamental steps [1] has to be achieved. First, most of the novice security users should understand how experts approach key security problems. Second, security experts should be able to identify, name, discuss and teach both problems and solutions efficiently. Third, problems should be solved in a structured way. Fourth, dependencies and side-effects should be identified and considered appropriately. The connotation of these objectives emerged as appealing for research studies.

The usual natural language description for security patterns opens room for different interpretation of solutions provided and problems described by these patterns. Hence, none of the previously four objectives of the patterns' mission can be achieved.

Following this particular path, Schumacher et al.[6] presented a set of security patterns for the development process.

978-0-7695-3329-2/08 $25.00 © 2008 IEEE
DOI 10.1109/SECURWARE.2008.17

Yoder and Barcalow [7] propose architectural patterns that can be applied when adding security to an application. Fernandez and Pan [8] describe patterns for the most common security models such as Authorization, Role-Based Access Control, and Multilevel Security. These security patterns had a shy adoptions in the security field. Indeed their description in natural language limits their applicability and forbid any reasoning mechanism.
Serenity EU project through a list of narrow yet complex studies [5], [9], [10] tackles the security patterns objectives. One of the essential proposals from Serenity is to provide novice users the Serenity Security & Dependability pattern package. This package comprises the *expert-proofed* security solutions and *tested* plug-n-play deployable implementations.

The research interest in security patterns focuses in particular on capturing solutions for recurring security problems that arise in specific contexts. One could wonder at which granularity security problems should be analyzed to be captured in a Pattern? There is no clear answer in the literature rather, as long as 'smaller' security problems can be isolated and efficiently solved separately, they suit better as isolated patterns. Usually a complex security solution is not captured in one stand-alone pattern, in order to cover the generality aspect of the abstract solution.
To have an intuitive description for the solution proposed in this paper, we adopt the Serenity approach using three artefacts. In Serenity the description of these artefacts enable selection, adaptation, usage and monitoring at runtime by automated means. The hierarchy is composed by three artefacts, Security Classes, Security Patterns, and Security Implementations. Although this paper emphasized the use of Security Pattern artefact, [11] presents an intuitive and extensive description of them all.
In Serenity, *security patterns* are detailed descriptions of abstract security solutions that contain all the information necessary for the selection, instantiation and adaptation performed on them. Such descriptions provide a precise foundation for the informed use of the solution and enhance the trust in the model.
An *integration scheme (IS)* is an additional artefact defining the combination of security patterns. Since complex solutions rely on the use of several patterns, they have to be defined as Integration Schemes. In the IS, the relations among the patterns are established in order to describe a complex security solution.
In a nutshell, to capture our expertise, we describe security patterns providing simple tasks which could be used in different other security solutions and we depict an IS merging these simple tasks and describing the full security solution.

This paper relies on the Serenity representation of security patterns [12] to transfer the first three objectives of security patterns for the Encryption-Based Access Control to Sensor Data to non-security experts. The most important parts of a security pattern description are in the following:

- **Problem/requirements and context**: The problem is the vulnerable part in an asset that can also be described as requirements which need to be solved. The context defines the recurring situation where the problem/requirement can occur.
- **Solution:** The solution is defined as a mechanism that is used to resolve the corresponding requirement/problem. It defines the sequential flow of operations in solving the security problem.
- **Pre-Conditions:** They indicate assumptions and restrictions related to the deployment of the pattern. Before applying a pattern, users or applications in some cases should check the satisfyability of these pre-conditions. Obviously, pre-conditions are elements used during the selection of suitable patterns for a particular problem.
- **Properties:** They describe which security elements the pattern is providing. This is the basic element used to discriminate whether a pattern is useful for a security problem or not.
- **Features:** They are additional characteristics to the patterns' properties. They are additional criteria in selecting the suitable patterns.
- **Consequences:** They are the effects of the compromise resulting from the application of the pattern's solution. In particular cases, using security patterns implies an increase in cost (economic, more complex mechanisms, etc).

III. Scenarios

In order to illustrate the definition of security patterns, we introduce two scenarios describing the integration of WSN within business application. Additionally, we identify a few security requirements of business applications closely related to confidentiality of sensor data.

The ability to monitor and control physical environments (e.g. building, battlefield or body) makes Wireless Sensor Networks very attractive to business application. From military to traffic control, including healthcare, a lot of different business application domains have a strong interest in WSN [3]. We can illustrate the integration of WSNs with the two following examples:

- **Remote Patient Monitoring**: A patient is monitored remotely at home after surgery. Several physiological information such as pulse, body temperature are measured from a set of wearable sensor nodes, or Body Sensor Network. In addition, ambient sensor nodes acquire ambient temperature, patient activities, or her position. This information is delivered to a Medical Emergency Response Center (MERC), 24 hours per day. In case of any irregularities in patient health condition, an alert is triggered to the MERC which contacts a physician for a home visit.
- **Energy Consumption Monitoring**: It is motivated by the energy consumption optimization within a building. Such monitoring application allows the detection of energy loss (e.g. electricity, heat) and support building administrator with large amount source of information for further optimization. In this scenario, an ambient sensor

network is deployed within a building and monitor light, temperature, humidity, room presence etc.

These scenarios introduce several technical challenges related to the integration of WSN into existing business application (e.g. data routing from the sensor nodes to the business applications, data processing, filtering). In addition, business application raise many security requirements when it comes to the integration with WSNs. In the WASP project [2], we identified security requirements for three business domains (healthcare, road management and herd control). Each business application has its own specific security requirements. Although, security requirements are tightly related to the business application, end-to-end confidentiality of sensor data is generally required by business application. The security goal is to prevent disclosure of any sensitive information (e.g. patient location) to unauthorized entities.

IV. Solution Description

In [4], the authors propose a novel approach for access control for wireless sensor data, based on an end-to-end symmetric encryption of data from sensor node to business application. This encryption-based access control scheme restricts sensor data access to only authorized business applications.

This approach is a hybrid approach, relying to active contributions of the nodes (distributed approach) but maintaining most of the intelligence of the system in a centralized entity. The key intuition is to use light-weight cryptography to achieve access control: if sensor data is encrypted, only the owner of the decryption key can access the data. The scheme allows the generation of hierarchical, time-bounded cryptographic keys. Sensor nodes just have to perform simple encryption operations to enforce data access control, regardless of who the listeners are. A central authority is in charge of delivering keys to applications according to the access control policy.

First, a classification of sensor data is established, based on their level of sensitivity, or **authorization level**. Data whose disclosure does not arise high privacy issues is mapped to low authorization levels. Similarly, highly sensitive data will be mapped to high authorization levels. The resulting mapping expresses the security preferences of an **Access Control Module (ACM)** and defines the access control policy.

Each authorization class is then mapped to a cryptographic material. Sensor data associated to the same class is encrypted with the same cryptographic material. In addition, and following the hierarchical organisation of the sensor data classification, this material allows to encrypt any lower class of sensor data.

The ACM is then in charge of (i) the distribution of the cryptographic material to the nodes, and (ii) to the business applications. Based on the **Credentials** of business applications, the ACM assesses their authorization classes and provides them with decryption material. Business applications receive **Grant** which allows them to decrypt sensor data from a certain authorization class, and lower.

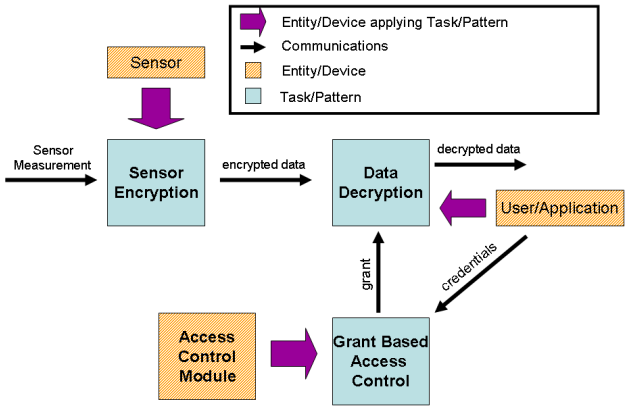

Fig. 1. Conceptual Model

V. Conceptual Model

"A picture is worth a thousand words", toward providing complex security solutions to software developers, we start by capturing it with an abstract model. Figure 1 depicts the Encryption-Based Access Control to Sensor Data conceptual model highlighting input, output and entities specific for the understanding of this solution. As shown in the abstract model, the proposed solution is clearly divided in different related tasks performed by different entities.

The input for our model is any sensor measurement including: physical medium measures (e.g. temperature, humidity, etc), sensor's parameters (e.g. sensor's battery value), etc. This value is translated into digital data and encrypted within the sensors which apply the *Sensor Encryption* task. Encrypted data are retrieved by the users[1] who want to obtain the confidential information. In order to get that information, they need to apply the *Data Decryption* task. It produces the output for our model; the decrypted data or clear information. However, before decrypting the data some grant provision from the Access Control Module (ACM) is required. Therefore, the ACM uses the *Grant-Based Access Control* task in order to provide the grant based on credentials received from the users.

In our abstract model for the Encryption-Based Access Control to Sensor Data we distinguish tasks based on the security requirements they are able to satisfy as isolated solutions. Therefore, each one of these tasks can be captured as a security pattern.

Actually, the result of a well synchronized sequential combination of those separate solutions provides the Encryption-Based Access Control to Sensor Data solution and it is described using an integration scheme.

In the next section, we provide an 'intuitive' description of each of these separate solutions carrying out one of the four basic security patterns objectives: *transferring knowledge from security experts to software developers.*

[1]The term *user* is used in this paper including different roles. A user could be: a person, a hardware device, an application, etc.

VI. Security Patterns and Integration Scheme

Following our abstract model, we identified the following three security patterns: *Sensor Encryption*, *Data Decryption* and *Grant-Based Access Control*. The integration scheme that describes our security solution entirely is named *Encryption-Based Access Control to Sensor Data*. In this section, each of them are defined in detail.

A. Sensor Encryption Pattern

Problem/requirements and context: Only legitimate users should get access to sensor data. WSN applications require confidentiality of sensor data. For instance, in the remote patient monitoring scenario introduced in section III, patient data are very sensitive and only authorized users like doctors or nurses should have access to that information.
Since sensors are usually simple and low power devices, they cannot apply complex security solutions. A suitable solution is that sensors use a low cost processing encryption function.
Solution: The *Sensor Encryption Pattern* provides a low cost encryption functionality and defines the following operations:

- **generateKey**: The input for this function is defined as a list of parameters, because each particular implementation could use a different number and type of parameters. The output is the encryption key.
- **encryptData**: This function provides the encrypted data. The input for this function are the clear data and the encryption key. The output is the encrypted data.
- **sendEncryptedData**: This function sends the encrypted data. It takes as input parameter the encrypted data and the output is a boolean flag which indicates if the operation was successful.

Pre-Conditions:

- The sensors must have enough processing capacity (e.g. memory, CPU, battery) to apply the encryption function.
- Sensors are able to get the information to generate the key (e.g. cryptographic material) by a secure channel.

Properties: Confidentiality (restricted access to information only to authorized users).
Features: Low cost encryption function.
Consequences: If the *Sensor Encryption Pattern* is applied, more resources are needed within the sensors (memory to store the keys, CPU to compute the key and encrypt the data,...). Also, message overhead increases when sending the encrypted data. In addition, more overhead is generated due to the necessity of distributing and revoking cryptographic material.

B. Data Decryption Pattern

Problem/requirements and context: Users requesting access to encrypted data need to decrypt them. Several approaches require that users receive some kind of grant to be able to create a decryption key. Therefore, this pattern is focusing on data decryption based on grant provision. This pattern could be applied in many fields, in particular one of them is users accessing sensor data (e.g. remote monitoring patient scenario).

Pattern: Grant Based Access Control Pattern

Provided Properties
Property: Access_Control

Pattern Features
Feature: Grant-based_access_control

Interface
Operations
Operation: receiveGrantRequest
input credentials [ ]: Text
output result: Boolean
Operation: computeAuthLevel
input credentials [ ]: Text
Output auth_level: Text
Operation: computeGrant
input auth_level: Text
output grant [ ]: Text *//A grant could be a set of cryptographical items*
Operation: sendGrant
input grant [ ]: Text
output result: Boolean

Preconditions
pre-condition Initial definition of access control policy is required. It maps (i) credentials to authorization levels and (ii) authorization levels to grants.
pre-condition An entity applying this pattern must be able to distribute cryptographical material to sensors and users.
pre-condition Communications channels are secure if required.

Fig. 2. Grant-Based Access Control Pattern

Solution: This solution comprises the reception of a grant containing the cryptographic material to generate the suitable key. The generated key is used to decrypt the data. We propose the following operations to solve the problem:

- **receiveGrant**: This function is used to receive a grant. This grant is defined as a list of parameters which are used to generate the decryption key. The output is a boolean which indicates if the reception was successful.
- **generateKey**: This operation generates the decryption key. It receives as input the grant, which is a set of parameters and generate as output the key itself.
- **decryptData**: This function decrypts the data, it takes as inputs the encrypted data and the key and provides the decrypted data as output.

Pre-conditions:

- Users are able to get the encrypted data.
- Users are able to get the cryptographic material.
- Communications channels should be secure if required.

Properties: Disclosing encrypted data.
Feature: None.
Consequences: More complex devices are needed because they must implement decryption mechanisms. This pattern leads to extra communications necessary to communicate with the ACM for getting the grants.

C. Grant-Based Access Control Pattern (GBAC)

Problem/requirements and context: Many security solutions rely on a central entity which provides cryptographic material to users based on their credentials. This security solution enables the establishment of different authorization levels.

For each authorization level, there exists a grant providing access to legitimate users based on need-to-know-principle. For instance, according to section III, the doctor is granted higher access privilege than the nurses.
This pattern could be applied by a central entity which in the proposed solution is the ACM.
Solution: After receiving the credentials, the authorization level of the requester is computed. Based on that level, the user is granted the suitable cryptographic material to be able to decrypt the ciphered data.

- **receiveGrantRequest**: This function receives the credentials provided by a given requester. These credentials are a list of parameters. This operation takes as input the credentials and returns as output a boolean flag indicating if the reception was successful.
- **computeAuthLevel**: This operation calculates an authorization level based on the received credentials. It takes as input the credential and provides as output an authorization level.
- **computeGrant**: This function computes a grant from a given authorization level. The input parameter is an authorization level and the output is a grant.
- **sendGrant**: This function sends the grant. It takes as input parameter the grant and returns a boolean value indicating if the operation was successful.

Pre-Conditions:

- Initial definition of access control policy is required. It maps (i) credentials to authorization levels and (ii) authorization levels to grants.
- An entity applying this pattern must be able to distribute cryptographic material to sensors and users.
- Communications channels are secure if required.

Properties: Access control.
Features: Grant-based access control.
Consequences: Additional messages for grant request and sending. Processing efforts to evaluate the access control policies and mapping credentials to authorization levels and authorization levels to grants.

Eventually, the provided security properties can be compromized. To take accordingly required countermeasures, we need to supervise the behavior of this pattern. We adopt the work in [13] that is actually used in the Serenity's patterns to fulfill this activity.
Figure 2 is presenting a schematic view of this pattern which can be also used for the previous defined patterns.

D. Integration Scheme: Encryption-Based Access Control to Sensor Data

This integration scheme describes the full solution made of a combination of the defined security patterns. It synchronizes the operations among the patterns in order to provide the desired security solution.

The sequence of the IS operations for the proposed solution are provided in table I.

TABLE I
IS OPERATIONS

```
Sensor Encryption Pattern ← SEN_ENC
Data Decryption Pattern ← DATA_DEC
Grant Based Access Control ← GBAC
SEN_ENC.generateKey (in:parameters[], out:key);
SEN_ENC.encryptData (in:data, in:key, out:encryptedData);
SEN_ENC.sendEncryptedData(in:encryptedSata, out:sendSuccess);
GBAC.receiveGrantsRequest(in:credentials[], out:receiveSuccess);
If receiveSuccess is True then
    GBAC.computeAuthLevel(in:credentials[], out:authLevel);
    GBAC.computeGrant(in:authLevel, out:grant[]);
    GBAC.sendGrant(in:grant[], out:sendSuccess);
end if
DATA_DEC.receiveGrant(in:grant[], out:receiveGrantSuccess);
If receiveGrantSuccess is True then
    DATA_DEC.generateKey(in:grants[], out:key);
    DATA_DEC.decryptData(in:encryptedData, in:key, out:data);
end if
```

The IS contains the security property for the complex security solution: **sensor-to-user end-to-end confidentiality**. In addition, this IS describes its list of components (patterns) used for providing the complex solution. All the pre-conditions defined in the embedded patterns have to be considered before IS can be applied.

VII. RELATED WORK

In the introductory section of this paper, we presented the reasons for describing our solution with the Serenity's scheme for security patterns. Hereafter, we show the alternatives of the patterns' approach. Actually, several approaches going from component-based to multi agent systems have been proposed for this problem in the literature.

Component-based software engineering covers studies from the software engineering discipline that accentuate on the need for the reusability aspect in system engineering. They focus on dividing well engineered systems into elementary functional components with well-defined interfaces used for communication [14], [15], [16]. Several 'oversimplified' component-based security models have been proposed in the literature based on high level (management-oriented) security mechanisms, like those based on multi-level security [17], not suitable to our constraints.

The agent paradigm is conceived for highly distributed environments required within the fields of IT up to cognitive science where independent components from different owners coexist and interact. In [18], using the agent paradigm, the authors present a methodology that considering the organizational structures of agent systems, designs the abstract models of agents in a top-down manner. This method could cope with simple access control solutions, but with requirements such as in the context of WSN the efficiency of this paradigm is quite limited in its description of the details of the security solution.

In [19] the authors presented a security architecture that system administrators, users, and application developers can use to compose secure systems. This architecture is designed to support the dynamic composition of systems and applica-

tions from individual components, but it lacks flexibility and is restricted to access control models.

VIII. Conclusions and Future Work

In this paper, we use security patterns for the definition of an abstract model for encryption-based access control to sensor data. This security mechanism addresses an increasing demand regarding secure integration of WSN to business applications. To that effect, we introduce three security patterns: *Sensor Data Encryption*, *Data Decryption* and *Grant-Based Access Control*. We then combine these three security patterns to an integration scheme in order to capture the *Encryption-Based Access Control*.

We will use the security analysis tool presented in [20] for formally validating our security protocol using Dynamic Asynchronous Product Automata.

Moreover, we planned three different prototypes to make the validate new access control scheme exploitable. It actually shows clearly that the security pattern approach, which we have chosen to describe an abstract solution, is helpful when implementing the security scheme.

Our first prototype is implemented in Java and collects sensor data from the WSN simulator Ptolemy II [21]. The sensor encryption is part of the Ptolemy II sensor model, whereas the Access Control Module is being implemented in Java on SAP Netweaver CE 7.1. Our second implementation is under development in collaboration with CISCO System France. We focus on energy consumption monitoring and use Xbow Micaz 802.15.4 motes with MTS300/310 sensor board to measure environmental information such as temperature, light, etc. The nodes run TinyOS and therefore the sensor encryption pattern is implemented in NesC. The ACM run on a Xbow NB100 Startgate Netbridge and is implemented in C. We also plan to implement this scheme within the WASP project, with Imperial College of London nodes[22] for remote patient monitoring.

The different implementations validate our approach to compose the solution out of three different security patterns. All prototypes use the same implementation for the Data Decryption pattern which is integrated in business applications on SAP NetWeaver. The Ptolemy II and WASP implementation can use the same ACM. The sensor encryption part has the most variation, related to different node hardware and operating system. This shows that the Serenity's approach helps not only the application programmer to apply security to his system, it helps also the security expert to define a structured solution with reusable components based on generic interfaces.

Acknowledgment

This work is partially funded by the European Commission under the Framework 6 IST Projects "Wirelessly Accessible Sensor Populations (WASP)" and "SERENITY", and by the Spanish Government through the project IMPROVISA (TSI2005-07384-C03-027).

References

[1] M. Schumacher, E. Fernandez-Buglioni, D. Hybertson, F. Buschmann, and P. Sommerlad, *Security Patterns : Integrating Security and Systems Engineering (Wiley Software Patterns Series)*. John Wiley & Sons, March 2006.

[2] WASP, "WASP (Wirelessly Accessible Sensor Populations), IST 034963," 2006. [Online]. Available: www.wasp-project.org

[3] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: a survey," *Computer Networks (Amsterdam, Netherlands: 1999)*, vol. 38, no. 4, pp. 393–422, 2002. [Online]. Available: citeseer.ist.psu.edu/akyildiz02wireless.html

[4] A. Sorniotti, R. Molva, and L. Gomez, "Efficient access control for wireless sensor data," *19th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC 2008). Accepted for publication.*

[5] A. Mana, C. Rodolph, G. Spanoudakis, V. Lotz, F. Massacci, M. Molideo, and J. S. Lopez-Cobo, *Security Engineering for Ambient Intelligence: A Manifesto*. IGI Publishing, 2007.

[6] M. Schumacher, *Security Engineering with Patterns: Origins, Theoretical Models, and New Applications*, 2003.

[7] J. Yoder and J. Barcalow, "Architectural Patterns for Enabling Application Security," in *In Proc. of PLoP'97*, 1997.

[8] E. Fernandez and R. Pan, "A Pattern Language for Security Models," in *In Proc. of PLoP'01*, 2001.

[9] L. Compagna, P. E. Khoury, F. Massacci, R. Thomas, and N. Zannone, "How to capture, model, and verify the knowledge of legal, security, and privacy experts: a pattern-based approach," in *ICAIL*, 2007, pp. 149–153.

[10] F. Sanchez-Cid, A. Munoz, P. E. Khoury, and L. Compagna, "Xacml as a security and dependability pattern for access control in ami environments," in *Proceedings of the Ambient Intelligence Developments Conference (AmI.d 2007)*. Springer, September 2007.

[11] F. Sanchez-Cid and A. Maña., "Patterns for automated management of security and dependability solutions." *1st International Workshop on Secure systems methodologies using patterns (SPattern'07)*, 2007.

[12] F. Sanchez-Cid, A. Muñoz, P. El Khoury, and L. Compagna, "XACML as a Security and Dependability (S&D) pattern for Access Control in AmI environments," in *Ambient Intelligence Developments - AmI.d*, Sep. 2007. [Online]. Available: http://liris.cnrs.fr/publis/?id=3166

[13] G. Spanoudakis, K. Mahbub, and A. Zisman, "A platform for context aware runtime web service discovery." in *ICWS*. IEEE Computer Society, 2007, pp. 233–240.

[14] S. Becker, C. Canal, N. Diakov, J. Murillo, P. Poizat, and M. Tivoli, "Coordination and adaptation techniques: Bridging the gap between design and implementation." *ECOOP Workshop Reader, LNCS, Springer*, 2006.

[15] K. Khan and J. Han, "Composing security-aware software." *IEEE Software*, vol. 19, Issue 1, pp. 34-41, 2002.

[16] A. Brogi, J. Cámara, C. Canal, J. Cubo, and E. Pimentel, "Dynamic contextual adaptation," in *Electronic Notes in Theoretical Computer Science, Elsevier, ISSN 1571-0661. Also in proceedings of CONCUR'2006 Workshop on the Foundations of Coordination Languages and Software Architectures*, 2006.

[17] J. McDermid and Q. A. Shi, "Secure composition of systems." in *Proceedings of Eighth Annual Computer Security Applications Conference*, 1992, 112-122.

[18] M. Wooldridge, N. R. Jennings, and D. Kinny., "The gaia methodology for agent-oriented analysis and design." in *Journal of Autonomous Agents and Multi-Agent Systems*, vol. 3, 285-312, 2000,.

[19] T. Jaeger, J. Liedtke, V. Pantellenko, Y. Park, and N. Islam, "Security architecture for component-based operating system," *ACM Special Interest Group in Operating Systems (SIGOPS) European Workshop*, vol. 118, 1998.

[20] S. Gurgens, P. Ochsenschlager, and C. Rudolph, "ole based specification and security analysis of cryptographic protocols using asynchronous product automata," *International Workshop on Trust and Privacy in Digital Business*, vol. 106, no. 2, 2002.

[21] E. A. Lee and S. Neuendorffer, "Tutorial: Building Ptolemy II Models Graphically," EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2007-129, Oct 2007. [Online]. Available: http://www.eecs.berkeley.edu/Pubs/TechRpts/2007/EECS-2007-129.html

[22] Imperial College of London, "Sensor nodes," 2008. [Online]. Available: http://ubimon.doc.ic.ac.uk/bsn/index.php?article=167