Policy-based Access Control for DPWS-enabled Ubiquitous Devices

Konstantinos Fysarakis, Ioannis Papaefstathiou
Dept. of Electronic & Computer Engineering
Technical University of Crete
Chania, Crete, Greece
kfysarakis@isc.tuc.gr, ypg@mhl.tuc.gr

Konstantinos Rantos
Dept. of Computer and Informatics Engineering
Eastern Macedonia and Thrace Institute of Technology
Kavala, Greece
krantos@teikav.edu.gr

Charalampos Manifavas
Dept. of Informatics Engineering
Technological Educational Institute of Crete
Heraklion, Crete, Greece
harryman@ie.teicrete.gr

Othonas Sultatos
Dept. of Computer Science
University of Crete
Heraklion, Crete, Greece
sultatos@csd.uoc.gr

***Abstract*—As computing becomes ubiquitous, researchers and engineers aim to exploit the potential of the pervasive systems in order to introduce new types of services and address inveterate and emerging problems. This process will, eventually, lead us to the era of urban computing and the Internet of Things; the ultimate goal being to improve our quality of life. But these concepts typically require direct and constant interaction of computing systems with the physical world in order to be realized, which inevitably leads to the introduction of a range of safety and privacy issues that must be addressed. One such important aspect is the fine-grained control of access to the resources of these pervasive embedded systems, in a secure and scalable manner. This paper presents an implementation of such a secure policy-based access control scheme, focusing on the use of well-established, standardized technologies and considering the potential resource-constraints of the target heterogeneous embedded devices. The proposed framework adopts a DPWS-compliant approach for smart devices and introduces XACML-based access control mechanisms. The proof-of-concept implementation is presented in detail, along with a performance evaluation on typical embedded platforms.**

Keywords—access control; authorization; XACML; DPWS; security; ubiquitous computing

I. INTRODUCTION

Ubiquitous devices in the form of limited-resources interconnected embedded systems are constantly gaining popularity. Such devices nowadays offer real-time Web-accessibility and have the capability to provide various services to remote parties when deployed as Wireless Sensor Networks (WSNs) nodes. Although such connectivity has paved the road to many new applications boosted from technological advances in processors, memory and communication technologies, it has raised many privacy and security concerns.

In many environments, these networks of devices manage well-defined services and information that requires protection from malicious entities attempting to perform unauthorized access and modifications. The information they manage might be safety critical, sensitive, or confidential, and therefore constitutes an attractive target. Their protection necessitates the deployment of robust mechanisms that will, among the others, control access to participating nodes' resources and prevent data breaches. This is not a trivial task though considering the severe resource limitations of some of these nodes which prohibit the deployment of computationally expensive and high-demanding mechanisms. Moreover, parameters related to the environment that these nodes operate also have to be considered. Unattended nodes operating in hostile environments should not be left with the responsibility to make critical decisions on requests issued by remote third parties regarding access permission to their resources. These nodes are subject to physical compromise. In this case it is essential not to expose any unnecessary functionality to unauthorized entities to protect the whole system. Besides that, these nodes might not even have the capacity to handle such requests and make appropriate decisions. In others, it might not be appropriate to consume valuable energy resources on the decision making process. Such requests should therefore be off-loaded to more powerful, protected and authorized nodes that have the capability and functionality to handle them and make decisions based on robust access control mechanisms

The scheme proposed in this paper addresses the above requirements and defines a policy-based access control (PBAC) mechanism based on eXtensible Access control Markup Language (XACML, [1]) policies. It provides serving nodes the ability to control access to their resources based on policy constraints set by the system owner. Considering that nodes participating in a WSN might not have the computing resources to accommodate expensive mechanisms the core decision process is undertaken by more powerful nodes expected to operate within node's trusted environment. Such an approach allows requests to be directly addressed to the node in question, while maintaining the capability to centrally control access to said nodes.

The proposed architecture can be considered a generic policy based access control model which can be easily adapted to meet certain requirements of various urban computing related scenarios. It is based on standardized mechanisms, thus allowing new devices to easily join existing networks and offer services protected by a predefined or dynamic policy set. It facilitates use of smart devices for enhancing citizens' lives while preserving their privacy and addressing their security concerns. Based on the policy rules set by the system owner, the proposed architecture provides fine-grained access control in ubiquitous computing and the Internet of Things (IoT).

The service oriented nature of the proposed scheme is accomplished by the use of the Devices Profile for Web Services (DPWS, [2]), an OASIS standard which allows the deployment of devices aligned with the Web Services technologies, thus facilitating interoperability among services provided by resource-constrained devices.

This work is organized as follows: Section 2 identifies related work and application areas where the proposed scheme can be deployed in the context of ubiquitous devices, Section 3 details the architecture of the proposed access control framework while Section 4 includes lower-level implementation details. In Section 5 a performance evaluation of the proof-of-concept implementation is presented, while Section 6 includes the concluding remarks and key areas for future work.

II. Access Control and Smart Devices

Access control for wireless sensor networks, which is only a subset of ubiquitous devices, has been studied in the context of authentication and authorization schemes and on enhancing privacy features on basic access control models. Such schemes can be found in [4],[5],[6],[7],[8],[9] and [10]. Some of the proposed mechanisms are based on the use of public-key cryptography, a choice that is very computationally expensive for resource-constrained ubiquitous devices.

More importantly, little work has been carried out on policy-based access control for ubiquitous devices. The EU-project Internet-of-Things Architecture (IoT-A) worked on the adoption of XACML in the Internet of Things [11]. The proposed architecture is a generic model whose functional components are mapped to a set of well-defined components that comprise the IoT-A. The authors use a logistics scenario for demonstration purposes. Such an environment, however, has different requirements compared to the smart-cities oriented models examined in this paper. Moreover, no actual implementation could be identified. Another European project, namely Internet-of-Things-at-Work (IoT@Work), demonstrated the use of XACML for a capability based authorization access control system [12][13], yet for harnessing IoT technologies in industrial and automation environments. Authors in [14] present a related secure service infrastructure, utilizing XACML and DPWS. The focus of said work is smart home networks and the distribution of e.g. multimedia content. Thus, deployment on resource-constrained devices is not considered (e.g. public-key cryptography is used), nor is the scalability needed for large scale deployments on other smart environments; the consequences of some of these design choices are also evident in the performance evaluation that accompanies the cited work.

There are many examples that can demonstrate the applicability of the proposed policy-based architecture.

a. Energy-related devices, ranging from information and services provided by metering devices to remote control of smart appliances to control power consumption, can benefit from the use of policy based access control. The proposed PBAC scheme can be used to allow for a fine-grained access control to all the functional (e.g. "kill-switch") and non-functional (e.g. monitoring) elements of the devices. Moreover, the underlying technologies (i.e. XACML and DPWS) are fitting for the large-scale deployments of this application area.

b. A smart vehicle may feature various hosted services (e.g. a temperature service, a location service, a fuel consumption service, an engine-failure report service etc.), access to which can be controlled via the proposed PBAC framework. As an example, consider the scenario where the emergency services need to gain access to certain vehicle functionality provided as a service by a node deployed in the vehicle's network, such as the vehicle's engine control unit, in order to directly issue a command to this unit to turn off the engine after proper authentication. Access to such a service might be decided by the car owner or the applicable law, in which case the corresponding policy has to be defined.

c. eHealth and remote monitoring of seniors, is another area where the proposed PBAC perfectly fits. Consider the scenario where the patient has multiple physiological sensors and actuators deployed to monitor his/her condition and react based on patient's reported medical condition. One of the critical security requirements in this scenario is providing authorized-only access to these resources. Only medical staff is supposed to read this information and only authorized doctors shall be able to modify actuator's (e.g. insulin pump) status. In this model PBAC provides the necessary fine-grained access control to resources.

III. Proposed Architecture

The secure policy-based access control (PBAC) framework presented in this work facilitates the control of access to devices and their resources via security policies residing on resource-rich infrastructure nodes. It consists of several components that run on different nodes of the architecture. These components are:

- Policy Enforcement Point (PEP): Performs access control, by making decision requests and enforcing authorization decisions
- Policy Decision Point (PDP): Evaluates requests against applicable policies and renders an authorization decision

- Policy Information and Policy Administration Point (PIP/PAP): Acts as a source of attribute values. Creates and manages policies or policy sets

A node, depending on its capabilities and the available resources, might include one or more of these functional components.

XACML is used to convey policy requirements in a unified and unambiguous manner, which fits well into the model of a network of heterogeneous embedded systems where access to resources is provided by nodes as a service. Thus, XACML defines the structure and content of access requests and responses exchanged among PEPs and PDPs. In terms of mechanism(s) used to transfer these messages, Service Oriented Architectures (SOAs) provide many alternatives that can be chosen to convey policy information, such as the use of SOAP (Simple Object Access Protocol) and HTTP protocols. In this work, the typical policy based access control architecture, combined with XACML, is mapped to a SOA network of nodes to provide protected access to their distributed resources.

In more detail, the PBAC framework's implementation adopts the DPWS OASIS standard (at version 1.1 since July 2009). It utilizes the relevant specifications to provide fine-grained security policy functionality while maintaining interoperability with said standard. DPWS is the "UPnP [15] for the Internet of Things"; a unified protocol platform that provides dynamic and secure discovery of devices and Web Services (including messaging, description, interactions, event-driven changes etc.) on resource constrained devices. It also features wide support by various computing industry leaders and modern operating systems. While UPnP and DLNA (Digital Living Network Alliance) are favored for home entertainment scenarios, DPWS is recommended for enterprise and vertical applications. By adopting a DPWS-compliant mechanism, the PBAC framework offers seamless integration of new devices and good scaling.

By combining the above technologies, the proposed scheme allows for fine-grained, policy-based control of all smart resources (e.g. DPWS-enabled cameras or sensors or control stations) from remote locations, via any compatible app developed for the purpose or even typical browsers and off the shelf mobile phones. Said access may be used to access the resources provided (e.g. sensor data or video stream), update settings or even receive alerts (e.g. in case of emergencies), all based on what the active policy dictates.

IV. Implementation Details

There are many open-source implementations of the XACML handling and decision-making process that can be utilized for the proposed architecture. Such resources include Sun's XACML implementation [16], PicketBox XACML [17] (formerly JBossXACML), the Holistic Enterprise-Ready Application Security Architecture Framework (Heras AF) XACML [18] and the Enterprise Java XACML project [19]. Considering the above options, the authors chose Sun's XACML for this implementation, as it remains popular among developers and is actually the basis of various current open source and commercial offerings.

All of the framework's entities are implemented using DPWS. This facilitates the discovery [20] and description of the devices involved, and also offers control and eventing mechanisms [21] which assist in the communication of the necessary information among the entities. In terms of open–source resources aimed at developing DPWS-compliant applications for resource-constrained devices, Service-Oriented Architecture for Devices (SOA4D, [22]) provides development toolkits in C and Java. Alternatively, Web Services for Devices (WS4D, [23]) is another open source initiative which provides a number of toolkits [24] for various platforms. The authors' APIs of choice is the WS4D-JMEDS (Java-based) stack [26] as it is the most advanced and active work of the WS4D initiative, supporting almost all of the existing DPWS features and providing portability to a wide range of platforms.

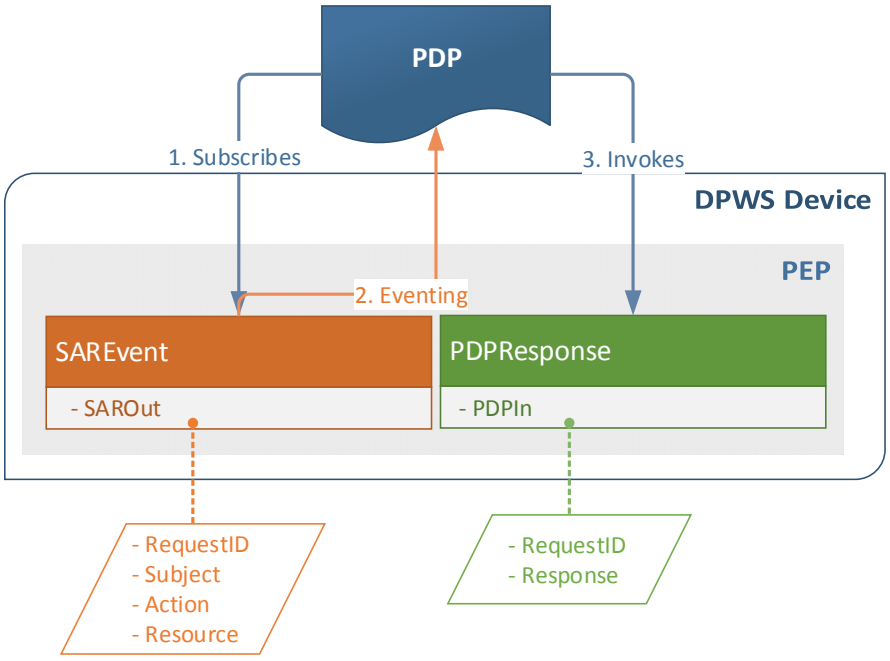

Fig. 1. PEP-to-PDP implementation

The exact implementation of the framework's entities and their communications are detailed in the sections below.

A. PEP-to-PDP

The Policy Enforcement Point must reside on every device with resources that must be protected from unauthorized access. Other than the functional elements of the devices which the framework intends to protect (e.g. access to its sensors), two extra operations must be present on each DPWS device. These operations, in essence, constitute the PEP functionality and its communication with the PDP. The latter acts as a DPWS client which accesses these PBAC-specific operations. In more detail, these operations are:

- **SAREvent:** Service Access Request Event. A WS-Eventing operation to which devices can subscribe. When fired, the operation outputs "SAROut", a message which includes all the information the PDP needs to have in order to evaluate a request.
- **PDPResponse:** Policy Decision Point Response. An operation invoked by the PDP with its answer to an access request. This response is a "PDPIn" message.

The message types are defined as follows:

- ***SAROut***
  - *RequestID:* A counter of requests issued by the specific device
  - *Subject:* An actor whose attributes may be referenced to validate that a request is authorized
  - *Action:* An operation on a resource
  - *Resource:* Data/service or system component that the "Subject" attempts to access via the "Action".
- ***PDPIn***
  - *RequestID:* A counter of requests issued by the specific device. This value is used to match SAROut messages sent by the PEP to PDP responses and to avoid replay attacks.
  - *Response:* The response issued by the PDP regarding the specific request, in the form of an integer (0: "Deny", 1: "Permit", 2: "Indeterminate" or 3: "Not Applicable").

The above, including a definition of the message types, can be seen in detail in Fig. 1.

B. PDP-to-PIP/PAP

In terms of the discovery and information exchange that must take place between infrastructure entities (PDP, PIP, PAP), an extra operation must reside with the entity that stores the active policy set (namely the PIP/PAP). In specific, this extra operation is formed as follows:

- **PIPOperation:** Policy Information Point Operation. Features an input in the form of a "PIPIn" message and an output in the form of a "PIPOut" message. The former is the request issued by the PDP (requesting all applicable policy rules), while the latter contains all the information, i.e. policies and rules pertinent to the specific request, the PIP has identified.

The above message types, are detailed below:

- ***PIPIn***
  - *RequestID:* A counter of requests issued by the PDP to the PIP
  - *Request:* The request to be evaluated in the form of a string. Upon receipt, the PIP enriches the request (with current time and date etc.) and it passes the request to a "policyFinder" module to find the appropriate policies.
- ***PIPOut***
  - *RequestID:* A counter of requests issued by the PDP to the PIP. This value is used to match PIPIn messages sent by the PDP to PIP responses and to avoid replay attacks.
  - *PolicyResponse:* A field used to return pertinent policies
  - *StatusResponse:* In cases of errors, e.g. when no pertinent policies are identified, the exact issue is identified via this field ("Non-Applicable", "Cannot determine" or "Other error")

The above, including a definition of the message types, can be seen in detail in Fig. 2.

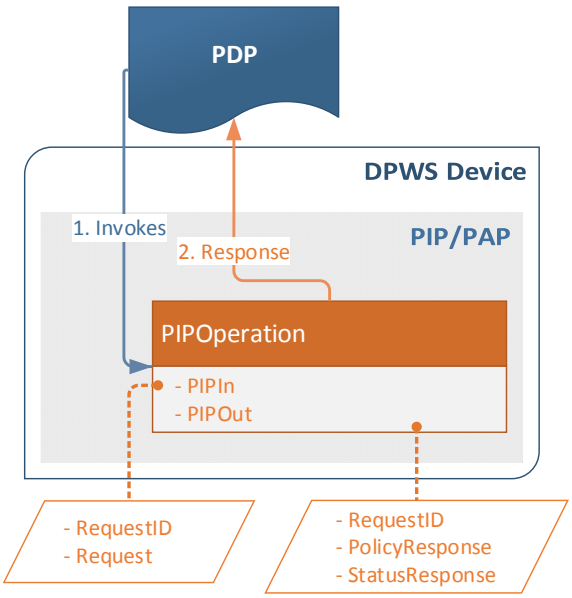

Fig. 2. PDP-to-PIP/PAP communication

C. Information Flow

The information flow that takes place whenever a request to a PBAC-protected resource is issued is as follows:

1) The PDP constantly monitors the network in order to listen to "Hello" messages that PEP-equipped Devices broadcast when initializing.
2) As soon as the PDP catches such a "Hello" message, it subscribes to the SAREvent of that PEP.
3) When a client tries to access a PBAC-protected feature on the device (i.e. invokes the protected operation), the SAREvent is fired which contains the Request ID, the Client's identifier, e.g. IP and/or username (Subject), the Invoked Operation (Action) and the Device's UUID, i.e. its Universally Unique Identifier (Resource)
4) The PDP generates an XACML request with the above data, in order to be handled and evaluated by the XACML modules of the infrastructure entities.
5) The PDP invokes the PIPOperation present on the PIP/PAP, in order to retrieve applicable policies.
6) The PIP replies with all applicable policies, otherwise it returns the error status.
7) The PDP then decides, based on information retrieved by the PIP, and invokes the PDPResponse operation on the Device, transmitting the result of the query's evaluation.

It should be noted that, regarding policy look-ups, the authors chose to implement the system so that the PEP checks with the PDP for every single request. This is essential when considering scenarios where policies change dynamically (even in an automated fashion when certain conditions are triggered), and where it is desirable to have the access control system enforce said changes in real-time. Conversely, in deployments where policy changes are expected to be infrequent or less dynamic in nature, access tokens with a predetermined validity period could be introduced to reduce the load on the PDP, e.g. as defined in [25].

D. Message Protection

The above access control mechanisms can be of limited efficacy if the actual associated messaging is not protected. Malicious entities can eavesdrop, replay or tamper with the framework's messaging, potentially overriding the offered protection. One may consider deployments of the framework's entities over trusted and/or secure networks (e.g. a VPN), but an alternative mechanism has to be considered for deployments where this is not the case.

The Web Services Security Specification (WS-Security or WSS) [27] is part of the WS-* family of specifications published by OASIS. The protocol specifies enhancements to existing SOAP messaging, integrating security features in the header of SOAP messages (working in the application layer). However, the public-key security primitives detailed in this specification can impose a significant performance overhead [28]. Therefore, considering the resource-constrained nature of the devices where the PBAC framework may be deployed as well as the need to minimize its performance impact in general, symmetric cryptographic primitives were considered more appropriate.

To this end, a security mechanism based on the AES-CCM [29] authenticated encryption algorithm was implemented and was deployed to protect both PEP-PDP as well as PDP-PIP/PAP communications, using 256bit pre-shared keys. The utilization of the AES/CCM algorithm guarantees that the PBAC-related messages exchanged between the framework's entities are fully protected in terms of confidentiality, integrity and authenticity with an acceptable performance overhead (CCM requires two block cipher encryption operations per each block of an encrypted & authenticated message).

To implement this security mechanism, the message types detailed above (namely SAROut, PDPIn, PIPIn & PIPOut) are all replaced by a SecureMessage type. So, e.g., the SAREvent operation depicted in Fig. 1 now takes the form that is shown in Fig. 3 and is defined as follows:

- **SecureMessage**
  - *Payload:* The actual message transmitted (i.e. information previously included in the replaced message type, e.g. RequestID, Subject, Action, Resource in the case of SAROut messages), in encrypted form
  - *MAC:* The output of the CBC-MAC, i.e. the message authentication part of AES-CCM.
  - *RandomData:* Random data, necessary to guarantee the security of the authenticated encryption mechanisms [29].

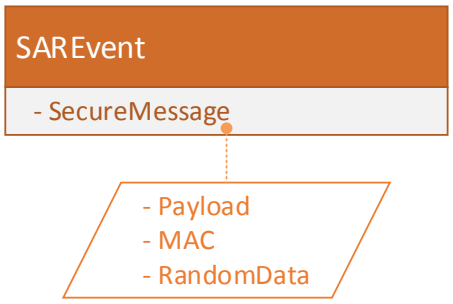

Fig. 3. Secure SAREvent

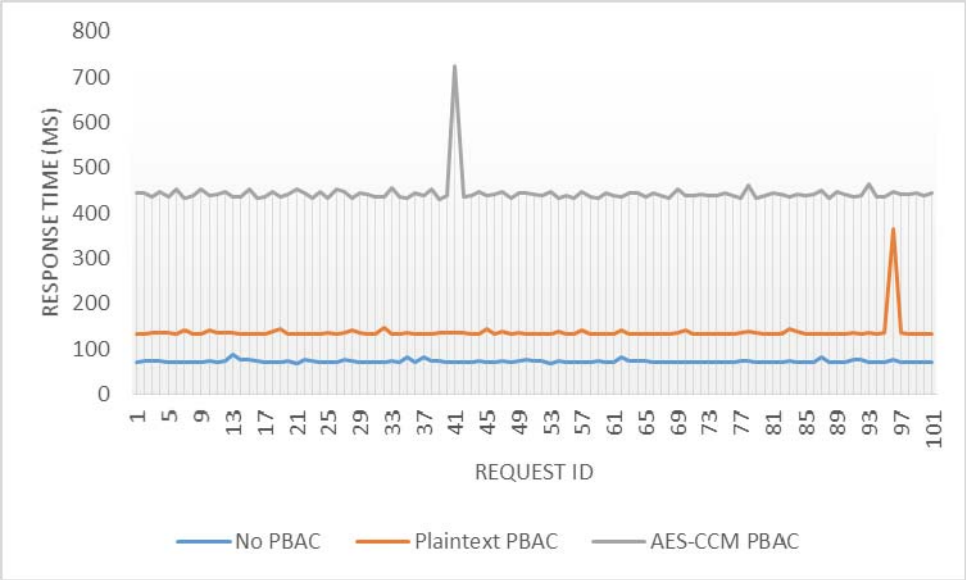

Fig. 4. Client-side Response Time

All other operations appearing in Fig. 1 and Fig. 2, i.e. *PDPResponse* and *PIPOperation*, are also adjusted accordingly.

V. Performance Evaluation

A performance evaluation was carried out in order to assess the performance overhead incurred by the proposed mechanisms on typical embedded systems that may be found in smart environments. To this end, the PEP-equipped DPWS devices were deployed on Beaglebone [30] platforms, equipped with a 720MHz ARM Cortex-A8 processor, 256MB of RAM, running on a minimal Linux-based operating system. The test-bed for the PDP and PIP devices was a Beagleboard-xM [31] platform, featuring a 1GHz ARM Cortex-A8 processor (throttled to run at 600MHz during testing), 512MB of RAM and a minimal Linux-based operating system with a lightweight window manager (LXDE, [32]). The test-bed also featured a client application developed to query the PBAC-protected DPWS devices for benchmarking purposes, which run on a PC attached to the same network via a wired LAN connection. The PDP and PIP/PAP applications are deployed as Knopflerfish [33] bundles, which is an open source service platform following the Open Services Gateway initiative (OSGi, [34]) specification. The PBAC service is expected to be deployed alongside other services on the main infrastructure systems. In view of that, the authors consider that the modular and dynamic service deployment as well as the service orchestration features provided by the OSGi framework will be advantageous in actual deployments.

For the evaluation process, two functional operations were added to the test DPWS device deployed on the Beaglebone. So, a GetStatusPBAC operation and a GetStatus operation were also implemented on the device, in addition to the PEP-related operations appearing in Fig. 1. Both these extra operations, when invoked, returned a static integer value, but the latter did so immediately while the former was PEP-protected, i.e. the test client was only allowed to invoke it if the PDP allowed so. This facilitated the evaluation of the response time overhead imposed by the PBAC framework, as any extra delay when invoking the GetStatusPBAC operation can be attributed to the PBAC mechanisms. Aiming to also weigh the impact of the security mechanisms, the assessment included tests with and without encryption on both the PEP-to-PDP link and the PDP-to-PIP link (plaintext vs. AES/CCM-protected message exchange).

A total of 100 consecutive requests were issued from the client application to the DPWS device residing on the Beaglebone. The response time recorded by the test client trying to access the device's resources appear in Fig. 4. "No PBAC" refers to the invocation of the GetStatus operation, while the "Plaintext PBAC" & "AES-CCM PBAC" columns refer to the invocation of the GetStatusPBAC operation,

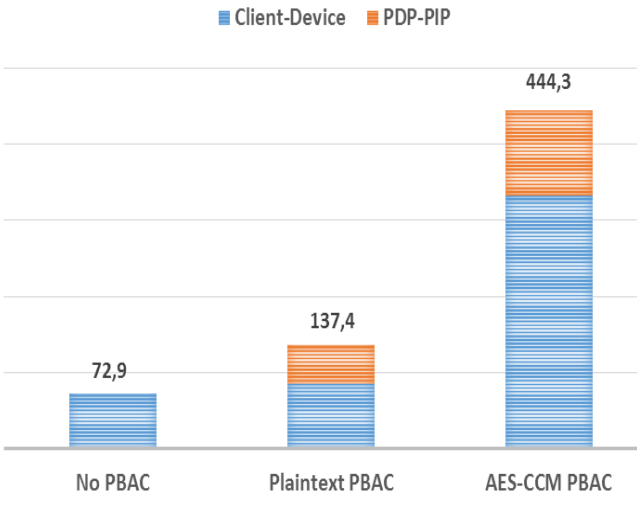

Fig. 5. Response time (ms) breakdown, averaged over 100 requests.

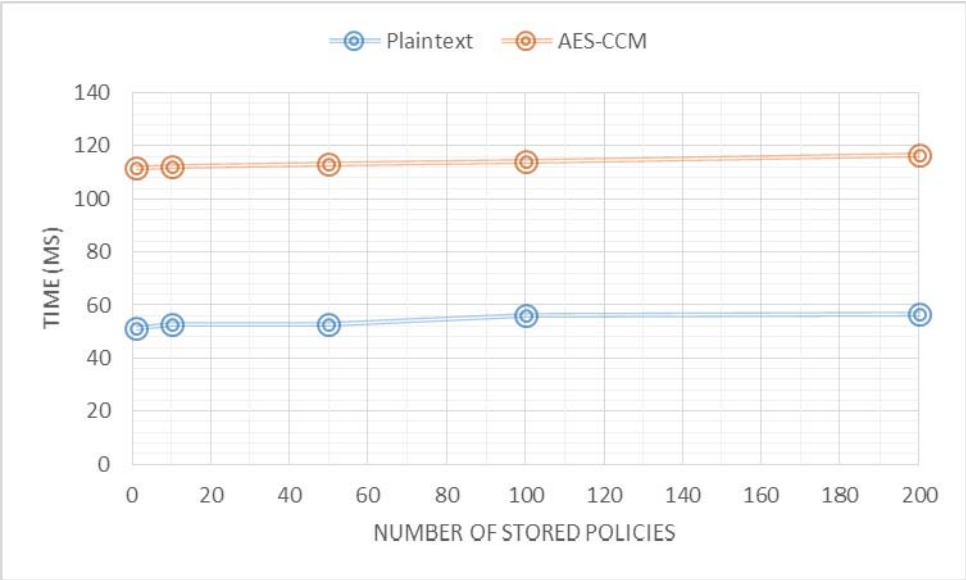

Fig. 6. PDP-PIP communication. Average response time (ms) depending on number of policies stored in PIP.

without and with AES-CCM encryption of the PBAC message exchanges, respectively. It must be reiterated that the AES-CCM response time includes the overhead introduced by the encryption mechanisms on both the PEP-to-PDP and the PDP-to-PIP/PAP communications. Random spikes on the response time can be attributed to the triggering of "housekeeping" operations of the Java-based applications running on the target platforms. A breakdown of the response times (averaged over 100 requests) can be seen in Fig. 5, where it is evident that the bulk of the delay can be attributed to the Client-Device (i.e. PEP-PDP) communication and to a lesser extent to the PDP-PIP link. As the DPWS devices featuring the PEP functionality are bound to be deployed on resource-constrained devices, the resources on the target devices themselves were monitored during testing; the results appear in Table 1.

Review of related work indicated that the number of stored policies can significantly affect the performance of the access control system [14], to the point where the response time overhead can become prohibitive in certain application scenarios. This was taken under consideration during development and, thus, the PIP stores policies in memory in the form of a hash table. The effectiveness of this approach was validated during the performance evaluation. Fig. 6 depicts the PDP-PIP communication, focusing on the time the PDP has to wait before it receives pertinent policies from the PIP, in scenarios were the number of stored policies varies from 1 to 200. As is obvious from said figure, the impact of the number of policies stored on the PIP is negligible.

TABLE 1. RESOURCE CONSUMPTION ON PBAC-PROTECTED DPWS DEVICES DURING BENCHMARKS

Average	No PBAC	Plaintext PBAC	AES-CCM PBAC
CPU (%)	87,4	94,5	97,7
Memory (bytes)	28461	29909	36077

VI. CONCLUSIONS & FUTURE WORK

This paper presented an implementation of a policy-based access control scheme appropriate for resource-constrained ubiquitous devices. In the context of modern "smart environments" and the IoT, said devices should exhibit adequate protection of their resources and services they provide to remote entities. Yet, the severe limitations they often face prohibit the deployment of resource-expensive access control schemes. With the proposed scheme computationally-intensive tasks related to the decision making process are offloaded to more powerful devices that operate within the device's trusted environment. We demonstrated that standardized protocols, widely used in other environments, can also be deployed for resource-constrained devices, thus promoting interoperability and secure wide accessibility for those devices. Moreover, messages exchanged among stakeholders are sufficiently secured, hence exposure of sensitive information pertinent to the access control system or policies to unauthorized parties is properly avoided.

While this paper focused on authorization aspects of ubiquitous smart devices, another important building block is the user authentication, which will be investigated in future work. A SAML-based federated identity model would fit well into the XACML architecture presented in this work, extending authentication infrastructures typically used for protecting browser-based resources, like in [35]. Additional work can also be carried out on adapting the utilized standards to better facilitate smart city and IoT deployments in general. More specifically, XACML policies could be tailored to the needs of specific urban computing deployments and their intrinsic requirements, while more lightweight implementations of DPWS could be investigated for scenarios involving extremely resource-constrained devices (e.g. sensors, as in [36]).

ACKNOWLEDGMENT

This work has been supported by the Greek General Secretariat for Research and Technology (GSRT), under the ARTEMIS JU research program nSHIELD (new embedded Systems arcHItecturE for multi-Layer Dependable solutions) project. Call: ARTEMIS-2010-1, Grant Agreement No.:269317.

REFERENCES

[1] B. Parducci and H. Lockhart, "eXtensible Access Control Markup Language (XACML) Version 3.0," OASIS Standard, pp. 1–154, 2013.

[2] "Devices profile for web services, version 1.1," 2009. [Online]. Available: http://docs.oasis-open.org/ws-dd/dpws/1.1/os/wsdd-dpws-1.1-spec-os.pdf. [Accessed: 25-Apr-2014].

[3] "Organization for the Advancement of Structured Information Standards (OASIS)." [Online]. Available: https://www.oasis-open.org.

[4] H. Wang, B. Sheng, and Q. Li, "Elliptic curve cryptography-based access control in sensor networks". Int. J. Secur. Networks. 1, 127, 2006.

[5] M.L. Das, "Two-factor user authentication in wireless sensor networks". IEEE Trans. Wirel. Commun. 8, pp. 1086–1090, 2009.

[6] D. He, J. Bu, S. Zhu, S. Chan, and C. Chen, "Distributed Access Control with Privacy Support in Wireless Sensor Networks". IEEE Trans. Wirel. Commun. 10, pp. 3472–3481, 2011.

[7] Y. Zhou, Y. Zhang and Y. Fang, "Access control in wireless sensor networks". Ad Hoc Networks. 5, pp. 3–13, 2007.

[8] Y. Faye, I. Niang and T. Noel, "A survey of access control schemes in wireless sensor networks". Proc. World Acad. Sci. Eng. Tech. pp. 814–823, 2011.

[9] S. Yu, K. Ren and W. Lou, "FDAC: Toward Fine-Grained Distributed Data Access Control in Wireless Sensor Networks". IEEE Trans. Parallel Distrib. Syst. 22, pp. 352–362, 2011.

[10] J. Maerien, S. Michiels, C. Huygens, D. Hughes and W. Joosen, "Access Control in Multi-party Wireless Sensor Networks". In: Demeester, P., Moerman, I., and Terzis, A. (eds.) Wireless Sensor Networks SE - 3. pp. 34–49. Springer Berlin Heidelberg (2013).

[11] A. Serbanati, A. S. Segura, A. Oliverau, Y. B. Saied, N. Gruschka, D. Gessner, and F. Gomez-Marmol, "Internet of Things Architecture, Concept and Solutions for Privacy and Security in the Resolution Infrastructure". EU project IoT-A, Project report D4.2, February 2012, [Online]. Available: http://www.iot-a.eu/public/public-documents. [Accessed: 25-Apr-2014].

[12] D. Rotondi, S. Gusmeroli, S. Piccione, "A capability-based security approach to manage access control in the Internet of Things", Mathematical and Computer Modelling Journal, March 2013, ISSN 0895-7177,http://dx.doi.org/10.1016/j.mcm.2013.02.006.

[13] D. Rotondi, C. Seccia, S. Piccione. "Access Control & IoT: Capability Based Authorization Access Control System", 1st IoT International Forum, Berlin, November 2011. Available at: https://www.iot-at-work.eu/data/Capability-Based Authorization_Summary.pdf [Accessed: 9-Apr-2014].

[14] A. Muller, H. Kinkelin, S. K. Ghai, and G. Carle, "A secure service infrastructure for interconnecting future home networks based on DPWS and XACML," in 2010 ACM SIGCOMM Workshop on Home Networks, ser. HomeNets '10. New Delhi, India: ACM New York, NY, USA, 2010, pp. 31–36.

[15] "Universal Plug and Play (UPnP), ISO/IEC 29341-x." [Online]. Available: http://upnp.org/sdcps-and-certification/standards/. [Accessed: 18-Apr-2013].

[16] "Sun Microsystems Laboratories, XACML." [Online]. Available: http://sunxacml.sourceforge.net. [Accessed: 25-Apr-2014].

[17] "PicketBox XACML." [Online]. Available: https://community.jboss.org/wiki/PicketBoxXACMLJBossXACML. [Accessed: 25-Apr-2014].

[18] "Holistic Enterprise-Ready Application Security Architecture Framework (Heras AF) XACML." [Online]. Available: http://www.herasaf.org/heras-af-xacml.html. [Accessed: 25-Apr-2014].

[19] "Enterprise Java XACML." [Online]. Available: http://code.google.com/p/enterprise-java-xacml/. [Accessed: 25-Apr-2014].

[20] T. Nixon, A. Regnier, V. Modi, and D. Kemp, "Web Services Dynamic Discovery (WS- Discovery), version 1.1," OASIS, 2009. [Online]. Available: http://docs.oasis-open.org/ws-dd/discovery/1.1/os/wsdd-discovery-1.1-spec-os.pdf. [Accessed: 28-Sep-2013].

[21] D. Box, L. F. Cabrera, C. Critchley, F. Curbera, D. Ferguson, S. Graham, D. Hull, G. Kakivaya, A. Lewis, B. Lovering, P. Niblett, D. Orchard, S. Samdarshi, J. Schlimmer, I. Sedukhin, J. Shewchuk, S. Weerawarana, and D. Wortendyke, "Web Services Eventing (WS-Eventing)," W3C Member Submission, vol. 2009. pp. 1–34, 2006.

[22] "Service-Oriented Architecture for Devices (SOA4D)." [Online]. Available: http://cms.soa4d.org/. [Accessed: 28-Sep-2013].

[23] "Web Services for Devices (WS4D)." [Online]. Available: http://ws4d.e-technik.uni-rostock.de. [Accessed: 25-Apr-2014].

[24] E. Zeeb, G. Moritz, D. Timmermann, and F. Golatowski, "WS4D: Toolkits for Networked Embedded Systems Based on the Devices Profile for Web Services," in 2010 39th International Conference on Parallel Processing Workshops, 2010, pp. 1–8.

[25] H. Lockhart, B. Parducci, and E. Rissanen, "SAML 2.0 Profile of XACML, Version 2.0," OASIS, 2010. [Online]. Available: http://docs.oasis-open.org/xacml/3.0/xacml-profile-saml2.0-v2-spec-cd-03-en.pdf. [Accessed: 28-Sep-2013].

[26] "WS4D-JMEDS DPWS Stack." [Online]. Available: http://sourceforge.net/projects/ws4d-javame/. [Accessed: 25-Apr-2014].

[27] K. Lawrence, C. Kaler, A. Nadalin, R. Monzilo, and P. Hallam-Baker, "Web Services Security: SOAP Message Security 1.1," OASIS Standard Specification, 2006. [Online]. Available: https://www.oasis-open.org/committees/download.php/16790/wss-v1.1-spec-os-SOAPMessageSecurity.pdf. [Accessed: 25-Apr-2014].

[28] F. Lascelles and A. Flint, "WS-Security Performance," Websphere Journal, 2006. [Online]. Available: http://websphere.sys-con.com/node/204424. [Accessed: 25-Apr-2014].

[29] D. Whiting, R. Housley, and N. Ferguson, "Counter with CBC-MAC (CCM)," 2003. [Online]. Available: http://tools.ietf.org/rfc/rfc3610.txt.

[30] "BeagleBone System Reference Manual, RevA3_1.0." [Online]. Available: http://beagleboard.org/static/beaglebone/a3/Docs/Hardware/BONE_SRM.pdf. [Accessed: 25-Apr-2014].

[31] "BeagleBoard-xM System Reference Manual, Rev. C." [Online]. Available: http://beagleboard.org/static/BBxMSRM_latest.pdf. [Accessed: 25-Apr-2014].

[32] "Lightweight X11 Desktop Environment (LXDE)." [Online]. Available: http://lxde.org/.

[33] "Knopflerfish - Open Source OSGi service platform." [Online]. Available: http://www.knopflerfish.org/. [Accessed: 28-Sep-2013].

[34] "Open Services Gateway Initiative (OSGi)." [Online]. Available: http://www.osgi.org/. [Accessed: 28-Sep-2013].

[35] M. Anwander, T. Braun, P. Hurni, T. Staub, and G. Wagenknecht, "User and Machine Authentication and Authorization Infrastructure for Distributed Wireless Sensor Network Testbeds," J. Sens. Actuator Networks, vol. 2, no. 1, pp. 109–121, 2013.

[36] I. K. Samaras, G. D. Hassapis, and J. V. Gialelis, "A Modified DPWS Protocol Stack for 6LoWPAN-Based Wireless Sensor Networks," IEEE Trans. Ind. Informatics, vol. 9, no. 1, pp. 209–217, Feb. 2013.