

Technical University of Crete  
Electronic and Computer Engineering Department  
Microprocessor & Hardware Laboratory

---

ANALYSIS AND COMPOSITION OF SECURITY  
PRIMITIVES TOWARDS A FRAMEWORK THAT  
SAFEGUARDS THE CONFIDENTIALITY, INTEGRITY  
AND AVAILABILITY OF EMBEDDED SYSTEMS:  
USPBM – A SECURE POLICY-BASED  
MANAGEMENT FRAMEWORK FOR  
UBIQUITOUS SMART DEVICES

A DISSERTATION  
SUBMITTED TO THE DEPARTMENT OF ELECTRONIC AND COMPUTER ENGINEERING  
OF THE TECHNICAL UNIVERSITY OF CRETE  
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS  
FOR THE DEGREE OF  
DOCTOR OF PHILOSOPHY

Konstantinos Fysarakis

Supervisor: Ioannis Papaefstathiou

Chania 2016

*This page intentionally left blank.*

## ΠΕΡΙΛΗΨΗ

Πλήθος υπολογιστικών συστημάτων βρίσκονται ήδη γύρω μας, σε διάφορες μορφές – μία πραγματικότητα που επηρεάζει όλες τις πτυχές της σύγχρονης ζωής και μία τάση που αναμένεται να ενταθεί τα επόμενα χρόνια. Ερευνητές και μηχανικοί εργάζονται προκειμένου να παρουσιάσουν νέοι τύποι συσκευών και υπηρεσιών, με σκοπό να αντιμετωπιστούν αποτελεσματικότερα τα υφιστάμενα και αναδυόμενα προβλήματα της καθημερινότητας και να βελτιωθεί η ποιότητα της ζωής μας. Αυτή η διαδικασία θα οδηγήσει στην εποχή του Internet of Things (IoT), όπου όλα τα αντικείμενα που κατέχουμε και αλληλοεπιδρούμε θα αποτελούνται από υπολογιστικές συσκευές συνδεδεμένες στο διαδίκτυο.

Ωστόσο, οι μεγάλες αυτές αλλαγές δεν πρόκειται να πραγματοποιηθούν χωρίς την υπέρβαση κάποιων σημαντικών εμποδίων. Οι έξυπνες συσκευές έχουν, συχνά, άμεση επαφή με τον φυσικό κόσμο και, επιπλέον, επεξεργάζονται, αποθηκεύουν και μεταφέρουν δεδομένα ευαίσθητου προσωπικού χαρακτήρα, φέρνοντας έτσι στο προσκήνιο σημαντικά θέματα ασφάλειας και ιδιωτικότητας. Τόσο οι ερευνητές, όσο και οι επιχειρήσεις, αλλά και οι τελικοί χρήστες, αναγνωρίζουν ως ένα τέτοιο σημαντικό πρόβλημα την έλλειψη ασφαλούς, επεκτάσιμου και λεπτομερούς (fine-grained) ελέγχου πρόσβασης στα ενσωματωμένα αυτά συστήματα και τους πόρους/υπηρεσίες τους, με επίγνωση πλαισίου (context-awareness). Οι περιορισμοί των διαθέσιμων πόρων των συσκευών που ενσωματώνονται σε έξυπνα περιβάλλοντα και η ετερογένειά τους (σε υλικό, δικτύωση, εφαρμογές κλπ.), επιδεινώνουν τα προβλήματα αυτά και δυσχεραίνουν την αντιμετώπισή τους. Έτσι, συχνά συνυφασμένο με τα θέματα ασφάλειας, είναι ένα άλλο σημαντικό εμπόδιο: η έλλειψη διαλειτουργικότητας που θα διευκόλυνε τη χρήση, παρακολούθηση και διαχείριση της πληθώρας των έξυπνων συσκευών και των υπηρεσιών τους. Παρόλο που οι απρόσκοπτες αλληλεπιδράσεις μηχανής-προς-μηχανή (M2M) και ανθρώπου-προς-μηχανή (H2M) είναι αναγκαίες για ένα πραγματικό και ασφαλές περιβάλλον διάχυτης νοημοσύνης, σήμερα υπάρχει μια κατακερματισμένη αγοράς με ποικιλία ασύμβατων μεταξύ τους συσκευών.

Τα παραπάνω προβλήματα αποτέλεσαν το κίνητρο για τη διατριβή αυτή, που παρουσιάζει το uSPBM, ένα ασφαλές πλαίσιο διαχείρισης και προστασίας έξυπνων συσκευών, μέσω πολιτικών ασφάλειας, με έμφαση στη χρήση τυποποιημένων τεχνολογιών, λαμβάνοντας υπόψιν και τους περιορισμούς πόρων των συσκευών αυτών. Με το συνδυασμό του λεπτομερούς ελέγχου πρόσβασης που παρέχεται από την eXtensible Access Control Markup Language (XACML) με τα οφέλη των Service Oriented Αρχιτεκτονικών, μέσω του Devices Profile for Web Services (DPWS), επιτρέπει την απρόσκοπτη αλληλεπίδραση και τη διαχείριση, σε πραγματικό χρόνο, ετερογενών έξυπνων συσκευών, μέσω πολιτικών ασφάλειας με επίγνωση πλαισίου. Επιπλέον, το uSPBM περιλαμβάνει αρθρωτά στοιχεία που επιτρέπουν την αυθεντικοποίηση (authentication) των χρηστών και των συσκευών, επικοινωνία μεταξύ διαφορετικών, κατανεμημένων δικτύων, καθώς και αυτοματοποιημένα, πραγματικού χρόνου παρακολούθηση και διαχείριση των συσκευών, των παραμέτρων λειτουργίας τους, και των υπηρεσιών τους, μέσω διεπαφών ενδιάμεσου λογισμικού.

Το παρουσιαζόμενο έργο περιλαμβάνει proof of concept υλοποιήσεις όλων των οντοτήτων του πλαισίου σε μία ποικιλία από πλατφόρμες υλικού, συμπεριλαμβανομένων καινοτόμων εργαλείων ανάπτυξης τα οποία ξεπερνούν σε απόδοση τις προϋπάρχουσες λύσεις. Οι υλοποιήσεις αξιολογούνται λεπτομερώς σε μια σειρά από περιπτώσεις χρήσης, όπου η εφαρμογή του uSPBM ξεπερνάει την τρέχουσα τεχνολογία αιχμής από άποψη διαλειτουργικότητας, ελέγχου πρόσβασης, και παρακολούθησης και διαχείρισης σε πραγματικό χρόνο των έξυπνων συσκευών. Τα αποτελέσματα επικυρώνουν την εφαρμοσιμότητα του uSPBM και τη σημασία του στην ευρύτερη υιοθέτηση του IoT, επιτρέποντας έτσι στους χρήστες να αποκομίσουν όλα τα οφέλη της νέας αυτής πραγματικότητας.

## ABSTRACT

Computing devices already permeate working and living environments, a trend affecting all aspects of modern everyday lives, and one that is expected to intensify in the coming years. As computing becomes ubiquitous, researchers and engineers aim to exploit the potential of pervasive systems in order to introduce new types of services and address inveterate and emerging problems. This process will lead us eventually to the era of urban computing and the Internet of Things (IoT), where all objects we own and interact with will be computerized and connected to the Internet.

However, these long-promised improvements cannot be realized without overcoming some significant obstacles introduced by these technological advancements. The direct interaction smart devices often have with the physical world, along with the processing, storage and communication of data pertaining to users' lives, i.e. private sensitive in nature, bring security and privacy concerns into the limelight. Researchers, business stakeholders and end-users alike, recognize that one such important security-related barrier is the lack of fine-grained and context-aware control of access to the resources of these pervasive embedded systems, in a secure and scalable manner.

The resource-constraints of the platforms integrated into smart environments, and their heterogeneity in hardware, network and overlaying technologies, only exacerbate the above security issues. Thus, often intertwined with the security issues, is another important barrier: the lack of interoperable solutions, to facilitate the use, monitoring and management of the plethora of devices and their services. Therefore, while seamless machine-to-machine (M2M) and human-to-machine (H2M) interactions are a necessity for secure and truly ubiquitous computing, the current status quo is that of a segregated and incompatible assortment of devices.

Motivated by the above, this thesis presents uSPBM, a secure policy-based management framework, focusing on the use of well-established, standardized technologies, while considering the potential resource-constraints of the target heterogeneous embedded devices. By combining the well-studied fine-grained access control provided by the eXtensible Access Control Markup Language (XACML) with the benefits of Service Oriented Architectures, via the Devices Profile for Web Services (DPWS), it enables seamless interactions and fine-grained, context-aware policy-based management of heterogeneous smart devices. Moreover, the framework includes modular elements that allow the authentication of users and devices, communication between different domains, as well as automated, real-time monitoring and management of the devices', their operating parameters and their services, via the appropriate middleware interfaces.

The work includes proof-of-concept implementations of all of the framework's entities, on a variety of hardware platforms, including purpose-built novel development tools, which outperform existing solutions. All implementations are evaluated in detail on a number of use cases where applying the proposed framework enhances the current state of the art in terms of the interoperability, security, real-time monitoring and management of smart devices. The results validate the feasibility of uSPBM's approach and its applicability in enabling the wider adoption of the IoT, thus allowing users to reap the associated benefits.

## ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ

Έλεγχος πρόσβασης μέσω πολιτικών; Εξουσιοδότηση; Αυθεντικοποίηση; Υπηρεσίες Διαδικτύου; Αρχιτεκτονικές Υπηρεσιών; Διάχυτη Υπολογιστική; Διαδίκτυο των Αντικειμένων; Ίντερνερντ των Πραγμάτων; Devices Profile for Web Services (DPWS); eXtensible Access Control Markup Language (XACML); MQ Telemetry Transport (MQTT); Ασφάλεια; Έξυπνο σπίτι; Έξυπνα οχήματα; Δίκτυα αισθητήρων σώματος; Δίκτυα χαμηλής ισχύος και απωλειών; Ενσωματωμένα συστήματα; Έξυπνες συσκευές; Αισθητήρες;

## KEYWORDS

Policy-based Access Control; Authorisation; Authentication; Web Services; Service Architectures; Ubiquitous Computing; Pervasive Computing; Internet of Things (IoT); Devices Profile for Web Services (DPWS); eXtensible Access Control Markup Language (XACML); MQ Telemetry Transport (MQTT); Security; Smart Home; Smart Vehicles; Body Sensor Networks; Low power and lossy networks (LLNs); Embedded systems; Smart devices; Sensors;

## ACKNOWLEDGEMENTS

I would like to thank Dr. Charalampos Manifavas for the excellent collaboration throughout the years, and my advisor Dr. Ioannis Papaefstathiou for allowing me to conduct my research and providing remarkable support. I also wish to thank my colleagues and co-authors who helped design and implement the work presented herein, as well as the committee members for their valuable expertise and precious time.

Finally, I would like to extend my deepest gratitude to my family and friends; they stood by me throughout these years, each helping me in their own way, and for that, I will always be grateful.

## PUBLICATIONS

Below follow the journal and conference publications stemming from the research work presented in this dissertation, listed in reverse chronological order.

### JOURNAL PUBLICATIONS

5. "XSACd: Cross-domain Resource Sharing & Access Control for Smart Environments", K. Fysarakis, O. Sultatos, C. Manifavas, I. Papaefstathiou, I. Askoxylakis, Future Generation Computer Systems, Elsevier, 2015. (minor revision submitted) [IF: 2.786]
4. "RtVMF : A Secure Real-time Vehicle Management Framework", K. Fysarakis, G. Hatzivasilis, C. Manifavas and I. Papaefstathiou, IEEE Pervasive Computing, Special Issue - Smart Vehicle Spaces, vol.15, no.1, pp.22-30, Jan.-Mar. 2016/2015. (DOI: 10.1109/MPRV.2016.15) [IF: 2.103]
3. "Node.DPWS: Efficient Web Services for the IoT", K.Fysarakis, D. Mylonakis, C. Manifavas and I. Papaefstathiou, IEEE Software, vol.PP, no.99, pp.1-1, 2015. (DOI:10.1109/MS.2015.155). [IF: 1.23]
2. "Policy-controlled authenticated access to LLN-connected healthcare resources", K. Rantos, K.Fysarakis, C. Manifavas and I. Askoxylakis, IEEE Systems Journal, vol.PP, no.99, pp.1,11, 2015. (DOI: 0.1109/JSYST.2015.2450313) [IF: 1.98]
1. "Embedded Systems Security: A Survey of EU Research Efforts", C. Manifavas, K. Fysarakis, A. Papanikolaou and I. Papaefstathiou, Security and Communication Networks, Wiley, 2014. (DOI: 10.1002/sec.1151) [IF: 0.72]

## CONFERENCE PUBLICATIONS

9. WSACd - A Usable Access Control Framework for Smart Home Devices, K. Fysarakis, C. Konstantourakis, K. Rantos, C. Manifavas and I. Papaefstathiou, 9th WISTP International Conference on Information Security Theory and Practice (WISTP 2015), Heraklion, Crete, Greece, August 24-25, 2015. (DOI: 10.1007/978-3-319-24018-3\_8)
8. Secure and Authenticated Access to LLN Resources Through Policy Constraints, K. Rantos, K. Fysarakis, O. Soultatos and I. Askoxylakis, 9th WISTP International Conference on Information Security Theory and Practice (WISTP 2015), Heraklion, Crete, Greece, August 24-25, 2015. (DOI: 10.1007/978-3-319-24018-3\_18)
7. "RT-SPDM: Real-time Security Privacy & Dependability Management of Heterogeneous Systems", K. Fysarakis, G. Hatzivasilis, I. Askoxylakis and C. Manifavas, 3rd Int. Conference on Human Aspects of Information Security, Privacy and Trust (HAS 2015), within the 17th Int. Conference on Human-Computer Interaction (HCI 2015), Los Angeles, USA, 2-7 Aug. 2015. (DOI: 10.1007/978-3-319-20376-8\_55)
6. "Policy-based Access Control for DPWS-enabled Ubiquitous Devices", K. Fysarakis, K. Rantos, O. Sultatos, C. Manifavas and I. Papaefstathiou, 19th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA 2014), Barcelona, Spain, Sept. 16-19, 2014. (DOI: 10.1109/ETFA.2014.7005233)
5. "Policy-based Access Control for Body Sensor Networks", C. Manifavas, K. Fysarakis, K. Rantos, K. Kagiambakis and I. Papaefstathiou, 8th Workshop in Information Security Theory & Practice (WISTP 2014), Heraklion, Crete, Greece, Jun. 30 - Jul. 02, 2014. (DOI: 10.1007/978-3-662-43826-8\_11)
4. "Embedded Systems Security Challenges", K. Fysarakis, K. Rantos, A. Papanikolaou, G. Hatzivasilis and C. Manifavas, Measurable security for Embedded Computing and Communication Systems (MeSeCCS 2014), within the International Conference on Pervasive and Embedded Computing and Communication Systems (PECCS 2014), Lisbon, Portugal, Jan. 7-9, 2014. (DOI: 10.5220/0004901602550266)
3. "A Lightweight Anonymity & Location Privacy Service", K. Fysarakis, A. Adamopoulos, C. Manifavas, I. Papaefstathiou, 13th IEEE Symposium on Signal Processing and Information Technology (ISSPIT), Athens, Greece, Dec. 12-15, 2013. (DOI: 10.1109/ISSPIT.2013.6781866)
2. "Lightweight Cryptography for Embedded Systems – A Comparative Analysis", G. Hatzivasilis, K. Fysarakis, C. Manifavas and K. Rantos. Accepted. 6th International Workshop on Autonomous and Spontaneous Security (SETOP 2013), Egham, UK, Sep. 12-13, 2013. (DOI: 10.1007/978-3-642-54568-9\_21)
1. "Secure Policy-Based Management Solutions in Heterogeneous Embedded Systems Networks", K. Rantos, A. Papanikolaou, K. Fysarakis and C. Manifavas, IEEE International Conference on Telecommunications & Multimedia (TEMU 2012), Heraklion, Greece, Jul. 30 – Aug. 1, 2012. (DOI: 10.1109/TEMU.2012.6294723)



# CONTENTS

Περίληψη.....	3
Abstract .....	4
Λέξεις Κλειδιά.....	5
Keywords .....	5
Acknowledgements .....	6
Publications .....	7
Journal Publications .....	7
Conference Publications .....	8
Contents .....	9
List of Tables.....	12
List of Figures.....	13
1. Introduction.....	15
1.1 Motivation.....	16
1.2 The Framework .....	20
1.3 Use Cases .....	22
1.3.1 Energy / Smart Metering .....	22
1.3.2 Smart Vehicles .....	23
1.3.3 e-Health .....	23
1.3.4 Transport Infrastructure .....	24
1.3.5 Smart Home .....	24
1.3.6 Ambient Assisted Living.....	25
1.3.7 Smart Factory (Industry 4.0) .....	26
2. Background & Related Work .....	27
3. Core Technologies & Entities.....	34
3.1 Policy-Based Access Control.....	34
3.1.1 Typical Application Scenario & Data Flow .....	35
3.1.2 Access Control Policies .....	37
3.2 Service-Oriented Architectures.....	43
3.2.1 The Devices Profile for Web Services (DPWS) .....	43
3.2.2 Node.DPWS .....	46
3.3 Combining the Technologies.....	53
4. Implementation Approach .....	54
4.1 Node Classification.....	54

4.1.1	Power Devices .....	54
4.1.2	Mobile Devices .....	55
4.1.3	Embedded/Micro Devices .....	56
4.1.4	Sensors/Nano Devices .....	57
4.2	DPWS Implementation of Access Control Mechanisms.....	60
4.2.1	PEP-PDP Communication.....	60
4.2.2	PDP-PIP/PAP Communication.....	61
4.2.3	Information Flow .....	62
4.3	Message Protection .....	63
4.3.1	Asymmetric Variant .....	64
4.3.2	Symmetric Variant .....	66
4.3.3	Security Analysis .....	69
4.4	Interfacing with Middleware & Management Systems .....	71
4.5	Performance Evaluation of Core Entities .....	74
5.	Applications & Extensions .....	80
5.1	Body Sensor Networks .....	80
5.1.1	Motivation .....	80
5.1.2	Proposed Architecture.....	82
5.1.3	Implementation Approach .....	84
5.1.4	Proof of Concept.....	85
5.1.5	Summary.....	90
5.2	Authenticated Access to LLN-Connected Resources.....	90
5.2.1	Motivation .....	91
5.2.2	Proposed Architecture.....	93
5.2.3	Implementation Approach .....	96
5.2.4	Performance Evaluation .....	97
5.2.5	Security Considerations .....	101
5.2.6	Summary.....	102
5.3	Cross-Domain Smart Environments –The XSACd Variant .....	103
5.3.1	Motivation .....	104
5.3.2	Proposed Architecture.....	105
5.3.3	Implementation Approach .....	106
5.3.4	Event sequence.....	107
5.3.5	uSPBM/XSACd Cross-domain Proxy.....	108
5.3.6	Security considerations .....	112
5.3.7	Performance Evaluation .....	112
5.3.8	Summary.....	117

5.4	Smart Vehicles – The RtVMF Framework.....	117
5.4.1	Motivation .....	118
5.4.2	The RtVMF Architecture .....	119
5.4.3	Proof of Concept.....	123
5.4.4	Summary.....	128
6.	Conclusions & Future Work.....	129
7.	References .....	133
	Annex A – Embedded Systems Security .....	158
	Physical Security Issues.....	158
	Access Control .....	160
	Cryptographic Mechanisms .....	162
	Network Protocol and Management Issues .....	166
	Annex B – Pertinent EU-funded Research .....	171
	Node Technologies.....	175
	Hardware-Related Security Modules.....	176
	Virtualisation .....	178
	Lightweight Cryptography .....	179
	Miscellaneous Node Topics .....	180
	Network Technologies .....	180
	Node Attestation and Authentication .....	182
	Privacy and Anonymity.....	182
	Secure Routing.....	184
	Intrusion and Malicious Node Detection.....	185
	Secure Aggregation.....	186
	Miscellaneous Network Topics .....	187
	Middleware and Overlay Technologies.....	187
	Trusted Middleware .....	189
	Service-Oriented Middleware .....	189
	Context-Aware Middleware .....	190
	Reconfigurable and Fault-Tolerant Middleware.....	190
	Overlay Applications.....	191
	Architectures and Formalisation .....	192
	Identified Open Issues.....	194

## LIST OF TABLES

Table 1. uSPBM Compared to Recent Related Work.....	32
Table 2. Policy Attributes Template .....	38
Table 3. Comparison of SDPs.....	46
Table 4. Overview of DPWS Toolkits .....	47
Table 5. Prototype Platform Specifications .....	58
Table 6. WS-Security and TLS Benchmark Results for 25 Concurrent Requestors [133].....	66
Table 7. Resource Consumption on uSPBM-Protected DPWS Devices During Benchmarks .....	77
Table 8. Performance & Reliability Comparison of MQTT and HTTP on Typical Mobile Application...	111
Table 9. Scenario steps .....	125
Table 10. Selected EU-funded projects related to embedded systems security. ....	171
Table 11. Application areas overview.....	173
Table 12. Node technologies overview (projects that did not focus on these aspects have been left unchecked). ....	175
Table 13. Network technologies overview (projects that did not focus on these aspects have been left unchecked). ....	181
Table 14. Middleware and overlay technologies overview (projects that did not focus on these aspects have been left unchecked). ....	188
Table 15. Architecture and formalisation efforts overview (projects that did not focus on these aspects have been left unchecked). ....	192

## LIST OF FIGURES

Figure 1. Internet of Things. Vertical & Horizontal Markets [1] .....	15
Figure 2. Number of IoT –related Articles on the IEEE Xplore Library.....	16
Figure 3. The uSPBM Framework .....	21
Figure 4. uSPBM Entities, Aggregated by Role/Service they Provide.....	22
Figure 5. Basic uSPBM Access Control Architecture.....	35
Figure 6. Data flow model of policy-based access control .....	37
Figure 7: XACML Policy components .....	38
Figure 8. uSPDM Policy example .....	42
Figure 9. The DPWS protocol stack [23] .....	44
Figure 10. From the Internet of Things (left) to the Web of Things (right) .....	45
Figure 11. Creating a simple DPWS device using Node.DPWS (left) and the same device in WS4D-JMEDS (right).....	49
Figure 12. Average response time (in ms) for 500 requests on both the simple (left) and the policy-based access control implementation (right) .....	51
Figure 13. Energy (in mJ) required to handle 500 requests.....	52
Figure 14. The Evolution of Access Control Models [105].....	54
Figure 15. Beagleboard xM.....	55
Figure 16. Smartphone and Tablet Devices .....	56
Figure 17. Beaglebone Embedded Platform.....	57
Figure 18. SunSPOT Sensor .....	57
Figure 19. IRIS Mote .....	58
Figure 20. PEP-PDP Implementation .....	61
Figure 21. PDP-PIP/PAP Implementation .....	62
Figure 22. Secure SAREvent.....	67
Figure 23. Implementation of AES/CCM-protected PEP-PDP Communication .....	68
Figure 24. Implementation of AES/CCM-protected PDP-PIP/PAP Communication .....	69
Figure 25. STRIDE Analysis of uSPBM's Core Access Control Interactions.....	70
Figure 26. OSGi - DPWS Interfacing .....	71
Figure 27. An uSPBM Device Deployed over OSGi and Discovered on Local Network.....	72
Figure 28. Implementation of the uSPBM Device Operator.....	73
Figure 29. Simple Scenario Demonstrating Device Operator's Functionality.....	74
Figure 30. Proof-of-Concept Testbed Setup .....	75
Figure 31. Client-Side Response Time (ms) .....	76
Figure 32. Response Time (ms) Breakdown, Averaged Over 100 Requests .....	77
Figure 33. PDP Processing Time. Average Response Time (ms) Depending on Number of Stored Policies. ....	78
Figure 34. PDP Processing Time vs. Number of Stored Policies (in ms). Comparison to Related Work. ....	78
Figure 35. The Effect of the Different Security Mechanisms Under Two Use Cases (times in ms, averaged over 50 requests).....	79
Figure 36. Functional Model of Policy-based Access Control for Healthcare Environments .....	83
Figure 37. uSPBM BSN implementation using DPWS .....	85
Figure 38. The BSN Bridge .....	86
Figure 39. The BSN DPWS Provider .....	87
Figure 40. Discovering the Sensors and Their Hosted Services on the Network.....	88
Figure 41. Proposed deployment of the proof-of-concept uSPBM BSN application.....	89
Figure 42. Average response time (in ms) for 50 requests. Columns in blue depict the scenario where there is no security between the sensor and the Provider, while columns in green correspond to the scenarios where AES-CBC encryption was used to protect said link.....	90

Figure 43. Authenticated Access Control. ....	95
Figure 44. DPWS-based implementation of the authentication scheme. ....	97
Figure 45. The test-bed setup, featuring embedded devices and desktop PCs. Orange lines indicate communication where WS-Security is optionally enabled. Also depicts the extra PDP & PIP/PAP introduced in the second test scenario. ....	98
Figure 46. Client-side response time for 100 requests to the Service Orchestrator. ....	99
Figure 47. Service Orchestrator's average CPU load (%). ....	100
Figure 48. Service Orchestrator's and Target device's memory utilization (in bytes) for Scenario 1....	101
Figure 49. Target device's CPU load (%) for Scenario 1. ....	101
Figure 50. Smart Home Access Control Architecture & Cross-domain Communication. Main Entities. ....	106
Figure 51. DPWS implementation of the XACML mechanisms ....	107
Figure 52. XSACd Cross-Domain Proxy Implementation and Main Steps. Simplified View. ....	110
Figure 53. XSACd Cross-domain Proxy. Command Line Remote Connection to the Embedded Test Platform.....	111
Figure 54. Screen Capture of the (PEP-protected) DPWS Test Device Deployed on the Touch-enabled Smart Platform. ....	113
Figure 55. The "Smart Home Browser"; an Application Developed to Facilitate the Discovery of DPWS Devices and Provide Access to Their Hosted Services. ....	114
Figure 56. Client-side Response Time for 100 Concurrent Requests (in ms).....	114
Figure 57. Average Client-side Response Time (in ms) for the Investigated Deployments and Usage Scenario. ....	115
Figure 58. Memory Footprint (in KB, Logarithmic Scale) on the PEP-protected Device, including the overhead compared to the simple DPWS device with no access control protection. ....	116
Figure 59. Network Throughput on Target Device During Tests. ....	116
Figure 60. RtVMF architecture and demonstrator deployment.....	120
Figure 61. Test-bed used for demonstration & performance evaluation. Proof of concept RtVMF retrofitting (insert) ....	124
Figure 62. Response time (in ms) per request, for both test platforms .....	127

# 1. INTRODUCTION

Advances in computing and communication technologies have enabled a new reality where interconnected computing systems, in various forms, are constantly gaining popularity, permeating our environments and aiming to enhance all aspects of our everyday lives. The IP-based connectivity of devices, systems and services, which goes beyond the traditional human-to-machine (H2M) and machine to machine (M2M) interactions, is nowadays labeled the Internet of Things (IoT). Ubiquitous computing devices, featuring sensors and actuators, are already deployed in a variety of domains (residential/home automation, industrial systems, military, e-textiles, healthcare and automobiles, among others; see Figure 1).

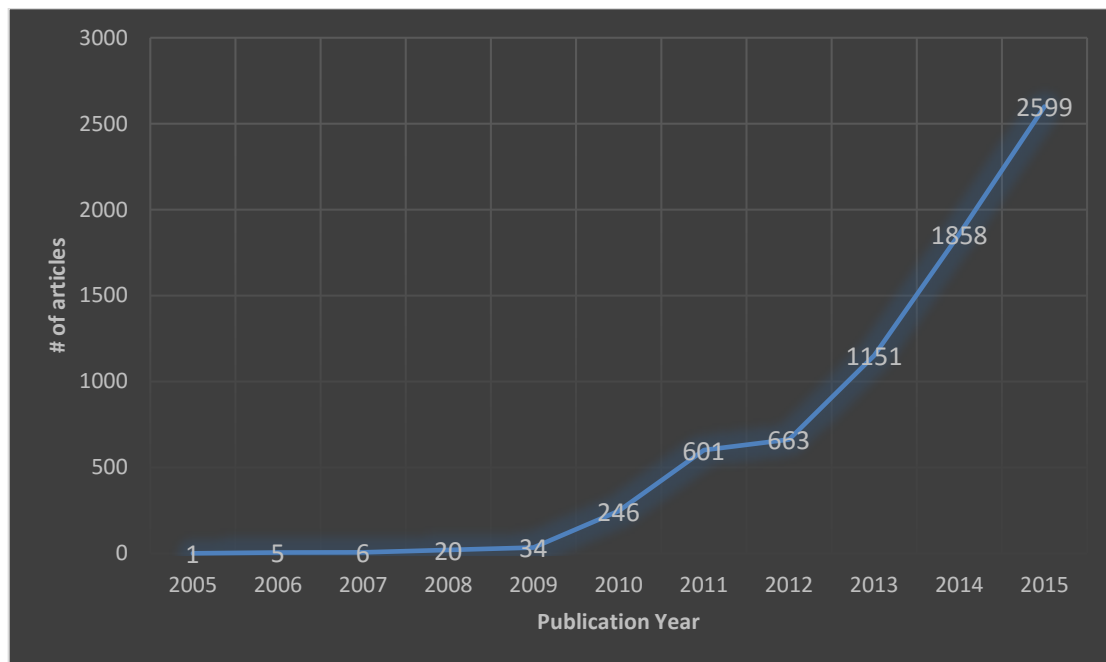


FIGURE 1. INTERNET OF THINGS. VERTICAL & HORIZONTAL MARKETS [1]

These devices come in many different forms, small or large, visible or invisible, attached or embedded, simple or complex, wired or wireless, in coordinated or ad hoc networks etc. Some examples of devices already available in the market include smart thermostats, smart fridges, doors, locks, switches and power outlets, motion & environmental sensors, smart vehicles, wireless defibrillators & insulin pumps, internet-connected MRI scanners and critical infrastructures such as power grids and nuclear centrifuges.

It is estimated that up to 200 billion devices will be connected to the IoT by 2020 (i.e. 26 connected objects per person [2]), while 5.5 million new things will be connected every day in 2016 alone<sup>1</sup>. This plethora of devices is expected to cause a surge in IP traffic, reaching 1.6 zettabytes in 2018, 57% of which will originate from devices other than personal computers [3]. These significant changes did not leave the industrial and enterprise environments unaffected, with ubiquitous computing acting as an enabler for new business opportunities and services, but also providing more sophisticated tools for monitoring and managing the existing business functions and infrastructure. Thus, the market potential is equivalently promising, with estimates of a value (net profit) of \$14.4 trillion being available to enterprises globally by IoT applications and services [3].

The research community spearheads the IoT developments in various fronts, from the hardware and sensing elements to the communication interfaces and protocols and the value added (e.g. knowledge extraction) from the data generated by the devices. The research interest in the IoT is evident from Figure 2, which presents the search results in the IEEE Xplore online library, when searching with the term “Internet of Things”.



**FIGURE 2. NUMBER OF IOT –RELATED ARTICLES ON THE IEEE XPLORE LIBRARY.**

## 1.1 MOTIVATION

While existing networking and security mechanisms are updated and adapted to handle the vast population of IoT devices, higher level, seamless M2M and H2M interactions, are a requirement in order to effectively monitor and manage the infrastructure, allowing the use of its full potential. However, at its current state, the ubiquitous computing landscape is segregated, consisting of numerous proprietary solutions, which are typically incompatible

<sup>1</sup> Gartner, 2015, <https://www.gartner.com/newsroom/id/3165317>



with each other. This makes setting up, managing and, by extension, securing a smart device ecosystem, significantly challenging.

Moreover, end-users typically do not possess the skills to configure and setup the devices that may be found in smart environments; in large-scale deployments, individually setting up devices is not even feasible. From the perspective of implementers, there is a need for rapid development and deployment, while simultaneously tackling issues of scaling and inherent limitations in terms of resources (CPU, memory, power etc.).

Furthermore, managing a large number of heterogeneous nodes in a network of embedded systems is a challenging task, mainly due to differences in requirements and resources. Nano nodes with very limited capabilities, such as the nodes of a Wireless Sensor Network (WSN), may not be suitable for adopting solutions designed for power nodes that have no such constraints. Using these devices in dynamic, ad-hoc infrastructures that feature a plethora of characteristics, has brought up the need for appropriate management of participating nodes to satisfy the corresponding policy restrictions.

This heterogeneity in large-scale deployments combined with the dynamic nature of these systems requires utilization of specialized techniques in order to manage their resources and corresponding services.

The above developments also introduce significant challenges in terms of the security and privacy of the data processed, stored and communicated by these systems [4]. The IoT applications typically handle private sensitive data (such as health readings or the users' exact location) and include direct interaction with the physical world (cyber-physical systems).

Some key ES security research challenges, such as physical layer issues, access control/authorization, authentication & denial of service attacks, lightweight cryptographic and key exchange mechanisms, secure (e.g. reputation-based) routing, communications security, secure service discovery and anonymity and location privacy issues, are identified and analyzed in Annex A – Embedded Systems Security. Moreover, a survey of EU-funded research efforts in addressing these issues can be found in Annex B – Pertinent EU-funded Research.

These overviews of security challenges and research efforts highlight the importance of ESs security is important. There can be dire consequences from a successful security attack in the case of critical systems but also these attacks are bound to become more common as ESs become an even more integral part of our lives, with the widespread adoption of smart devices in our homes, cars, clothes etc. The particular characteristics of such resource-constrained devices and the varied requirements of their applications not only introduce new vulnerabilities but also intensify existing ones. Moreover, mechanisms and techniques (e.g. for access control, cryptography, network routing etc.) that would typically be deployed to secure other types of computing devices are not always applicable or have limited efficacy in the context of embedded systems. Thus, research should establish secure mechanisms tailored for ES, consider ES security a non-functional requirement and avoid building insecure solutions that “just work” and then try to correct flaws with patches. Security performance is going to be one of the next product differentiators in embedded products and services.

It may be irritating and costly to have a teenage neighbor control our smart thermostat, but a total blackout caused by a hack of the centralized power monitoring and distribution infrastructure can have significant consequences in terms of asset damage or panic to the public. In extreme cases, where a malicious entity might control a self-driving vehicle, an implantable smart medical device or even a critical infrastructure (such as a nuclear centrifuge), the security incident may directly lead to injuries or even deaths of one or more individuals.

According to a Cisco survey [5] on 7.501 business and IT decision makers, the biggest downside to the increased pervasiveness of computing devices are the associated new threats to data & physical security. A survey [6] from Accenture on 28.000 respondents from 28 countries, security is now a top barrier to IoT expansion, as early adopters are choosing to abandon IoT applications because of security concerns.

These concerns are not baseless, as there are already numerous successful attacks in a variety of smart domains. A critical vulnerability in smart heating and power systems of a German company, Vaillant, which allowed attackers to gain unauthorized access and turn them off or damage them at will<sup>2,3</sup>. A similar case [7] involved smart meters deployed in millions of households in Spain that allow attackers to take full control of the device, remotely shut down power and tamper with consumption readings, among others. Focusing on modern vehicles, Koscher et al. [8] have demonstrated that it is feasible to manipulate all critical sub-systems in modern automobiles, disabling or activating brakes, stopping the engine, injecting malicious code and completely ignoring driver input. More recently, a security bug enabled a remote attack of Fiat/Chrysler's Uconnect system<sup>4</sup>, letting hackers apply the brakes, kill the engine and take control of steering over the internet, forcing the company to issue an update<sup>5</sup> to the affected vehicles' software. Medical devices are also affected. Security researchers [9] have successfully hijacked a tele-operated surgical robot during surgery and made it impossible for a surgeon to remotely operate. A security researcher remotely disabled his own insulin pump live on stage during the 2011 BlackHat Conference [10]. More recently, researchers [11] identified 68,000 vulnerable medical systems, accessible online, belonging to an unnamed U.S. health organization. The equipment included 488 cardiology machines, 323 picture archiving and communication gear, 133 infusion systems, and 97 MRI scanners. The latter security analysis was a results of a 3-year research project on the security of medical devices, which also revealed [12] drug infusion pumps (for delivering morphine drips, chemotherapy and antibiotics) that can be remotely manipulated to change the dosage doled out to patients. Moreover, Bluetooth-enabled defibrillators were identified that can be manipulated to deliver random shocks to a patient's heart or prevent a medically needed shock from occurring. Successful attacks also included accessing x-rays from the hospital's network, resetting temperature settings on refrigerators storing blood and drugs (thus causing causing spoilage)

---

<sup>2</sup> <http://www.bhkw-infothek.de/nachrichten/18555/2013-04-15-kritische-sicherheitsluecke-ermoglicht-fremdzugriff-auf-systemregler-des-vaillant-ecopower-1-0/>

<sup>3</sup> <http://www.hotforsecurity.com/blog/vulnerability-in-vaillant-heating-systems-allows-unauthorized-access-5926.html>

<sup>4</sup> <http://www.wired.com/2015/07/hackers-remotely-kill-jEEP-highway/>

<sup>5</sup> <http://www.theguardian.com/technology/2015/jul/21/jeep-owners-urged-update-car-software-hackers-remote-control>

and digital medical records that could be altered (to cause physicians to misdiagnose, prescribe the wrong drugs or administer unwarranted care). One of the main problems identified by the researchers lay with embedded web services that allow devices to communicate with one another and feed digital data directly to patient medical records, as many of the web services allow unauthenticated or unencrypted communication between the devices.

Studies [13] and published reports [14] reveal that current smart device deployments have not adequately considered the threats that these nodes face when connected to the Internet, hence the lack of the security measures.

The Open Web Application Security Project (OWASP) organization includes “Insufficient Authentication/Authorization” in the second place of its list of top ten security problems identified on IoT devices [15], preceded only by the use of “Insecure Web Interfaces”. A recent Hewlett-Packard study [16] on smart home security appliances revealed that all devices contained significant authentication & authorization vulnerabilities. A risk analysis [17] applied on a smart home automation system (as part of a research project involving leading industrial actors), revealed that the highest ranked risk (i.e. the most probable and the one with the highest potential impact) affecting smart homes is derived from inadequate or absent access control configuration and policies.

Such negligence in terms of proper authentication and authorization are bound to inhibit any efforts made towards using these pervasive devices to handle our personal sensitive data. The expanded attack surface that results from the integration of the numerous smart devices around us with the Internet needs new or adapted mechanisms to mitigate these new threats.

Still, the intricacies of IoT applications should dictate the features of the adopted access control mechanisms. A survey [18] on smart home users revealed people need fine-grained, context-aware, dynamic access control. The required features included fine-grained division of people & resources, type of access (e.g. read/write), adapting to presence, location (local/remote), time of day etc., as well as reactive policy creation. The need for adaptability and context-awareness is commonly cited among users and other stakeholders for the industry and the research community.

The European Network and Information Security Agency (ENISA) recently concluded [19] that security services, such as authentication and access control, have to be non-intrusive and to be able adapt to the constantly changing contexts of smart spaces. This touches another important barrier to IoT expansion, and one that is intertwined with the inadequate security: the lack of interoperability. Indeed, a Microsoft survey [20] on smart home users highlighted inflexibility (vendor “lock-in”), poor manageability and difficulties in achieving security as significant barriers to the broader adoption of pertinent technologies and devices.

These concerns are not restricted to consumer audiences; an ENISA report [21] on security and resilience of e-health infrastructures and services (involving cyber security experts, academics and operators within the field of cyber security) identifies that access control is a very significant priority in securing e-health applications. Moreover, it was noted that a key

requirement in achieving effective and secure e-health services is having a high level of interoperability.

## 1.2 THE FRAMEWORK

Motivated by the above, uSPBM combines novel and standardized technologies to provide a lightweight and usable framework for policy-based management of ubiquitous smart devices.

The eXtensible Access control Markup Language (XACML [22]) is used to convey policy requirements in a unified and unambiguous manner. Thus, XACML defines the structure and content of access requests and responses exchanged among the framework's access control entities. In terms of mechanism(s) used to transfer these messages, Service Oriented Architectures (SOAs) provide an attractive option that can be chosen to convey policy information. Embedded devices and SOAs are becoming convergent technologies with several standards emerging from these efforts. This approach has already been successful in business environments, as web services allow stakeholders to focus on the services themselves, rather than the underlying hardware and network technologies. When deploying a SOA there are quite a few effective options to provide these services, but the Devices Profile for Web Services (DPWS [23]) specification stands out. It enables the adoption of a SOA approach on embedded and sensor devices with limited resources, allowing system owners to leverage the SOA benefits across heterogeneous systems that may be found in smart environments. In this work, the typical policy based access control architecture is mapped to a SOA network of nodes, enabled by the compact web service implementation provided by DPWS, to provide protected access to their distributed resources.

The layout of uSPBM architecture, along with its main entities, can be seen in Figure 3.

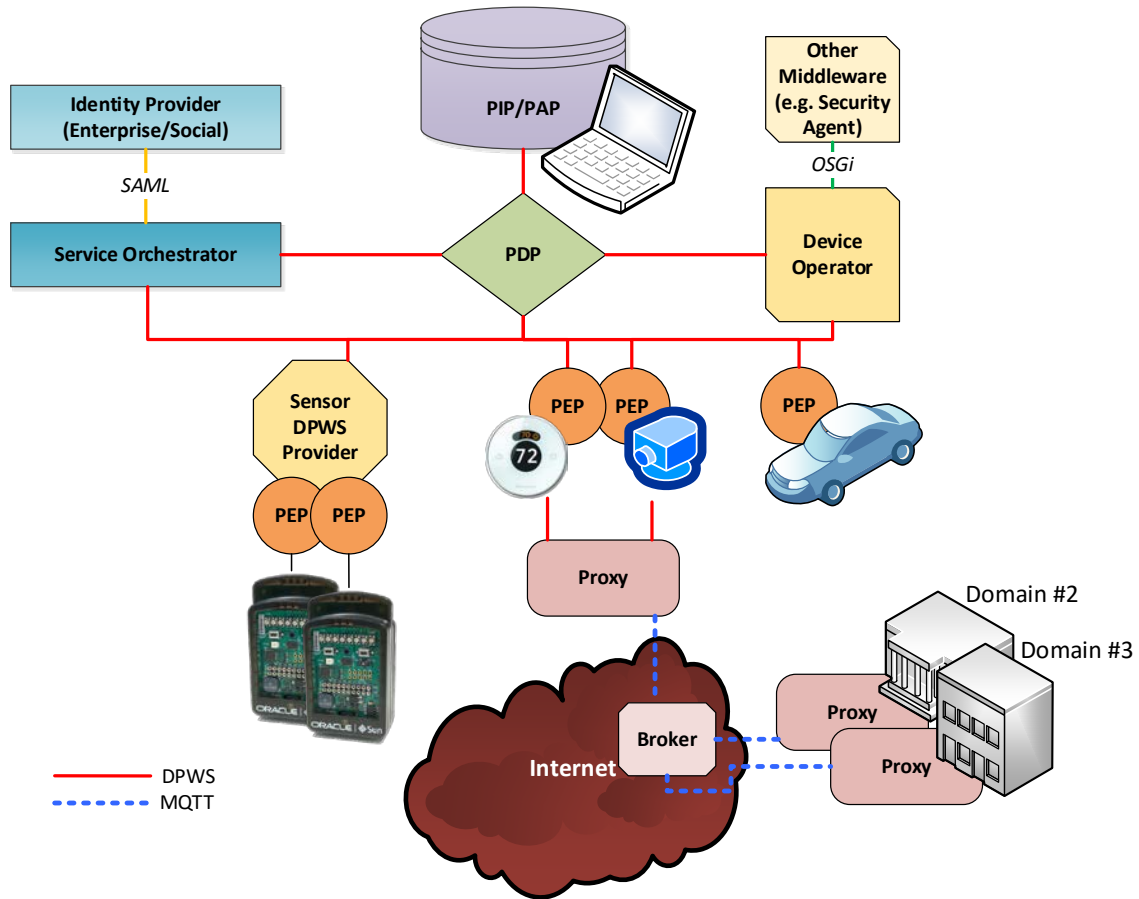


FIGURE 3. THE USPBM FRAMEWORK

At the core of the framework lay the main authorization-related entities, namely the Policy Decision Point (PDP) and the Policy Information Point (PIP) and Policy Administration Point (PAP), along with the various Policy Enforcement Points (PEPs) residing on each of the managed smart devices under uSPBM's control. The basic authorization functionality is augmented by other entities, responsible for further tasks which are equally important in having a usable framework that can be applied in real use cases across a variety of IoT domains. More specifically, the Service Orchestrator is interfaced with one or more identity providers to offer authentication services. The Device Operator acts as an interface between uSPBM and other middleware that may run at the backend to monitor, coordinate or even automate the management of the smart devices. When necessary, a DPWS Provider is deployed to allow for the integration of extremely resource constrained sensor devices under the framework's supervision. Finally, the Broker is responsible for coordinating the MQTT-based [24] communications among the uSPBM proxies residing in all domains that uSPBM manages. This association of entities to their specific services/tasks is better depicted in Figure 4.



Smart meters are typically used to record consumption of power and/or water and/or natural gas for monitoring and billing purposes. Such devices involve two-way communication between the device itself and vested party (or parties, e.g. utility companies, service providers, suppliers etc.) and feature real-time or near-real-time sensors.

The inherent security issues of the transmissions typically used in such deployments, the accompanying privacy issues and certain sensitive functionality present in such devices (e.g. remote “kill switch”), necessitate the deployment of strong security mechanisms.

The presented scheme can be used to allow for a fine-grained policy-defined access control to all the functional (e.g. “kill-switch”) and non-functional (e.g. monitoring) elements of the devices and among all interested parties/stakeholders. Moreover, the underlying technologies (i.e. XACML and DPWS) are fitting for the large-scale deployments of this application area.

### 1.3.2 SMART VEHICLES

The intelligence being built into various vehicle types, will not only improve their safety and comfort but also enable new modes of transportation and new types of services, creating the corresponding markets. Nevertheless, the existing attack surfaces are expanded by said intelligence; researchers have demonstrated that it is feasible to manipulate all critical sub-systems in modern automobiles by using a wireless-enabled MP3 player connected to the vehicle’s embedded control network [8]. The presented attacks include accessing the brake controller, thus disabling or forcibly activating the brakes and consequently compromising the safety of the driver and passengers, as well as injecting malicious code to erase any evidence of tampering after a crash.

In all cases, it will be important to be able to monitor, preferably in real-time, various parameters of the smart vehicles’ condition. 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 uSPBM 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 pro-per 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. Moreover, by constantly monitoring various operational parameters, uSPBM can enable real-time monitoring and interaction with a smart vehicle or a smart vehicle fleet.

### 1.3.3 E-HEALTH

Healthcare stands out as a key sector where these novel technologies and associated enhanced services can have a significant impact by improving the quality of life of patients, elderly people, but also the general population through real-time monitoring and intervention which enables proactive and more effective health management, justifying the intensive research efforts in the field [25]–[27].

The sophisticated sensor nodes can be deployed as standalone devices serving a single purpose, or as part of an infrastructure that consists of nodes with similar characteristics

comprising a so called low power and lossy network (LLN). Moreover, they can be used simply for monitoring various variables or for acting upon command issuance, be part of a closed system or provide advanced services to remote parties over public networks. The current trend for all these nodes is to adopt existing networking technologies and be reachable over the Internet, abandoning proprietary closed solutions.

In the context of healthcare applications, the security issues are exacerbated by the direct interaction with the human body and the associated safety and privacy concerns. In typical nodes used for eHealth purposes, environmental and physiological sensors are deployed for gathering all the required information depending on medical staff's prescribed needs, such as blood pressure and body and room temperature. On top of that, actuators controlled by authorized medical staff can also be deployed, such as an automatic insulin injection device used for remote treatment. Such sensitive actions, i.e. reading and issuing commands, need strict access control decisions before being authorized so that user's privacy and even safety are not jeopardized by unauthorized actions. Thus, this is another area where uSPBM 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 uSPBM provides the necessary fine-grained access control to resources.

#### 1.3.4 TRANSPORT INFRASTRUCTURE

In a smart transit infrastructure, such as a railway facility, the deployment of uSPBM framework would enable fine-grained, policy-based control of all smart devices that may be present in a railway deployment (e.g. DPWS-enabled cameras or sensors monitoring access to carriages or control stations) from remote locations. The infrastructure could be monitored via any compatible application 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.

#### 1.3.5 SMART HOME

Manufacturers would like to have secure access to their deployed appliance controller (node) to get necessary readings for appliance's maintenance or service. The manufacturer wants to have this communication protected to avoid any unnecessary disclosure of information to nearby devices and their manufacturers. Moreover, from an end-user perspective, it would be desirable to be able to control access to the various functional and non-functional elements of smart appliances. So, for example, all members of a family may be able to get room temperature readings using their smart devices (phones, tablets, watches etc.), but only the parents will be authorized to set desired temperatures, denying access to unauthorized entities (kids, visitors or entities intruding the smart home network).

The uSPBM scheme proposed here can be deployed to facilitate said functionality, via the appropriate policy definitions. While typical access control deployments require the setup of complex infrastructures to enable entities' interaction and policy retrieval (e.g. via the



Lightweight Directory Access Protocol, LDAP [28]); such an approach may be acceptable for corporate environments but is not suitable in the context of consumer applications and the average home user. To this end, the proposed framework leverages the benefits of DPWS, which allows the deployment of devices aligned with the Web Services technologies, thus facilitating interoperability among services provided by resource-constrained devices. The adoption of DPWS facilitates seamless Machine-to-Machine (M2M) discovery and interactions, allowing the deployment of the framework's entities to any platform, anywhere on the home network, with minimal involvement on behalf of the user.

### 1.3.6 AMBIENT ASSISTED LIVING

Ambient assisted living is one of the market niches with a highest growth in developed countries where revenues come from 195 million Euros in 2009 to a prevision of 525 million in 2015<sup>6</sup>. On the other hand, it's important to take into account that more than 25% of the European population will be over 65 years by 2035, giving this area a great growth potential [29]. These figures make private home-care companies and the public administration need to look for solutions to provide their users a good service level but without neglecting safety because personal, biomedical, location or presence data are used.

AAL systems include safety functions, like automatic shutdown of a stove the case of long absence of a user or informing emergency security or medical services in the case of burglary or a medical incident respectively. Other features are designed to enhance the everyday lives of individuals, e.g. room temperature or lightning controls that adapt to the needs and everyday routines of the inhabitants. Moreover, various eHealth-related devices may be present in an AAL environment.

Tenants can rely on these secure smart platforms for their security. A window left open during night or when not at home can trigger a series of actions that cannot be bypassed by an intruder due to the robust security features of uSPBM. A tenant can remotely control his/her home living conditions through smart appliances (air conditioning, alarm, water warming device), through customized applications in order to provide comfort without endangering user's privacy. Smart devices will be able to sense user's presence at home (e.g. through mobile phone presence) and act accordingly. The policy of all these can be set by the tenant or by the local authorities in case of emergency, such as environmental hazards. The desired security level and corresponding functionality can be adapted dynamically or during deployment to meet the tenant's specific requirements or mitigate unforeseen risks during operation phase. Such smart platforms can also enhance the tenants' safety by alerting the individuals about potentially dangerous situations, such as a heater, a kettle or a cooker that has been left on. What is more, in case of an accident, an appropriate operator could be automatically alerted by the system who in turn could authorize the immediate deployment of a team able to handle this particular emergency, so as to check upon the given individual and act accordingly. In an AAL deployment, uSPBM agents can monitor and manage the ambient environment of a home. In cases of emergency, the agent can inform the home-agents of the inhabitant's relatives and neighbors based on policies and user preferences. Moreover, the deployment of uSPBM, which is by design deployable on heterogeneous

---

<sup>6</sup> <http://www.slideshare.net/FrostandSullivan/assisted-living-in-europe-technology-and-market-trends-2010>

systems, will also facilitate the Integration of AAL technologies with smart home and smart grid infrastructures which are expected to be used concurrently.

#### 1.3.7 SMART FACTORY (INDUSTRY 4.0)

The computerization of traditional industries (e.g. manufacturing) will lead to the introduction of Smart Factory environments and, consequently, the fourth industrial revolution (i.e. “Industry 4.0”). Such examples include machines that predict failures and trigger maintenance processes autonomously or self-organized logistics that are able to react to unexpected changes in the production chain.

As with the previous cited scenarios, uSPBM can be deployed to guarantee that the pre-defined access control policies are followed enforced on all smart devices present on the production floor or elsewhere. The above can be significant enablers, as machine and, most importantly, worker safety are key areas of concern in such deployment scenarios.

This thesis is organized as follows: Background material and related research efforts are presented in section 2. Section 3 presents the core technologies used in uSPBM, focusing on authorization and communication mechanisms, the benefits of their combination and the advantages of the purpose-developed libraries. Details on the framework’s implementation are presented in Section 4, also focusing on the important aspects of the protection of the framework’s messaging. The rest of uSPBM’s functionality is presented in more detail in Section 5, in the context of different use cases. The work concludes in Section 6, where the highlights of the framework are presented, along with directions to future research that could further enhance the presented approach.

## 2. BACKGROUND & RELATED WORK

In recent years, we have experienced a lot of innovation in the Internet of Things (IoT) space. Collections of embedded and wearable nodes, typically bearing sensors and actuators, are becoming part of a networking infrastructure and gain connectivity to the Internet. The corresponding technologies are becoming mature enough to allow us to start looking into more advanced and comprehensive solutions that can enable these nodes to integrate smoothly with existing infrastructures, expanding, however, existing attack surfaces. As computing becomes ubiquitous, researchers and engineers aim to update and adapt existing technologies to efficiently handle the vast population of resource-constrained devices expected to co-exist in the IoT.

On the communication level, all efforts are towards the integration of low power and lossy networks (LLNs) with existing networking technologies to provide internet connectivity and realize the IoT. Several solutions have emerged through this process with their particular advantages, disadvantages and properties. Most of them have provided their own standards and specifications and have helped formulate antagonistic technologies. They might differ on various layers of the TCP/IP stack, such as the physical and the network layer or on the upper layers, i.e. presentation and application layers. Regarding the former we have technologies like open standards 6LoWPAN [30] and ZigBee [31] (which is free for non-commercial purposes), proprietary provided under a license like Z-Wave [32], and alternatives like Bluetooth [33] and Wi-Fi [34] usually met in other environments. It is beyond the scope of this work to name all these technologies and provide a comparative analysis. The proposed solution focuses on the architecture level and on the upper layers of the TCP/IP stack, thus making this solution underlying protocol independent. 6LoWPAN seems to outweigh other technologies given its Internet connectivity orientation which provides many benefits to adopting solutions.

On upper layers of the TCP/IP stack, protocols provide methods to exchange structured or unstructured messages that facilitate (secure) service access. Data are typically encapsulated in standardized protocols that allow the seamless exchange of messages between nodes and remote entities, outside the LLN boundaries, even if this is accomplished through the use of a bridge and/or router. Among the technologies being used are the service-oriented ones with several schemes being used for the way these services are provided and how a service consumer can access them. Standardization and research efforts in the area of Service Oriented Architectures (SOAs) have been taking place for more than a decade and schemes have been proposed and standardized regarding service discovery, registration, access and protection, and the corresponding communication protocols that enable the interoperable exchange of messages among remote participating entities. While in some cases efforts focus on adapting existing technologies to the constrained environment provided by such devices, other initiatives target for the introduction of new mechanisms specifically designed for such environments, without however neglecting interoperability with existing Internet technologies.

SOAs evolved from the need to have interoperable, cross-platform, cross-domain and network-agnostic access to devices and their services. This approach has already been

successful in business environments, as web services allow stakeholders to focus on the services themselves, rather than the underlying hardware and network technologies.

The deployment and orchestration of web services on heterogeneous embedded devices is a very active research area, following the success of some early research efforts. These include the "Service Infrastructure for Real time Embedded Networked Applications" (SIRENA, ITEA2 [35]) project which proved the feasibility and advantages of integrating web services, via DPWS, across business segments, including critical sections (e.g. the production floor [36]). Its follow-up projects, "Service-Oriented Device & Delivery Architecture" (SODA, ITEA2 [37]) and "Service-Oriented Cross-layer Infrastructure for Distributed Smart Embedded Devices" (SOCRADES, FP6 [38]), built upon the work of SIRENA and focused on providing a secure scalable ecosystem and adding more sophisticated features into the SOA-enabled devices to serve the requirements of future manufacturing [39], respectively. Some pervasive applications often require remote management and monitoring while maintaining interoperability, and the Web Services standard offers a solid basis for that. It is therefore justifiable that the runtime of the middleware developed for the MORE project [40] was based on the aforementioned DPWS specifications, as detailed in [41]. More recent research efforts include the "Architecture for Service-Oriented Process" (IMC-AESOP, FP7 [42]) and "Web of Objects" (WoO, ITEA2 [43]) research projects. Researchers in AESOP proposed a system-of-systems approach for monitoring and control, based on a SOA for very large scale (i.e. up to tens of thousands of devices) distributed systems in process control applications. WoO promoted the use of DPWS to build a secure, context-aware network and services infrastructure for smart objects, focusing on the interoperability of devices & services through the use of semantics.

As mentioned, the DPWS specification defines a minimal set of implementation constraints to enable secure Web Service messaging on resource-constrained devices. It employs similar messaging mechanisms as the Web Services Architecture (WSA), with restrictions on complexity and message size, allowing the provision of totally platform- and language-neutral services, similar to those offered by traditional web services, allowing system owners to leverage the SOA benefits across heterogeneous systems that may be found in the various smart environments (residential, enterprise etc.).

The SOA-based approach of DPWS acts as an enabler, with added potential stemming from the enhanced real-time monitoring and control features usable over the whole smart infrastructure. By facilitating the migration of low level data (e.g. sensing data) into higher level contexts (e.g. business operations or knowledge extraction from aggregated data), new types of services are made possible. These new types of enhanced features and services are expected to be vital for future end-users as well as enterprise deployments, in a number of industry domains [44], [45].

In this context, the work of Leong et al [46] presents a rule-based framework for heterogeneous smart-home systems management. Their work focuses on the use of SOAP for interoperability and uses an Event-Condition-Action (ECA) mechanism for machine-2-machine interactions and orchestration of the devices. The SOAP-based interoperability framework has been further extended by Perumal et al [47] with the addition of a service stub to facilitate

the addition of new devices and a database module to handle the queries of the SOAP messages (including home service functions, operation logic and access to other local or remote databases).

SOA-DOS [48] is a SOA-based distributed operating system proposed in the relevant literature, aiming to manage all embedded devices in a home network and facilitating interoperability between the various systems. The work manages to provide a SOA-based solution that is also applicable to very resource-constrained platforms (like sensor nodes), but deviates from standardized mechanisms, e.g. resorting to the use of the JSON [49] format instead of XML for data exchange.

The use of SOA concepts to tackle the dynamic and heterogeneous nature of home-control applications has also been proposed by Bourcier et al [50]. The authors introduce an implementation of their approach based on open source, standardized platforms, providing bridges to seamlessly integrate disparate devices (including DPWS devices) and their services into their home control infrastructure.

The DPWS stack also forms the basis of iVision [51], a purpose-built hardware platform used to add context-awareness to a service architecture for controlling home appliances, and its accompanying architecture. In the above work, the context information extracted by the iVision camera and all the necessary smart home appliance communications are exposed as web services using DPWS.

As many industry leaders from a variety of sectors (electronics, power, automation, enterprise, home etc.) have been involved in the above research efforts, it is evident that, in addition to the researchers' interest, there is also significant industry backing of DPWS and its use in future applications and products. Moreover, the use and benefits of DPWS have been studied extensively in the context of various applications areas, which, other than the ones already mentioned, include automotive and railway systems [52], industrial automation [53], eHealth [54], [55], smart cities [56], smart homes [57] and wireless sensor networks [58]. All of the above are positive indicators for the future of the technology chosen as the underlying implementation and communication mechanism for the presented framework, and its potential for ubiquitous adoption

Many access control schemes have been proposed for resource constrained devices, such as wireless sensor networks, yet most of them focus on authentication and authorization schemes and on enhancing basic access control models, such as providing additional features to address privacy matters. Such schemes can be found in [59]–[65]. Some of the proposed mechanisms are based on the use of public-key cryptography, a choice that is very expensive for nano nodes found in a BSN. The EU-project “Internet-of-Things Architecture” (IoT-A) worked on the adoption of XACML in the Internet of Things [66]. 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 than the use cases examined in this thesis.

Policy-based management has successfully been implemented in various types of sensor networks, even though none of the identified efforts focus on the use of standardized, platform-agnostic technologies. One such instance is presented in [67] and involves the use of policy-based management in body-area networks (BAN), where autonomous adaptation to changing conditions (failures, user activity, patients' clinical condition) is a requirement. The toolkit that was developed and deployed, *Ponder2*, allows the specification of rules in the form of event-condition-action, which enforce a given policy. Additional functionality of this toolkit includes the logical grouping of components in domains, as well as the dynamic loading of new functionality and communication protocols.

Another application of policy-based management was the SNOWMAN framework presented in [68] that allowed nodes of a WSN to autonomously organise themselves. A lightweight policy distribution protocol was developed, TinyCOPS-PR, as well as a policy information base (PIB). For facilitating scalable and localised management of WSNs, nodes are organised into three logical groups: regions, clusters and sensor nodes. The simulation results revealed that the scheme features lower power consumption compared to other schemes. The work in [69] proposes the use of an extra middleware layer that introduces a level of abstraction, thus making it easier to describe and enforce both functional and non-functional business requirements of different end-users.

An architecture for policy-based WSN management was proposed in [70]. It distributes the management functionality across the sensor network and employs clustering for improved management. The scheme also includes functionality for hierarchy maintenance management and cluster maintenance.

In [71] a framework for implementing policy-based management in WSN is proposed that makes use of the *Finger2* policy system which, although derived from the *Ponder2* policy system, it is considerably simplified so as to run on motes. Furthermore, examples of policies are given that deal with the self-healing aspects of sensor networks. Policy-based reconfiguration is thus able to deal with various network faults.

In [72] the authors also utilize XACML but focus on the privacy of eHealth data within the mobile environment. In contrast to the work presented here, a complete framework is not included and the authors choose computationally intensive security mechanisms such as XML encryption digital signatures. In [73] the authors propose a lightweight policy system for body sensors but they do so by presenting a custom API and policy definitions, thus sacrificing interoperability with existing standards and infrastructures.

Santos-Pereira et al [74] focus on enforceable security policies for systems interoperability and data exchange between healthcare entities. The authors present a Role-based Access Control mobile agent model, using public key infrastructure for authentication and access control, but the proposed scheme is presented at design-level, lacking implementation details and a performance evaluation.

Researchers have studied the use of access control mechanisms to safeguard the users' privacy, a key concern in the context of smart environments. Faravelon et al [75] outline such an architecture in the context of SOA-enabled pervasive environments, using a medical

scenario as a test case. The interoperability with DPWS is considered, among other SOA technologies, but a non-standardized approach is adopted for the access control functionality.

Privacy issues have also been considered by Jung et al [76], who have presented a generic concept of access control for home automation gateways, aiming to safeguard the privacy and security of users and their data. The scheme is based on a customized SOAP message structure that integrates XACML attributes within SAML-based access token. However, the initial, theoretical evaluation of the proposed scheme indicates that this approach is quite costly (especially in terms of packet size), which questions its applicability in the context of embedded smart home devices. The authors acknowledge this drawback and indicate it will be investigated, as future work, on actual platforms.

In uSPBM the typical policy based access control architecture of XACML is mapped to a DPWS-enabled network of nodes to provide protected access to their distributed resources. A survey of the literature reveals a wealth of related work, including various diverse approaches and attesting to the applicability of XACML in the context of ubiquitous environments.

Kim et al [77] have proposed the use of an OSGi (Open Services Gateway initiative [78]) -based framework to integrate heterogeneous smart-home devices and services. The proposed framework also includes an access control model, combining the XACML mechanisms with OSGi services to appropriately create the queries that will be forwarded to the entity responsible for access control decisions (i.e. the Policy Decision Point, PDP). While the proposed approach theoretically supports a variety of protocols (including DPWS devices), the presented analysis and proof of concept implementation are mainly based on UPnP, a protocol lacking in many respects, already noted above (e.g. security & scalability). Furthermore, the performance of the proposed mechanisms – an important aspect considering the resource-constraints of many smart devices – is not evaluated.

Another approach found in the literature is the use of a "Reference Monitor" entity [79], i.e. a home gateway, whose goal is to provide and enforce a collection of access control policies, aiming to help satisfy a user's access, convenience, and privacy requirements. The access control policies are de-fined using XML, but deviate from the standard XACML syntax and architecture.

Busnel et al [80] present a case study for remote healthcare assistance in smart homes. Most of the smart home security & dependability requirements are discussed extensively, identifying the use of SOAs along with XACML as the most applicable technologies to fulfill these requirements. An XACML-based authorization solution is applied using the security pattern approach to satisfy security requirements typically existing in such environments. This work presents the outline of such a framework, but not an actual implementation of the SOA and XACML mechanisms, nor a performance evaluation. The resource-constrained nature of the target devices and the use of appropriate security mechanisms do not appear to have been considered during the design phase.

Seitz et al. [81] have presented an authorization framework for the Internet of Things. The authors use XACML to offer fine-grained access control on resources and propose the use of CoAP as a lightweight transport protocol. Still, the adaptations proposed break compatibility

with standard XACML/SAML infrastructures and there is limited implementation and performance evaluation, while, no specific authentication or device management mechanisms considered.

Fremantle et al. [82] propose the combination of OAuth for identity and access management on IoT devices using MQTT for information distribution. MQTT is lightweight but only offers synchronous communication. On the other hand, OAuth is gaining traction of authorization of resources at the Internet scale. Still, it lacks the granularity of XACML for access control. Nevertheless, the proposed authentication mechanism is compatible with uSPBM, replacing the SAML-based solution adopted in our framework (something trivial to accomplish due to the modularity of uSPBM).

Müller et al [57] have also proposed the combined use of DPWS with XACML, but focusing on end-user content (e.g. the distribution of multimedia files). They also use proxies to establish trust relationships across smart home domains but the authors did not exploit DPWS in the implementation and deployment of the XACML architecture. The proposed model is based on X.509 certificates and tailored to the needs of a “smart home” environment but can easily be adopted to other scenarios (e.g. in our proposed scenarios, power nodes could adopt the CA roles described in the above work). It is demonstrated that processing time overhead (especially in the XACML lookup phase) is affected by the number of policies maintained at the PDP. It is, therefore, imperative to fine-tune the amount of queries the PDP needs to lookup or even the number of PDPs available on the network (e.g. segregating service discovery and application PDP duties), depending on application requirements.

Table 1 aggregates recent work identified in the literature that is related to uSPBM, categorizing and comparing their features. As the survey of the related work reveals, the problem of authentication & authorization for the Internet of Things is not conclusively solved yet. Current approaches typically concentrate on a number of very specific problems and/or are not implemented fully. Thus, our approach was to design and implement a complete authentication, authorization & management framework, based on established and emerging standards.

**TABLE 1. USPBM COMPARED TO RECENT RELATED WORK**

	uSPBM	Muller et al. [57]	Kim et al. [77]	Busnel et al. [80]	Jung et al. [76]	Serbanati et al. [66]	Seitz et al. [81]	Sleman et al. [48]	Bourdenas et al. [71]	Leong et al. & Perumal et al. [46], [47], [83]	Fremantle et al. [82]
Authorization	√	√	√	√	√	√	√	√	√	√	√
Authentication	√				√	√					√
Management	√							√	√	√	



Standardized Technologies	✓	✓	✓	✓	✓	✓	✓				✓
Lightweight	✓					✓	✓	✓	✓		✓
Cross-domain communication	✓	✓	✓	✓		✓					✓
Proof-of-concept implementation	✓	✓	✓		✓		✓	✓	✓	✓	✓
Performance evaluation	✓	✓			✓		✓	✓	✓	✓	
Applied to various use cases & heterogeneous devices	✓										

### 3. CORE TECHNOLOGIES & ENTITIES

This section details the core technological building blocks and key components of the uSPBM framework, focusing mostly on the components directly related to the framework's authorization mechanisms.

As uSPBM was designed and developed following a modular approach, the mechanisms not directly related directly to the authorization process (and the associated entities) are presented separately, in later chapters, in the context of specific use cases.

#### 3.1 POLICY-BASED ACCESS CONTROL

Among the studied schemes proposed for systems with different requirements and properties, a cross-platform solution that meets the requirements of all types of embedded systems and provides interoperability, crucial for next-generation pervasive computing devices, is based on eXtensible Access control Markup Language (XACML [22]) policies. XACML is an XML-based general-purpose access control policy language used for representing authorisation and entitlement policies for managing access to resources. However, it is also an access control decision request/response language. As such, it can be used to convey policy requirements in a unified and unambiguous manner, hence interoperable and secure, if appropriately deployed.

The above fit well into the model of a network of heterogeneous embedded systems where access to resources is provided by nodes as a service, and into the management architecture developed by IETF Policy Framework. This typical policy based management architecture combined with XACML, is mapped to a Service Oriented Architecture (SOA) network of nodes to provide protected access to their distributed resources. It consists of several components that run on different nodes of the architecture. These components are [22][84]:

- **Policy Enforcement Point (PEP):** The system entity that performs access control, by making decision requests and enforcing authorisation decisions.
- **Policy Administration Point (PAP):** The system entity that creates a policy or policy set.
- **Policy Decision Point (PDP):** The system entity that evaluates applicable policy and renders an authorization decision.
- **Policy Information Point (PIP):** The system entity that acts as a source of attribute value

Moreover, auxiliary entities may also co-exist with the above, depending on the specific application and deployment at hand. Some of these entities, which will be showcased in later chapters, may include the following:

- **Context Handler (CH):** Orchestrates the communications among the stakeholders, converts, if necessary, messages between their native forms and the XACML canonical form, and collects all necessary information for the PDP.

- **Obligation Handlers (OH):** Provide additional restrictions that should be taken into account when enforcing a decision, like the requirement to log any permitted access or to inform for unauthorized attempts.
- **Environment:** Provides additional information independent of a particular subject, resource or action.

Considering that some smart platforms may not have the computing resources to accommodate expensive mechanisms, some of these roles (e.g. PDP) may only be under-taken by more powerful nodes expected to operate within the target node's (i.e. the PEP's) trusted environment. Thus, a node, depending on its capabilities and the available resources, might include one or more of these functional components. A basic deployment of uSPBM's access control entities is depicted in Figure 5.

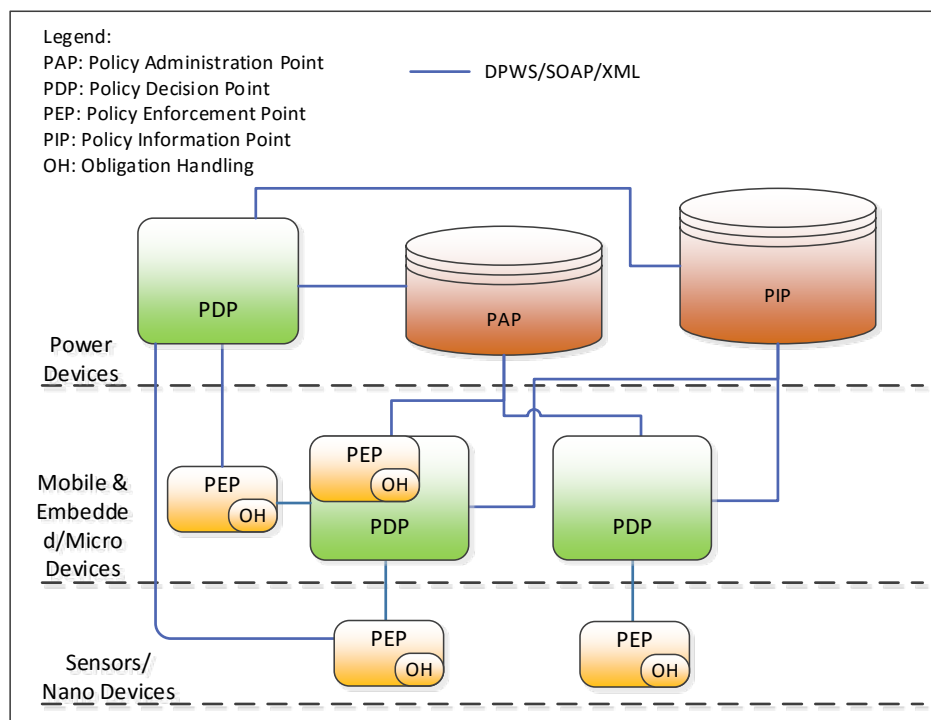


FIGURE 5. BASIC USPBM ACCESS CONTROL ARCHITECTURE

The XACML handling and decision-making engine can be adopted from any open source implementation. Such open-source resources include Sun's XACML implementation [85], PicketBox XACML [86] (formerly JBossXACML), the Holistic Enterprise-Ready Application Security Architecture Framework (Heras AF) XACML [87] and the Enterprise Java XACML project [88]. Closed-source commercial alternatives exist as well, but these are not as modifiable and, thus, often have limited usability in custom implementations. Considering the above options, Sun's XACML is the framework of choice for implementing uSPBM's access control engine, as it remains popular among developers and is actually the basis of various current open source and commercial offerings.

### 3.1.1 TYPICAL APPLICATION SCENARIO & DATA FLOW

As an example, consider the case of a person who owns a smart thermostat. The thermostat is a device that hosts a service which supports multiple operations such as setting the target temperature, selecting operation mode, enabling power save, getting the current status or even events such as notifications when the temperature in the room changes, or the target temperature has been reached. As soon as the available device and its service are discovered, a guest in the house can request access to the node that is of particular interest, e.g. in order to extract the latest values from the temperature sensor attached to it. The guest's request is intercepted by the nodes PEP module which then forwards the request to the PDP, the latter running on the house owner's trusted device. The PDP has to consider all applicable policies from the PAP, enriched by any relevant information residing on the PIP, while additional ones might be added in real time regarding the specific access. For instance, a question can be displayed on the house owner's mobile phone regarding this access request giving the user the option to explicitly grant or deny access. Once all the required information has been collected, the PDP issues a decision which is sent back to the node's PEP. Based on that decision the PEP may or may not allow the guest to access said nodes data of interest. It should be noted that, on top of the decision taken on the request, the PDP might set one or more obligations for the PEP. An obligation is additional restrictions that should be taken into account when enforcing a decision, like the requirement to log any permitted access or to inform for unauthorized attempts. Moreover, prior to this communication the PAP should have set all applicable policies and policy sets for all targets in the network. These policies are made available to PDP for subsequent request evaluations.

In more detail, the data flow, as shown in Figure 6, consists of the following steps:

1. A remote entity (requester) requests access to the node's resources.
2. The PEP, sends to the context handler a decision query about this particular access request, together with some details, like target's and requester's names, and type of requested access.
3. Upon receiving the request the context handler requests additional information and attributes about the requester from a PIP. Note that this might be the result of PDP's demand
4. The PIP collects the requested attributes and returns them to the context handler which forwards them to the PDP together with the request. At this point the PDP might request additional attributes from the context handler which has to repeat the communication with the PIP.
5. The PDP evaluates the request against the policies and sends the authorization decision to the PEP.
6. The PEP fulfils optional obligations and enforces PDP's decision.

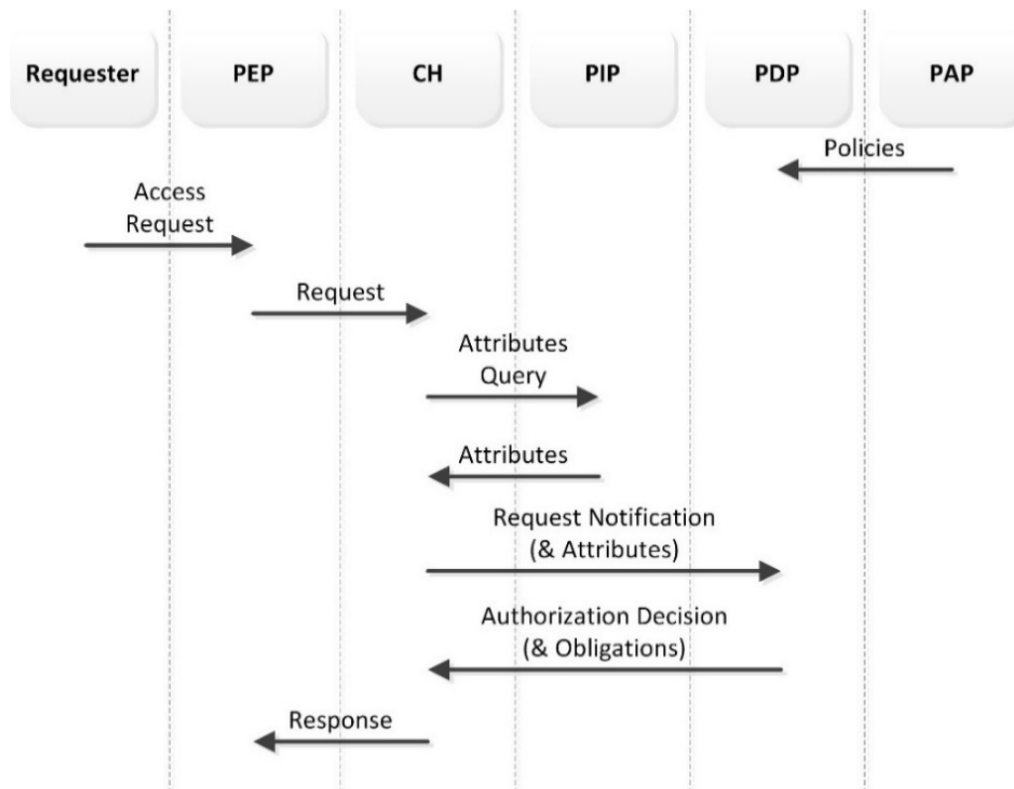


FIGURE 6. DATA FLOW MODEL OF POLICY-BASED ACCESS CONTROL

### 3.1.2 ACCESS CONTROL POLICIES

The uSPDM components are complemented by a well-defined set of XACML policies which define the rules that should be taken into account when examining access requests.

The main components of the XACML Policy are depicted in Figure 7. A policy set about a specific target consists of a number of applicable policies which in turn define a set of rules. Each rule contains information about the applicable Target, the effect, and additional Conditions. Note that the target does not only refer to resources. It might reference characteristics of a subject, resource, action or environment.

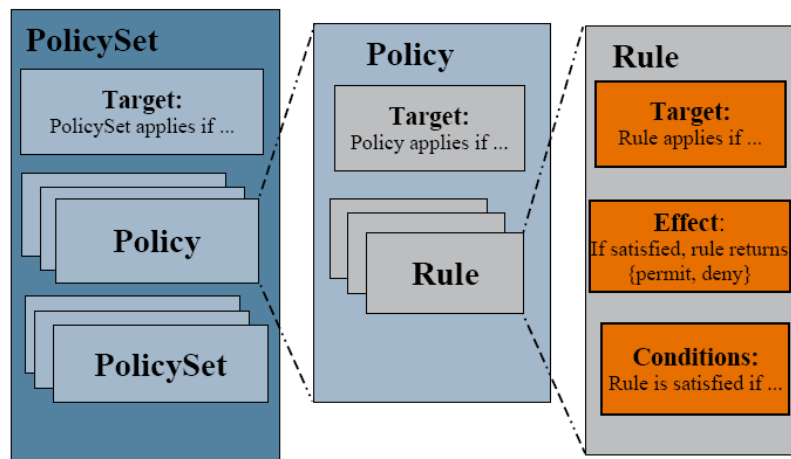


FIGURE 7: XACML POLICY COMPONENTS

#### RULE IMPLEMENTATION

A **rule** is the most elementary unit of **policy**, which is typically encapsulated within a policy. This also facilitates the exchange of rules among the stakeholders, i.e. PDP and PAP. The main components of a **rule** are:

- a **target**;
- an **effect**, indicates the **rule**-writer's intended consequence of a "True" evaluation for the **rule**. Two values are allowed: "Permit" and "Deny".
- a **condition**,
- **obligation** expressions, and
- **advice** expressions

#### POLICY IMPLEMENTATION

**Rules** are not exchanged amongst system entities. Therefore, a **PAP** combines **rules** in a **policy**.

A **policy** comprises four main components:

- a **target**;
- a **rule-combining algorithm**-identifier;
- a set of **rules**;
- **obligation** expressions and
- **advice** expressions

#### POLICY INFORMATION TEMPLATE

Table 2 defines the policy information template used to represent policies for controlling access to nSHIELD resources. This template is used to facilitate rules and policies defined by PAP (scenario owners in the context of the nSHIELD) to be imported into the system.

TABLE 2. POLICY ATTRIBUTES TEMPLATE

POLICY ATTRIBUTE	VALUE
---------------------	-------

<b>Policy ID</b>	A unique identifier that allows the policy to be referenced within a policy set.
<b>Rule Combining Algorithm</b>	The procedure for combining decisions from multiple rules. Valid values for this attribute are defined below.
<b>Description</b>	A textual description of the purpose of the policy. It typically provides information on most of the attributes found in this template.
<b>Policy Target</b>	The part of a policy that specifies matching criteria for figuring out whether a particular policy is applicable to an incoming service request. Contains three basic "matching" components: <b>Subjects</b> (An actor), <b>Actions</b> (An operation on a <i>resource</i> ), and <b>Resources</b> (Data, service or system component). Attributes of the subjects, resource, action, environment and other categories are included in the request sent by the PEP to the PDP.
<b>Effect</b>	The intended consequence of a satisfied rule. It can take the values "Permit" and "Deny". Note that this is not necessarily the authorization decision returned by the PDP to the PEP which, besides the above, can also include the values "Indeterminate" or "NotApplicable", and (optionally) a set of <b><i>obligations and advices</i></b>
<b>Condition (optional)</b>	Represents a Boolean expression that refines the applicability of the <b><i>rule</i></b>
<b>Obligations expressions (optional)</b>	Operation that should be performed by the PEP in conjunction with the enforcement of an authorization decision.
<b>Advice expressions (optional)</b>	A supplementary piece of information in a policy or policy set which is provided to the PEP with the decision of the PDP.

#### RULE- AND POLICY- COMBINING ALGORITHMS

The following algorithms are used for combining rules of a policy as well as policies from a policy set.

- **Deny-overrides:** It is intended for those cases where a deny decision should have priority over a permit decision
- **Ordered-deny-overrides:** The behavior of this algorithm is identical to that of the "Deny-overrides" rule-(policy-) combining algorithm with one exception. The order in which the collection of rules (policies) is evaluated SHALL match the order as listed in the policy (set).
- **Permit-overrides:** It is intended for those cases where a permit decision should have priority over a deny decision.
- **Ordered-permit-overrides:** The behavior of this algorithm is identical to that of the "Permit-overrides" rule-(policy-) combining algorithm with one exception. The order

in which the collection of rules (policies) is evaluated SHALL match the order as listed in the policy (set).

- **Deny-unless-permit:** It is intended for those cases where a permit decision should have priority over a deny decision, and an “Indeterminate” or “NotApplicable” must never be the result. It is particularly useful at the top level in a policy structure to ensure that a PDP will always return a definite “Permit” or “Deny” result. This algorithm has the following behavior.
  - 1. If any decision is “Permit”, the result is “Permit”.
  - 2. Otherwise, the result is “Deny”.
- **Permit-unless-deny:** It is intended for those cases where a deny decision should have priority over a permit decision, and an “Indeterminate” or “NotApplicable” must never be the result. It is particularly useful at the top level in a policy structure to ensure that a PDP will always return a definite “Permit” or “Deny” result. This algorithm has the following behavior.
  - 1. If any decision is “Deny”, the result is “Deny”.
  - 2. Otherwise, the result is “Permit”.
- **First-applicable (rule):** Each rule SHALL be evaluated in the order in which it is listed in the policy. For a particular rule, if the target matches and the condition evaluates to “True”, then the evaluation of the policy SHALL halt and the corresponding effect of the rule SHALL be the result of the evaluation of the policy (i.e. “Permit” or “Deny”). For a particular rule selected in the evaluation, if the target evaluates to “False” or the condition evaluates to “False”, then the next rule in the order SHALL be evaluated. If no further rule in the order exists, then the policy SHALL evaluate to “NotApplicable”.
- **First-applicable (policy):** Each *policy* is evaluated in the order that it appears in the *policy set*. For a particular *policy*, if the *target* evaluates to “True” and the *policy* evaluates to a determinate value of “Permit” or “Deny”, then the evaluation SHALL halt and the *policy set* SHALL evaluate to the *effect* value of that *policy*. For a particular *policy*, if the *target* evaluate to “False”, or the *policy* evaluates to “NotApplicable”, then the next *policy* in the order SHALL be evaluated. If no further *policy* exists in the order, then the *policy set* SHALL evaluate to “NotApplicable”.
- **Only-one-applicable (for policies only):** In the entire set of *policies* in the *policy set*, if no *policy* is considered applicable by virtue of its *target*, then the result of the policy-combination algorithm SHALL be “NotApplicable”. If more than one *policy* is considered applicable by virtue of its *target*, then the result of the policy-combination algorithm SHALL be “Indeterminate”. If only one *policy* is considered applicable by evaluation of its *target*, then the result of the *policy-combining algorithm* SHALL be the result of evaluating the *policy*.

#### POLICY EXAMPLE

In uSPBM the outcome of a request is “Permit”, “Deny” or “Not Applicable”. The evaluation of a request allows the PDP to make a decision based on a given policy or policy set, or combine all applicable rules and policies using a Rule-combining or Policy-combining algorithm respectively. Some of the standard combining algorithms used are “Deny-Overrides”, “Permit-Overrides”, “First-Applicable” and “Only-One-Applicable”.



An uSPBM sample XACML policy is depicted in Figure 8.

```
<?xml version="1.0" encoding="UTF-8"?>
<Policy
  xmlns="urn:oasis:names:tc:xacml:2.0:policy:schema:os"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="urn:oasis:names:tc:xacml:2.0:policy:schema:os
    access_control-xacml-2.0-policy-schema-os.xsd"
  PolicyId="0"
  RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-
algorithm:deny-overrides">
  <Target>
    <Subjects>
      <Subject>
        <SubjectMatch
          MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
            <AttributeValue
              DataType="http://www.w3.org/2001/XMLSchema#string">Subject</AttributeValue>
              <SubjectAttributeDesignator
                AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"
                DataType="http://www.w3.org/2001/XMLSchema#string"/>
              </SubjectMatch>
            </Subject>
          </Subjects>
          <Resources>
            <Resource>
              <ResourceMatch
                MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
                  <AttributeValue
                    DataType="http://www.w3.org/2001/XMLSchema#string">Resource</AttributeValue>
                    <ResourceAttributeDesignator
                      AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"
                      DataType="http://www.w3.org/2001/XMLSchema#string"/>
                    </ResourceMatch>
                  </Resource>
                </Resources>
                <Actions>
                  <Action>
                    <ActionMatch
                      MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
                        <AttributeValue
                          DataType="http://www.w3.org/2001/XMLSchema#string">Action</AttributeValue>
                          <ActionAttributeDesignator
                            AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
                            DataType="http://www.w3.org/2001/XMLSchema#string"/>
                          </ActionMatch>
                        </Action>
                      </Actions>
                    </Target>
                  <Rule
                    RuleId="urn:oasis:names:tc:xacml:2.0:conformance-test:IIA1:rule"
```

```

    Effect="Permit">
  <Target>
    <Subjects>
      <Subject>
        <SubjectMatch
          MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
            <AttributeValue
              DataType="http://www.w3.org/2001/XMLSchema#string">Subject
            </AttributeValue>
            <SubjectAttributeDesignator
              AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"
              DataType="http://www.w3.org/2001/XMLSchema#string"/>
            </SubjectMatch>
          </Subject>
        </Subjects>
        <Resources>
          <Resource>
            <ResourceMatch
              MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
                <AttributeValue
                  DataType="http://www.w3.org/2001/XMLSchema#string">Resource</AttributeValue>
                <ResourceAttributeDesignator
                  AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"
                  DataType="http://www.w3.org/2001/XMLSchema#string"/>
                </ResourceMatch>
              </Resource>
            </Resources>
            <Actions>
              <Action>
                <ActionMatch
                  MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
                    <AttributeValue
                      DataType="http://www.w3.org/2001/XMLSchema#string">Action</AttributeValue>
                    <ActionAttributeDesignator
                      AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
                      DataType="http://www.w3.org/2001/XMLSchema#string"/>
                    </ActionMatch>
                  </Action>
                </Actions>
              </Target>
            </Rule>
          </Policy>

```

FIGURE 8. USPDm POLICY EXAMPLE

## 3.2 SERVICE-ORIENTED ARCHITECTURES

While XACML defines the structure and content of access requests and responses exchanged among PEPs and PDPs, it does not provide any details regarding mechanism(s) used to transfer these messages, thus providing the necessary flexibility to adapt to diversified environments. Protocols that have been proposed for the communications among PEPs and PDPs are COPS [89], SNMP [90] and LDAP [28], which matches the requirements for accessing the policy repository and PIP.

Given the dynamic nature and the need for self-configurability, there will be situations where there is no coordination or central control over the network of nodes. Therefore, nodes that just joined the network have to discover which entity is responsible for making access decisions. In this case, the corresponding systems, such as power nodes or base stations, have to advertise their capabilities regarding PDP functionality while PEPs have to be able to discover PDPs and their corresponding provided services. Therefore, the communication mechanisms typically used alongside XACML were not appropriate for the ubiquitous computing deployments uSPBM is intended for.

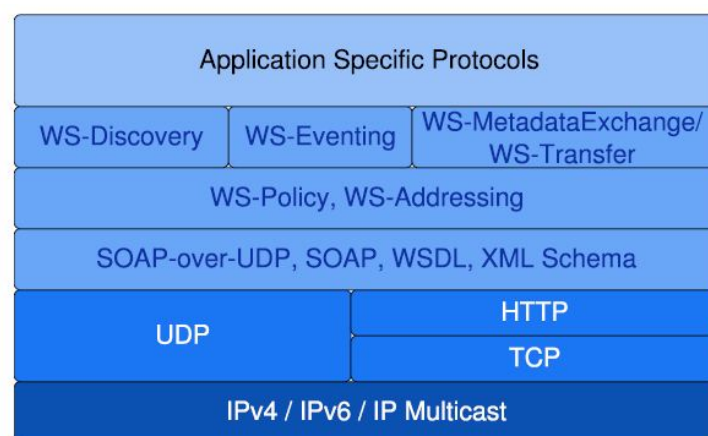
In the context of the IoT, existing networking mechanisms are updated to efficiently handle the vast population of ubiquitous resource-constrained devices (e.g. IETF's 6LoWPAN [30] for IPv6 over 802.15.4), higher level, machine to machine interactions, are often required to make use of the devices' full potential. The above have shifted researchers' and developers' focus on mechanisms that guarantee interoperability, providing seamless access to the various devices and their functional elements. Service Oriented Architectures (SOAs) have evolved from this need, providing interoperable, cross-platform, cross-domain and network-agnostic access to devices and their services. This approach has already been successful in business environments (e.g. [91]), as SOAs allow stakeholders to focus on the services themselves, rather than the underlying hardware, network technologies and architectures [92]. As SOAs are widely used, there is now an effort to apply this technology on embedded systems as well. To enable these services, various industry stakeholders introduced DPWS, a profile of Web Services protocols that enables Web Service messaging, discovery, description, and eventing on resource-constrained devices. DPWS messages are typically encapsulated in SOAP (Simple Object Access Protocol) envelopes and transported over any transport protocol, including HTTP and UDP using the SOAP-over-HTTP and SOAP-over-UDP bindings respectively [93], or even an SMTP binding which targets resource-constrained embedded devices and enables secure web service messaging, discovery, description, and eventing. More details on the specification follow below.

### 3.2.1 THE DEVICES PROFILE FOR WEB SERVICES (DPWS)

DPWS was introduced in 2004 by a consortium led by Microsoft and is now an OASIS open standard (at version 1.1 since July 2009). The DPWS specification defines a minimal set of implementation constraints to enable secure Web Service messaging, including discovery [94], description, interactions and event-driven changes [95] on resource-constrained devices. It employs similar messaging mechanisms as the Web Services Architecture (WSA, [96]), with restrictions to complexity and message size, allowing the provision of totally platform- and language-neutral services, similar to those offered by traditional web services.

The profile's architecture includes hosting and hosted services. A single hosting service is associated with each device while the same device may accommodate various hosted services. The latter represent the device's various functional elements and rely on the hosting service for discovery. Thus, a multifunctional sensor integrated into e.g. a smart home or enterprise environment, will have a single hosting service but may feature various hosted services (e.g. a temperature service, a light intensity service, a movement-sensing service etc.).

Furthermore, as discovery services are included, the device can advertise its presence on the network and search for other devices. Metadata exchange mechanisms provide dynamic access to service hosted on a device. Additionally, publish and subscribe eventing mechanisms allow clients to subscribe to services provided by devices. The DPWS protocol stack can be seen in Figure 9.



**FIGURE 9. THE DPWS PROTOCOL STACK [23]**

The DPWS specification enables the adoption of web services on embedded devices with limited resources, allowing system owners to enjoy these benefits across the heterogeneous systems of the IoT, and allowing us to overcome the important of interoperability barrier of current IoT deployments, by moving to what is often referred to as the “Web of Things” (see Figure 10).

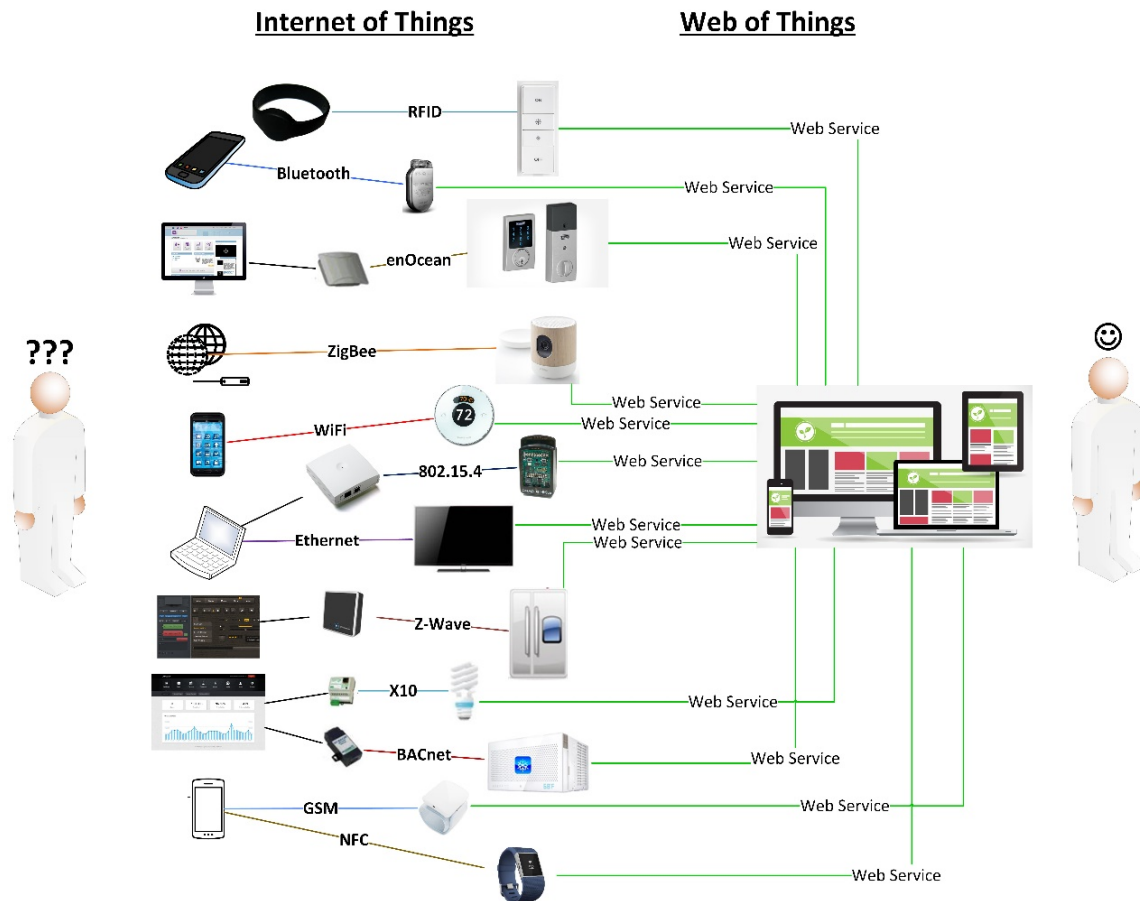


FIGURE 10. FROM THE INTERNET OF THINGS (LEFT) TO THE WEB OF THINGS (RIGHT)

Thus, DPWS can enable user-to-machine and machine-to-machine interactions in a unified manner, moving on from the current state of the field, where manufacturers offer a variety of proprietary protocols which are not interoperable and essentially lock-in users, forcing them to use a specific vendor/ecosystem. It is, therefore, the mechanism of choice for the underlying uSPBM communications and device interactions.

### 3.2.1.1 ALTERNATIVES TO DPWS

There is a variety of Service Discovery Protocols (SDPs) that can be used to provide seamless discovery and access to smart devices and their functional elements, each with its own set of features and intricacies. Other than DPWS, prominent solutions include the Service Location Protocol (SLP [97]) and the Universal Plug and Play (UPnP [98]).

These protocols provide mechanisms to search for devices and their services, select appropriate services and use them. In more detail, the devices typically rely on these protocols to perform the following tasks:

- **Discovery:** Discover pertinent devices on the network
- **Description:** Retrieval of descriptions of discovered devices, including hosted services and their descriptions (e.g. using WSDL [99], as is the case in DPWS).
- **Control:** Invocation of identified operations on selected devices
- **Eventing:** Subscription to and notification via service event sources

An analysis of the above protocols based on their key features and supported tasks can be found in Table 3.

**TABLE 3. COMPARISON OF SDPS**

	<b>DPWS</b>	<b>UPnP</b>	<b>SLP</b>
<b>Current Version</b>	1.1	1.1	2.0
<b>License</b>	Open	Open <sup>7</sup>	Open
<b>Initial developer</b>	Microsoft	Microsoft	IETF
<b>Security</b>	√		√
<b>Discovery</b>	√	√	√
<b>Description</b>	√	√	√
<b>Control</b>	√	√	
<b>Eventing</b>	√	√	

The comparison reveals that DPWS and UPnP offer a full set of features, including control and eventing mechanisms, but SLP only focuses on discovery and description of services and not their utilization, so it can be of limited use in the context of ubiquitous devices.

Moreover, while DPWS and UPnP appear to be similar in features, there are substantial differences between the two. The former is built around the OASIS Web Services (WS-) stack, offering significant benefits in terms of operational and development aspects, as well as of device-to-device and user-to-device communication. Deploying Web Services on devices allows the use of a single stack for local network and Internet interactions with devices and other Web Services, taking advantage of pre-existing tools and development know-how and leveraging the extensive work already carried out in defining the various WS-\* family of protocols (e.g. for security, an important aspect for which UPnP is lacking provisions). Thus, DPWS-enabled devices feature superior scaling and seamless integration capabilities, both in the context of enterprise environments (e.g. plant floors or enterprise-wide information systems) and the Internet (e.g. regular W3C-specified Web Services). Scalability is a key area where UPnP and SLP are found wanting. The former is suitable for small local networks, while the latter features some basic provisions (with the use of a Discovery Agent) which are limited to a local network or networks under common administration. Comparatively, the main disadvantage of DPWS is the existence of numerous specifications (protocols, bindings etc.), as is evident from Figure 9, which have not been consolidated yet.

It should be noted that DPWS was originally conceived and introduced as a successor to UPnP, but the lack of backward compatibility meant that such a transition has not taken place yet. Instead, nowadays DPWS is actively pushed by industry stakeholders as the solution of choice for large-scale enterprise (e.g. industrial) deployments, while UPnP is mostly targeted to the home environment (printers, home entertainment etc. [100]). Also, like UPnP, DPWS is natively integrated into the various versions of the Windows operating system, from Windows Vista onwards.

### 3.2.2 NODE.DPWS

---

<sup>7</sup> Membership to the UPnP Forum needed for certain features. Includes some proprietary protocols.

This section presents Node.DPWS, a novel implementation of the DPWS specification that focuses on the Node.js platform (<http://nodejs.org/>), also referred to as Node, a JavaScript-based runtime environment designed to maximize throughput and efficiency. The associated Node.DPWS libraries are the first to leverage the benefits of both DPWS and Node, allowing the creation of high performance, scalable and lightweight DPWS devices for the heterogeneous, often resource-constrained, platforms typically found in smart environments. Moreover, the libraries are easy to use and the devices can be defined with a minimum of code, reducing the development effort.

### 3.2.2.1 ALTERNATIVE DPWS LIBRARIES

A survey on alternative to Node.DPWS APIs for DPWS development reveals a plethora of available solutions with diverse characteristics. These include the libraries contained within Microsoft's .NET Micro Framework (<http://www.netmf.com>), the Web Services for Devices (WS4D, <http://ws4d.e-technik.uni-rostock.de>) toolkits which include WS4D-uDPWS, WS4D-JMEDS, WS4D-Axis2 and WS4D-gSOAP, and, finally, the Service-Oriented Architecture for Devices (SOA4D, <https://forge.soa4d.org>) solutions which include DPWS-Core and DPWS4J. Information for the identified DPWS implementations is aggregated in Table 4.

TABLE 4. OVERVIEW OF DPWS TOOLKITS

	.NET Micro Framework	WS4D-uDPWS	WS4D-JMEDS	WS4D-Axis2	WS4D-gSOAP	DPWS-Core	DPWS4J
<b>Language</b>	C#	C	Java	Java (Apache Axis2)	C	C	Java
<b>CPU/OS</b>	ARM	PC (VM), TelosB, AVR Raven	Java SE, Java CDC/CLDC, Android	Java SE	Linux, Windows, ARM	Linux, Windows	Java SE, Java CDC
<b>DPWS 1.0</b>	√	-	√	√	√	√	√
<b>DPWS 1.1</b>	√	√	√	-	Partial	Partial	-
<b>IPv4</b>	√	√	√	√	√	√	√
<b>IPv6</b>	-	√	√	-	Partial	√	-
<b>License</b>	Apache 2.0	FreeBSD	EPL	Apache 2.0	GPL/LGPL	LGPL	LGPL
<b>Active</b>	Yes	No	Yes	No	No	Yes	No

Nevertheless, when focusing on key features such as code portability, deployment on heterogeneous platforms, support for IPv6 (necessary for IoT applications) and active development and support of the tools, the valid options are actually fewer, with WS4D-JMEDS standing out as the most attractive choice. It is the most mature work of the WS4D initiative, providing a feature-rich platform which is being constantly updated and improved. It is, therefore, used as a benchmark to assess the potential benefits of Node.DPWS.

### 3.2.2.2 ABOUT NODE.JS

Node.js is a relatively new platform, introduced in May 2009 (in version v5.0.0, as of October 2015). It is an evented server-side implementation based on Google's V8 JavaScript engine. Both Node and the V8 engine are mostly implemented in C and C++, but Node's wrapper enhances the engine's basic features by allowing server-side deployment of JavaScript programs and the use of various C libraries, system calls, binary data manipulation and request handling. The core development concept was to create the building block for lightweight and scalable servers, providing an evented, non-blocking infrastructure for highly concurrent applications.

Node handles network input/output (I/O) operations in an evented, non-blocking fashion, while file I/O operations are handled asynchronously. This differentiates Node from typical implementations which follow the threaded model where for each new connection a thread is created, having inherent scaling issues. In Node, each new connection requires only a small heap allocation. Moreover, Node's executing thread cannot be blocked; in situations where this would normally happen (e.g. waiting for data from a remote database), the thread's runtime is utilized to serve other requests. The above result in very fast applications, which also scale well, even in the case of the resource-constrained devices (i.e. with no multi-core processors or large amounts of memory) typically embedded in smart environments. More details on Node's characteristics and code samples can be found in [101].

Research indicates that while Node.js offers significant benefits in terms of I/O performance and resource utilization, it is not as good at serving large static files [102]. This does not harm its applicability in developing fast, scalable network applications, and is expected to improve as the platform matures. Moreover, it is not an issue in the context of typical IoT applications, where end devices typically transmit low level information (e.g. sensing data) and receive commands on their functional elements (e.g. turn ON/OFF). Finally, some concerns regarding the security of Node.js applications can be attributed to the lack of a stable version and can be avoided by following security-conscious programming practices [103]. The platform is not inherently insecure and can safely be used in production-grade deployments.

As Node.js addresses many of the issues with real-time and lightweight application communications, it has quickly gained the support of the developers' community (there is a variety of libraries already available) and of major stakeholders in the industry [104], including companies such as Google, Microsoft and Yahoo!. Popular websites such as Wikipedia, LinkedIn, eBay and Microsoft's Azure cloud platform already make use of Node.js, even though it has not reached a stable version yet; an indicator that there is a strong demand for Node's features and that its user base will continue to grow over time.


### 3.2.2.3 *THE NODE.DPWS LIBRARIES*

Node's characteristics are a good match for event-driven web services deployable on the embedded devices expected to be present in smart enterprise, industrial, domestic and other ubiquitous-computing-enhanced environments. Thus, it is an attractive solution for implementing the DPWS specification, to potentially deliver highly scalable DPWS devices, able to handle many clients concurrently, while having very low resource consumption.



Node.DPWS provides such an implementation of DPWS using Node.js, supporting both versions 1.0 (2006) and 1.1 (2009) of the specification. The developer is responsible for describing the device's attributes (e.g. manufacturer or device name), its supported services (e.g. Temperature service), operations (e.g. get current temperature) and events (e.g. overheating alerts), and the libraries will properly advertise them and match them to requests. More complex operations are also supported by the library; e.g. allowing clients to subscribe to temperature readings at set intervals or when certain events occur, adopting the WS-Eventing specification [95]. Node.DPWS also supports auto-discovery, by implementing WS-Discovery [94], a multicast discovery protocol to locate services (the main mode of discovery being a client looking for target services). Apart from discovering devices, the developed library facilitates replies to discovery requests, forwarding the developer-defined device details to requesting nodes whose queries match the device.

A key characteristic of Node.DPWS is its compact code, which is also easy to use. Operations can be defined through minimal code and the developer only has to add the device's information and its operations (along with their callback functions, adding the desired functionality to the device). To highlight this, the source code of a DPWS device compared to the same device implemented using WS4D-JMEDS is provided in Figure 11.



```

var dpws = require('dpws')
var server = dpws.createServer({
  device: {
    address: '_IP_ADDRESS_',
    types: '_PORT_TYPE_',
    manufacturer: '_MANUFACTURER_',
    modelName: '_MODEL_NAME_',
    modelNumber: '_MODEL_NUMBER_',
    friendlyName: '_FRIENDLY_NAME_',
    firmwareVersion: '0.0.1',
    serialNumber: '12345'}})

var service = server.createService('_SERVICE_ID_', {
  types: {
    'temperature': 'int'}})

var temp = 0
service.createOperation('GetTemperature', {
  output: 'temperature'
}, function (input, cb) {
  cb(null, temp)})

server.listen(8080)
process.on('SIGINT', function () {
  server.bye(process.exit)})

```

```

import org.ws4d.java.communication.CommunicationException;
import org.ws4d.java.communication.DPWSCommunicationManager;
import org.ws4d.java.schema.SchemaUtil;
import org.ws4d.java.security.CredentialInfo;
import org.ws4d.java.service.DefaultDevice;
import org.ws4d.java.service.DefaultService;
import org.ws4d.java.service.InvocationException;
import org.ws4d.java.service.Operation;
import org.ws4d.java.service.parameter.ParameterValue;
import org.ws4d.java.service.parameter.ParameterValueManagement;
import org.ws4d.java.types.AttributedURI;
import org.ws4d.java.types.EndpointReference;
import org.ws4d.java.types.QName;
import org.ws4d.java.types.QNameSet;
import org.ws4d.java.types.URI;

public class SimpleDevice extends DefaultDevice {
  public SimpleDevice() {
    super(DPWSCommunicationManager.COMMUNICATION_MANAGER_ID);
    this.setEndpointReference(new EndpointReference (new AttributedURI ("_IP_ADDRESS_")));
    this.setPortTypes(new QNameSet (new QName ("_PORT_TYPE_")));
    this.addManufacturer("en-US", "_MANUFACTURER_");
    this.setModelName("_MODEL_NAME_");
    this.setModelNumber("_MODEL_NUMBER_");
    this.addFriendlyName("en-US", "_FRIENDLY_NAME_");
    this.setFirmwareVersion("0.0.1");
    this.setSerialNumber("12345");
    this.addService(new SimpleService());}

  class SimpleService extends DefaultService {
    public SimpleService() {
      super(DPWSCommunicationManager.COMMUNICATION_MANAGER_ID);
      this.setServiceId(new URI("_Service_ID_"));
      this.addOperation(new SimpleOperation());}

    class SimpleOperation extends Operation {
      public SimpleOperation() {
        super("GetTemperature");
        this.addOutputParameter("temperature", SchemaUtil.TYPE_INT);}

      @Override
      protected ParameterValue invokeImpl(ParameterValue arg0, CredentialInfo arg1)
        throws InvocationException, CommunicationException {
        ParameterValue output=createOutputValue();
        ParameterValueManagement.setString(output, "temperature", "SomeInteger");
        return output;}
    }
  }
}

```

**FIGURE 11. CREATING A SIMPLE DPWS DEVICE USING NODE.DPWS (LEFT) AND THE SAME DEVICE IN WS4D-JMEDS (RIGHT).**

In this example, the API's are used to expose a simple service providing temperature readings. In the case of Node.DPWS, the temperature operation is defined in a few lines of code:

input/output types are specified and then a handler is provided (called whenever the operation is invoked). And while Node.js code is generally compact, this high level of abstraction would not be possible without Node.DPWS, as the developer would have to deal with the low level aspects of the specification's implementation, such as the communications (sockets etc.) and all the XML parsing and processing, in her code. Choosing the most appropriate modules (e.g. server) where needed is a nontrivial task, especially for someone not familiar with the complexities of server-side JavaScript programming and the Node.js ecosystem. Further details on this can be gleaned by examining the freely available library sources, which also include samples to help familiarize developers with its functionality.

#### 3.2.2.4 *PERFORMANCE EVALUATION*

In order to assess the performance of Node.DPWS, we examined the behavior of a simple DPWS device featuring a "GetTemperature" operation which, when invoked, returned an integer value. Three versions of the device were developed: the Node.DPWS one and two versions using WS4D-JMEDS, one compiled using Java Standard Edition (SE) and the other following the Java Connected Device Configuration (CDC), part of the Java Platform Micro Edition (Java ME) designed for handheld and embedded systems.

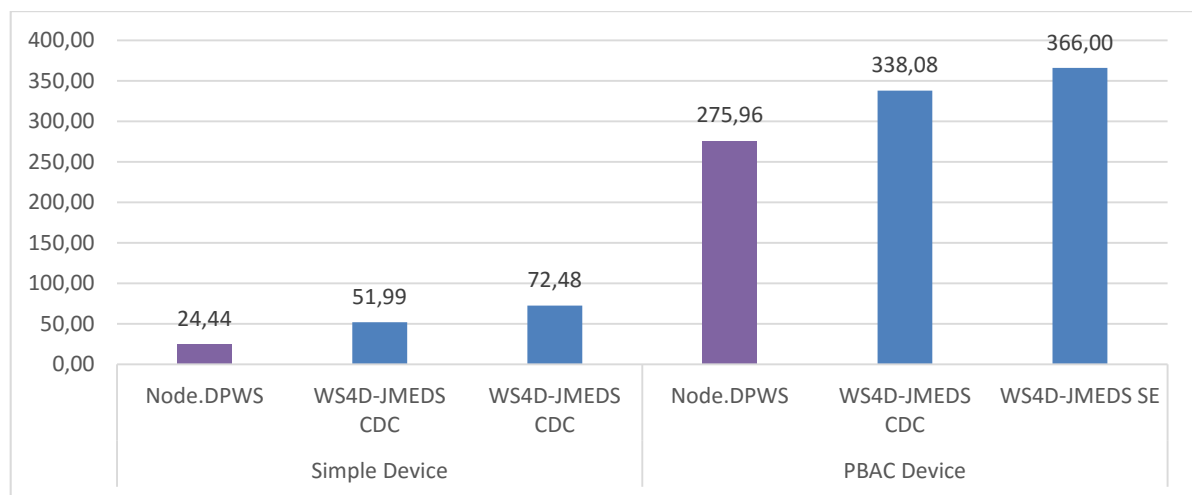
The applications were deployed on Beaglebone (<http://beagleboard.org/bone>) embedded platforms (720MHz ARM Cortex-A8, 256MB RAM, Arch Linux ARM operating system), interconnected via wired Ethernet to minimize the network's impact. The test-bed also featured a client application to discover and query the DPWS devices, recording response times.

A total of 500 requests were issued from the benchmarking client (running on a desktop PC) to each of the three DPWS devices, while various aspects of their performance were being monitored. Figure 12 (left side) presents the recorded response time, i.e. the time that the client had to wait to get a response to its "GetTemperature" invocation. The Node.DPWS device responded faster than the WS4D-JMEDS-based implementations. Averaging the response times over the 500 requests reveals that the Node.DPWS device performed significantly better than both the WS4D-JMEDS devices, featuring an average response time of 24.44ms, i.e. 53% and 66.3% faster than the CDC and SE variants respectively. In an alternative representation of this performance gap, the Node.DPWS device was able to handle 40.92 requests per second, compared to 19.3 and 13.8 requests for the CDC and SE devices respectively.

Moving to the DPWS device itself, CPU and memory load were recorded during benchmarking. There were no significant differences in terms of CPU load: the average load while handling the test requests was recorded at 90.4%, 96.7% and 91.9% for Node.DPWS, CDC and SE respectively. The Node.DPWS device demonstrated better behavior in terms of the memory consumed during benchmarks. Its memory footprint was measured at 26440 bytes on average, which was 10% lower than its WS4D-JMEDS CDC counterpart (29387 bytes) and 18% lower than the SE one (32280 bytes).

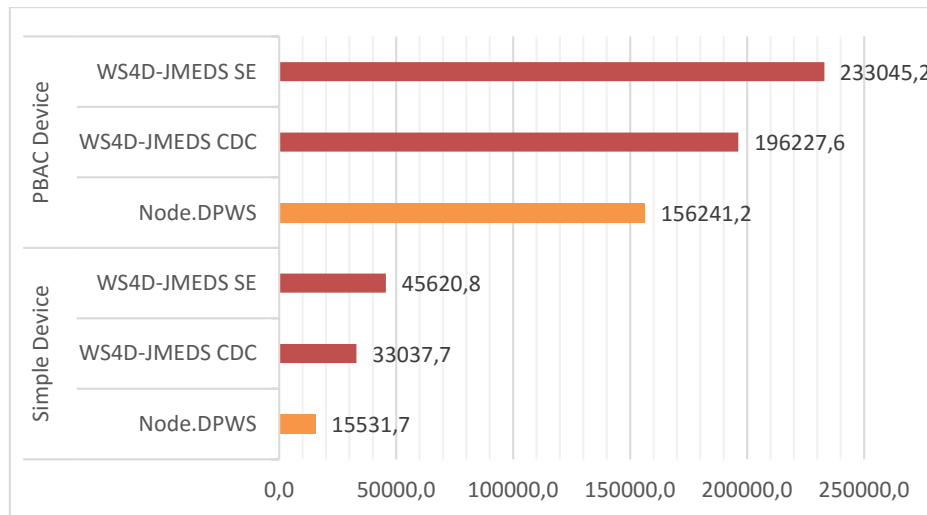
To examine the behavior of the APIs on complex applications, an implementation of uSPBM was also evaluated, as the framework involves complex communication mechanisms,

including automated discovery of devices, subscription and eventing. All of the framework's entities were developed using the investigated APIs, including the policy enforcement points (deployed on the Beaglebones) and the policy decision points and policy repositories (deployed on a desktop PC). Figure 12 (right side) presents the average response times for 500 requests to retrieve data via the pertinent (now access-control-protected) operation. The increased overhead of the access control mechanisms is evident, but the pattern formerly observed is maintained in this more demanding use case. The performance gap is not as evident as in the simpler scenario, as some of the delays are unrelated to the target device's implementation. For example, the device has to wait for the policy decision point to retrieve the relevant policies and issue a decision on the access request before responding to the benchmark client), but Node.DPWS still outperforms the alternatives, followed by JMEDS CDC (response time increased by 22.51%) and JMEDS SE (increased by 32.63%).



**FIGURE 12. AVERAGE RESPONSE TIME (IN MS) FOR 500 REQUESTS ON BOTH THE SIMPLE (LEFT) AND THE POLICY-BASED ACCESS CONTROL IMPLEMENTATION (RIGHT)**

Another important aspect, especially considering battery-powered IoT devices, is energy consumption. To this end, voltage and current were monitored during benchmarks, using hardware interfaces on the Beaglebones. The monitored consumption excluded the USB Host port and expansion boards, but these were not actively used during testing, thus their overhead should be minimal compared to that of the other hardware components. The energy consumed to serve 500 requests, on each of the investigated scenarios and implementations, are presented in Figure 13. As with the other examined parameters, the energy consumption gains when developing the devices using Node.DPWS are significant.



**FIGURE 13. ENERGY (IN MJ) REQUIRED TO HANDLE 500 REQUESTS.**

The advantages of adopting Node.DPWS are evident in the above results, but accurately quantifying the extent to which the noted differences can be attributed to the design of the investigated libraries or to the underlying environments (i.e. Node.js with its fast C/C++ -based engine and its event-driven, non-blocking design, set against the traditional Java environment) is nontrivial. Both libraries were designed with the intrinsic characteristics of the corresponding development environment in mind, so their behavior and features are, in great degree, interwoven with the platform they were built upon. Still, the scenarios were carefully chosen not to involve any unfair blocking requests (e.g. reading values from files), as this would exacerbate differences between the environments, simultaneously obfuscating differences between the libraries themselves.

Nevertheless, some design choices further differentiate the performance of the two libraries. The Elementtree XML parser chosen for Node.DPWS is more efficient and has a smaller memory footprint than the SAX parser used in JMEDS. Moreover, the Restify module employed in the presented library is an extremely lightweight HTTP server. Node.DPWS only binds services to a single developer-defined IP, while JMEDS automatically binds services to all available interfaces, thus consuming extra resources. Another key difference is that Node.DPWS only allows one instance of DPWS per device (still, all the desired elements can be exposed as different hosted services on this single hosting service). In contrast, JMEDS allows the deployment of multiple DPWS devices, an approach that adds complexity, as the middleware is responsible for managing the discovery, invocation and eventing mechanisms of all concurrently deployed devices, imposing extra processing overhead and delays.

### 3.2.2.5 SUMMARY

The above analysis demonstrates the benefits of leveraging the relatively novel Node.js platform to implement the DPWS specification in the form of Node.DPWS; an easy to use and lightweight set of libraries for creating and deploying DPWS devices on systems with limited resources. The performance assessment revealed that Node.DPWS outperforms the most attractive alternative currently available, the WS4D-JMEDS toolkit. The enhanced performance, scaling and compact code characteristics of the library show that there is significant room for improvement in the DPWS-related tools currently available. In the context

of the IoT, It is, therefore, worthwhile to pursue further work on the Node.DPWS implementation, in order to enrich its libraries with more features, e.g. extending it to support other WS-\* protocols (WS-Security being a good candidate).

The advent of the IoT leads to the modernization of software engineering, bringing it closer to the requirements of the ubiquitous computing world, with its plethora of heterogeneous, resource-starved end devices that will typically handle “small” (sensing) data. This intensifies the ever-present need of software practitioners to keep up to date with new tools and design approaches. Node.js is one such innovative platform and Node.DPWS aims to motivate researchers and developers alike to further explore the platform and the benefits of efficient, lightweight and scalable web services in IoT applications. Nonetheless, in software development one size does not fit all: the proposed library may be excellent for ubiquitous sensing devices but will have inherent limitations when used for backend devices handling large datasets. Thus, it is essential to examine use case scenarios at all development stages, selecting the appropriate tools for each application; an even more compelling argument for software practitioners maintaining an up-to-date and diverse skillset.

### 3.3 COMBINING THE TECHNOLOGIES

The combination of XACML with DPWS benefits both technologies, resulting in a solution that exceeds the “sum” of its parts.

In more detail, XACML helps DPWS by providing fine-grained access control at the operations level. This is important, as DPWS may have built-in security mechanisms, but offers no fine-grained access control; access can either be allowed to a resource, or denied completely. For many scenarios it is important to restrict the access of a service at the action level instead of at the service level only. Moreover, XACML allows devices to dynamically adapt the operation/features available to users, based on the environment and other variables. While dynamic reconfiguration of devices is not supported by DPWS, this functionality can be emulated by adjusting the applicable policies.

The XACML deployment also benefits by the adoption of DPWS. The most important gain pertains to the ease of deployment, as the adoption of DPWS facilitates seamless Machine-to-Machine (M2M) discovery and interactions, allowing the deployment of the XACML entities to any platform, anywhere on the network, with minimal involvement on behalf of the user. Moreover, the synchronous & asynchronous interactions enabled by DPWS, allow for reactive access control, providing the necessary variables (operational or situational context, i.e. the “Environment”) in a real-time, automated manner to the XACML entities. This improves upon the typical Policy-based Access Control approach of XACML, enabling more sophisticated, dynamic schemes (e.g. Risk Adaptive Access Control, RAdAC [105], see Figure 14)

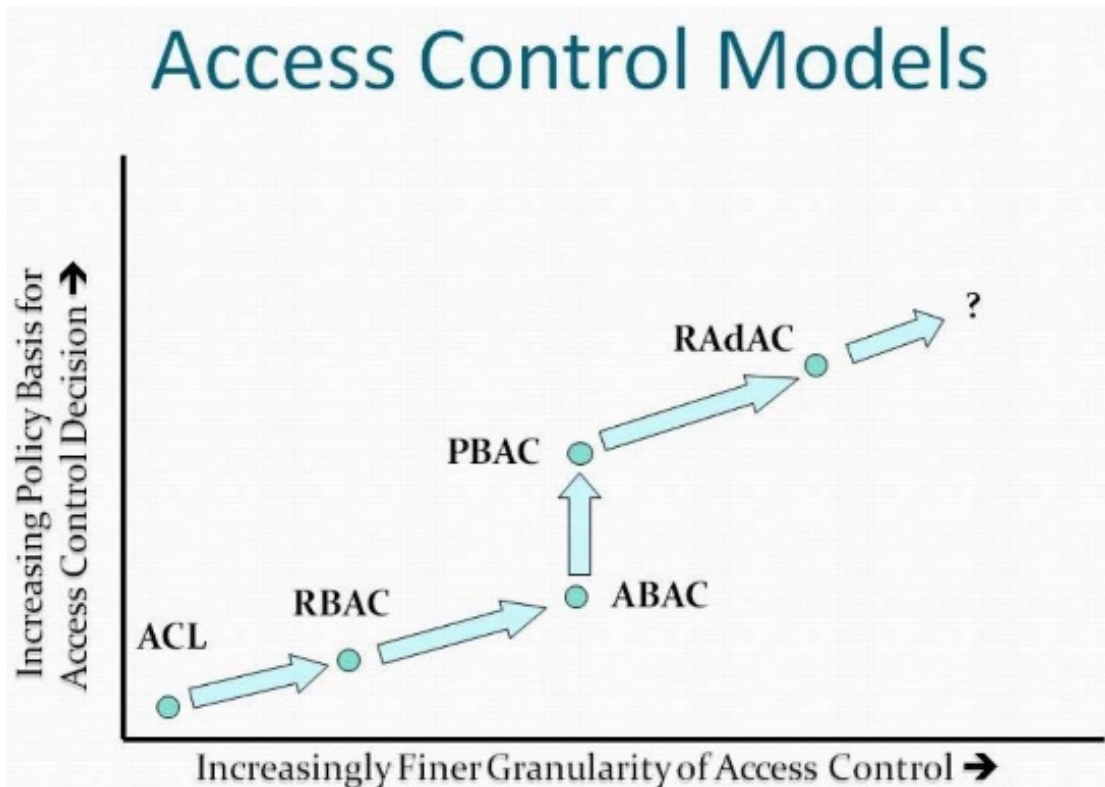


FIGURE 14. THE EVOLUTION OF ACCESS CONTROL MODELS [105]

## 4. IMPLEMENTATION APPROACH

The implementation of the framework's core entities (i.e. those dedicated to the authorization mechanisms) and their communications are detailed in the sections below.

### 4.1 NODE CLASSIFICATION

At the lower level of the uSPBM architecture lie the various computing devices that the framework's components are expected to be deployed. There is a plethora of heterogeneous devices that are expected to be found in ubiquitous computing environments, each with its own intrinsic characteristics in terms of size, performance, power source, networking capabilities etc. Given these diverse characteristics and hardware limitations, not all of uSPBM's mechanisms will be deployable on all types of devices. Therefore, it became apparent that it would help to classify the various platforms, depending on their intrinsic characteristics.

To this end, four different categories of smart devices were identified: power devices, mobile devices, embedded/micro devices and sensors/nano devices. These four categories represent the basic hardware platforms which (or subsets of which) are expected to be found in a typical uSPBM deployment, and cover the possible requirements of several market areas: from field data acquisition, to transportation, to personal space, to home environment, to public infrastructures, etc. A description of these categories can be found below.

#### 4.1.1 POWER DEVICES



Devices with medium to high performance in terms of computing power and no particular resource restrictions. Although these nodes are typically used as sink nodes and/or gateways to other networks they can also have their own on-board sensors for collecting additional information. Example of a power node is a laptop or a mains-powered embedded platform.

- Role
  - DPWS client and server (i.e. DPWS peers).
  - Responsible for interfacing with OSGi (Knopflerfish) framework.
  - Policy Administration Point, Policy Information Point/Attributes Repository and/or Policy Decision Point
- Underlying technologies
  - Windows or Linux operating system (optionally with desktop environment)
  - WiFi or Wired Ethernet, IPv4/IPv6 network
- Prototype platforms
  - Beagleboard/Beagleboard xM [106]
  - Typical PC/Laptop/Server

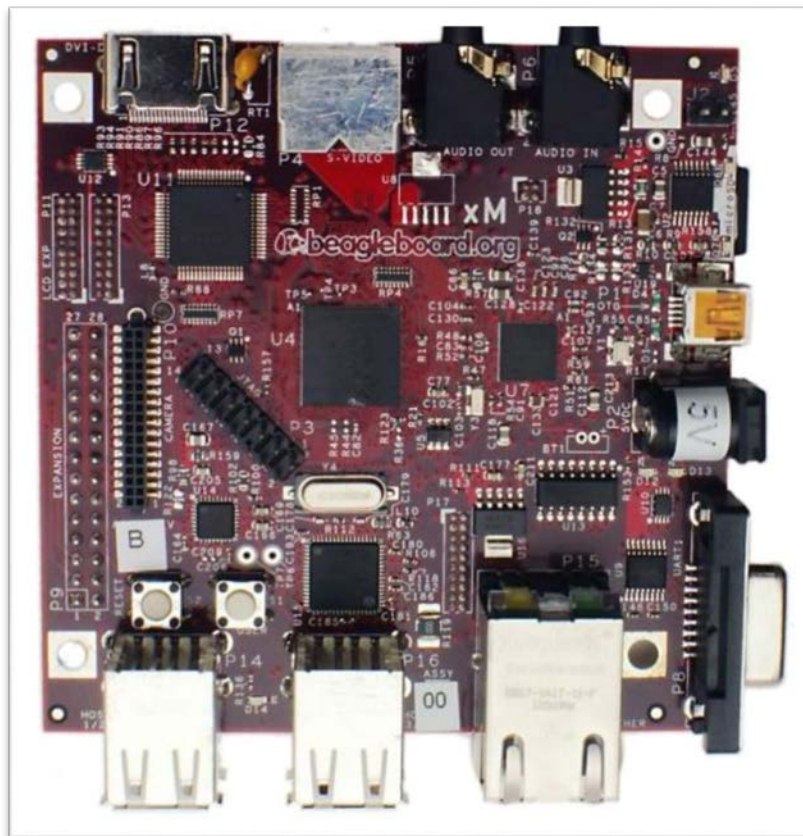


FIGURE 15. BEAGLEBOARD XM

#### 4.1.2 MOBILE DEVICES

Portable devices, often associated with a person. Their characteristics may vary from powerful devices with energy restrictions (e.g. smartphones) to devices with more resource constraints (e.g. devices attached onto smart clothing, smart watches etc.).

- Role
  - DPWS client and server (i.e. DPWS peer).
  - Policy Enforcement Point and/or Policy Decision Point
- Underlying technologies
  - Android mobile operating system
  - WiFi, IPv4/IPv6.
- Prototype platforms
  - Smartphone
  - Tablet



**FIGURE 16. SMARTPHONE AND TABLET DEVICES**

#### 4.1.3 EMBEDDED/MICRO DEVICES

Smaller devices, typically integrated into other entities (smart appliances, smart vehicles etc.) with limited capabilities and resources, such as computational power, memory, storage space and energy.

- Role
  - DPWS client and server (i.e. DPWS peer).
  - Policy Enforcement Point and/or Policy Decision Point
  - Bridge between 802.15.4/6LoWPAN and IPv4/IPv6 networks (optional)
- Underlying technologies
  - Angstrom or lightweight Ubuntu distribution operating system
  - WiFi or Wired Ethernet, IPv4/IPv6. Optionally 802.15.4/6LoWPAN.
- Prototype platforms
  - Beaglebone [107]



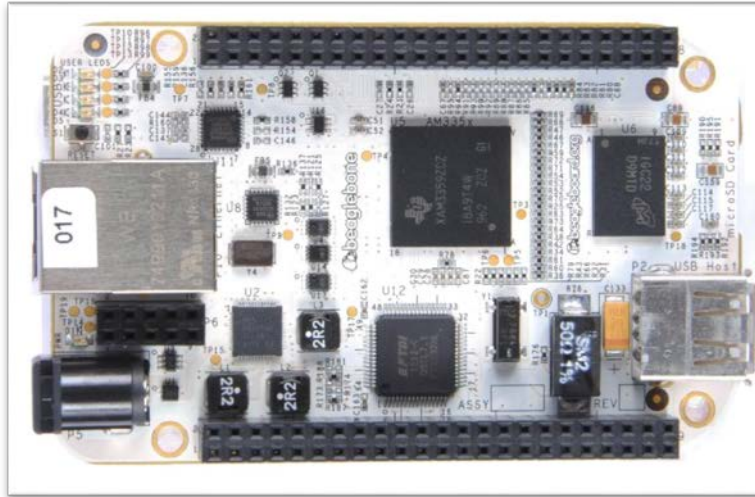


FIGURE 17. BEAGLEBONE EMBEDDED PLATFORM

#### 4.1.4 SENSORS/NANO DEVICES

Small battery-powered devices with very limited capabilities and resources; typically sensor motes.

- Role
  - Devices running a DPWS service.
  - Policy Enforcement Point (optionally)
- Underlying technologies
  - Contiki or Java ME Squawk operating system
  - 802.15.4/6LoWPAN network
- Prototype platforms
  - SunSPOTs [108]
  - Crossbow Technology IRIS motes <sup>8</sup>



FIGURE 18. SUNSPOT SENSOR

<sup>8</sup> <http://bullseye.xbow.com:81/Products/productdetails.aspx?sid=264>



FIGURE 19. IRIS MOTE

Some specification highlights of the abovementioned prototype platforms appear in Table 5.

TABLE 5. PROTOTYPE PLATFORM SPECIFICATIONS

	Sensor/Nano		Embedde d/Micro	Power Devices			Mobile Devices	
	Memsic IRIS	SunSP OT	Beaglebone	BeagleBoard	BeagleBoard- XM	Laptop	Mobile Phone	Tablet
Operating System	TinyOS / Contiki (port)	Java J2ME CLDC 1.1	Linux	Linux / WinCE	Linux / WinCE	Linux / Windows	Android	Android
Processor	Atmel ATMega 1281 @ 16MHz	ARMv4 T @ 180 MHz	MPU @ 720 MHz (OMAP)	MPU @ 720 MHz (OMAP)	DSP @ 1 GHz (DM3730)	Quad-core i5 @ 2.6GHz	Single-core Cortex-A5 @ 600MHz	Quad- core @ 1.9GHz
Memory	8KB RAM, 4KB EEPROM	512KB RAM	256MB RAM	256MB RAM	512MB RAM	8GB RAM	512MB RAM	2GB RAM
Storage	128KB Program Flash Memory, 512KB Measure ment (Serial) Flash	4MB Progra m Flash Memor y	SD/MMC/SD IO card slot	SD/MMC/SDI O card slot	SD/MMC/SDI O card slot	512GB, SD/MMC/SD IO card slot	16GB, SD/MMC/SDI O card slot	32GB, SD/MMC /SDIO card slot
Power	Batteries (2xAA)	3.7V Battery	USB / DC	USB / DC	USB / DC	Built-in battery / DC	Built-in battery / DC	Built-in battery / DC

Networking	2.4 Ghz radio	2.4 Ghz radio	Wired Ethernet, No built-in wireless module (but has interface)	Wired Ethernet, No built-in wireless module (but has interface)	Wired Ethernet, No built-in wireless module (but has interface)	Wired Ethernet, WiFi, Bluetooth	WiFi, Bluetooth	WiFi, Bluetooth
------------	---------------	---------------	---	---	---	---------------------------------	-----------------	-----------------

## 4.2 DPWS IMPLEMENTATION OF ACCESS CONTROL MECHANISMS

The section below details the approach followed to implement the authorization entities' interactions using web services and the APIs of choice.

### 4.2.1 PEP-PDP COMMUNICATION

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 that accesses these USPBM-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 Figure 20.

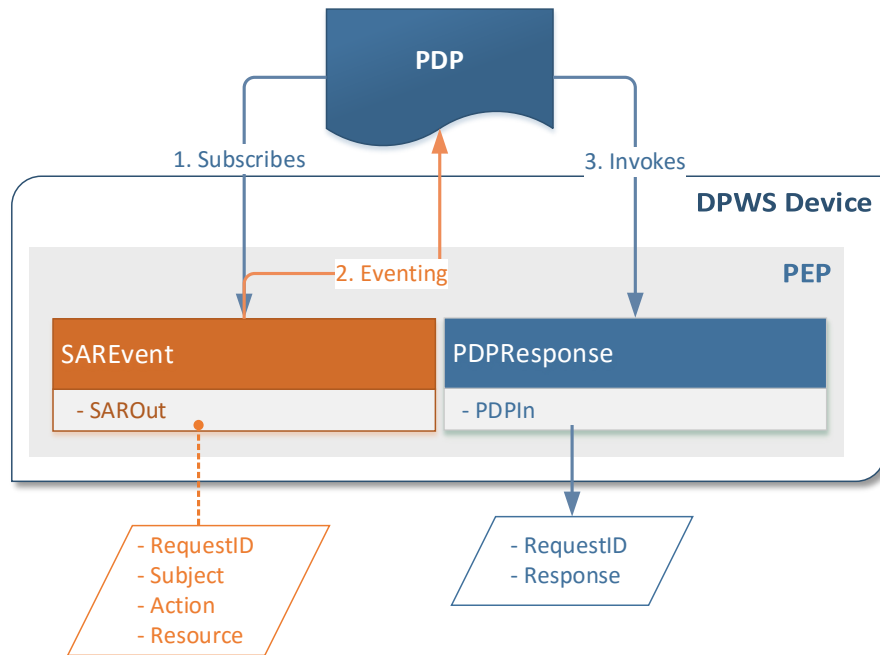


FIGURE 20. PEP-PDP IMPLEMENTATION

#### 4.2.2 PDP-PIP/PAP COMMUNICATION

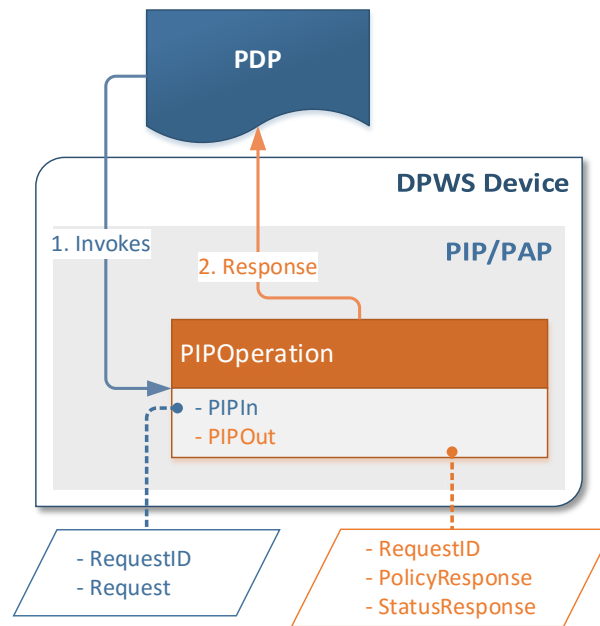
In terms of the discovery and information exchange that must take place between infrastructure entities (PDP, PIP, PAP), an extra operation must reside with the entity that stores the active policy set (namely the PIP/PAP). In specific, this extra operation is formed as follows:

- **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. On 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 Figure 21.



**FIGURE 21. PDP-PIP/PAP IMPLEMENTATION**

### 4.2.3 INFORMATION FLOW

The information flow that takes place whenever a request to an uSPBM-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 an uSPBM-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 [109].

### 4.3 MESSAGE PROTECTION

The uSPBM's 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. The exchange of unprotected policy messages might reveal useful information to attackers who will try to easily identify policy restrictions. This would allow them to do a mapping of the security measures taken for the specific environments hence exploit potential vulnerabilities. Moreover, in a more active approach, an attacker might modify policy related messages, such as authorization requests and/or decisions, obligations or advices in an attempt to downgrade adopted measures and bypass access controls. Masquerading is another threat to the system's integrity, where an attacker sends forged messages pretending to be a legitimate user. Lack of authorization requirements on requests might allow an attacker to make such message injections.

To avoid the aforementioned problems, security measures have to be taken to protect message confidentiality, integrity and authentication. These measures can be deployed in various layers in the OSI protocol stack, such as the application layer, the network layer or even the link layer.

At the lower layers, as existing networking mechanisms are updated and adapted to efficiently handle the vast population of the resource-constrained devices (e.g. work on the 6LoWPAN [30]), the pertinent cryptographic primitives are also adapted and improved accordingly. Such an example is the IPsec protocol and its variants that utilize header compression [110]–[112] which can provide similar levels of protection while preserving the valuable node resources. The security mechanisms inherent to the IEEE802.15.4 link-layer protocols [113] are also a viable option, though a comparative analysis made between IPsec and IEEE802.15.4 link-layer security [114], in some cases IPsec scales better while also offering end-to-end security.

Some prominent alternative schemes protect messages at the application or network layer and can provide end-to-end message protection. Well-known security mechanisms for these layers are the TLS (Transport Layer Security) protocol [115] and its counterpart proposed for securing UDP messages, namely DTLS [116]. TLS is suitable for providing confidentiality, integrity and authentication of messages exchanged between a client and a service, running on a node. However, the inherent expensive computations of TLS or IPsec (such as the TLS handshake protocol) do not match the requirements of resource-constrained environments, for the protection of the communications that take place among the less powerful micro/nano nodes (PEPs), or between PEPs and power nodes (PDPs). A TLS implementation on the Sun SPOT (Small Programmable Object Technology) Java-enabled WSN platform, has shown that the network lifetime is reduced by 70% [117].

Several lightweight alternatives based on TLS/SSL have been proposed for resource-constrained environments. One such approach was proposed in [118], which uses ECC for key exchange and authentication, RC4 for encryption and MD5 for integrity check. According to

the presented experimental results, it was able to complete a full SSL handshake within 2 seconds. Tiny 3-TLS proposed in [119] is an extension and adaptation of the TLS handshake sub-protocol, tailored for securing communications between sensing nodes and remote monitoring terminals. This protocol relies on the existence of an intermediate node, the sink node, which in the proposed framework can be assigned to a power node. In [120] an implementation of SSL was attempted through an enhanced version of *Sizzle* (a tiny-footprint HTTPS stack [121]). Measurements were performed on Telos motes and it was concluded that the exploitation of features such as session reuse and persistent HTTP(S) can spare multiple executions of the key exchange phase, which is the most energy-demanding part of the protocol. What is more, it was also shown that the extra cost for encrypting/authenticating application data with SSL is around 15%. Again, the key exchange phase is performed via ECC since it is significantly more efficient than the RSA alternative.

Other schemes focus on efficiently providing authenticated encryption, like the Identity-based Cryptosystem (IBC) signcryption mechanisms presented by Fagen et al [122]. Related to the above is the relatively novel concept of security fusion, whereby weak point-to-point properties are combined in order to produce strong security properties in a resource-aware manner [123].

The choice of security mechanisms is affected by many parameters, including the node capabilities. Communications between, e.g. a PDP running on a power node and the policy repository, can be protected using TLS as the power node can support the heavy computations required by this protocol. Given that there are no resources restrictions on power nodes there are no major obstacles, besides key management, to deploy TLS or even IPsec on these nodes. Nevertheless, considering the less powerful micro or nano nodes, deployed protection mechanisms have to be based on lightweight cryptographic protocols that satisfy the needs and match the capabilities of constrained environments.

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. Thus, in the case of uSPBM we make no assumptions about the existence of other security mechanisms and focus on solutions at the application layer instead.

#### 4.3.1 ASYMMETRIC VARIANT

A solution that can be adopted for this purpose is the Web Services Security Specification (WS-Security or WSS [124]) and the corresponding cryptographic mechanisms, i.e. XML encryption and signatures (enveloped, enveloping or detached) or a combination of those depending on the particular requirements to provide confidentiality and integrity of the exchanged messages. WS-Security 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), in order to provide message-level confidentiality, integrity and authentication. The main mechanisms detail signing SOAP messages (integrity, non-repudiation), encrypting SOAP messages (confidentiality) and attaching security tokens to SOAP messages (authentication). There is a variety of supported encryption, signature and security token formats (e.g. SAML Assertions [125], Kerberos tickets



[126], X.509 Certificates [127], Rights Expression Language (REL) Tokens [128], as well as custom tokens).

A structured exchange of secure XACML messages using XML encryption and signature is provided by SAML specifications (Security Assertion Markup Language [129]), thus offering the required protection at the application layer. SAML is a platform-independent, XML-based standard for exchanging authentication and authorization information. SAML assertions are typically transferred embedded using HTTP or XML-encoded SOAP messages that are transferred over HTTP or UDP [93]. OASIS has defined a profile in [109] for the integration of SAML with XACML and, among the others, the use of SAML for the secure transmission of XACML requests and responses.

Therefore, considering the networking requirements and the corresponding security mechanisms, the most appropriate options for securing XACML messages at the application layer, while providing interoperability, are the following:

- SAML-integrated XACML messages transferred using the SOAP protocol
- SOAP-encapsulated XACML messages protected with TLS. Such an approach requires using expensive TCP communications to transmit the resulting TLS messages. Another option is to use an adaptation of TLS over UDP, namely DTLS.

Both of the above solutions are typically independent of the protocols used at the underlying layers, allowing them to adapt in many environments, hence satisfying the interoperability requirement.

In either of the aforementioned approaches, one of the main problems related to the exchanged messages' protection is key management, especially when considering the following:

1. Communications might take place ad-hoc between nodes that do not have an established trust relationship, hence they do not (pre-)share any secrets. Dynamic structures and self-configuration capabilities demand for more flexible mechanisms.
2. Some nodes might not support public-key technologies, which further complicates the processes of establishing trust relationships and keys.

The inherent key management problems, especially in resource-constrained environments, have attracted a lot of attention in the research community and many schemes have been proposed within this context. A survey and taxonomy on proposed wireless sensor networks key management schemes is provided in [130].

Web Services Secure Conversation (WS-SecureConversation, [131]) is a WS-Security add-on which introduces, similarly to TLS, a session key to secure communication across one or more messages. The aim of the specification is to establish security context, share, renew, amend or cancel said context as well as derive (potentially more efficient) session keys from the aforementioned context. When multiple message exchanges are involved WS-SecureConversation has proven to be more efficient than a plain WS-Security implementation [132], but the former requires the presence of other WS-\* protocols as well, like WS-Trust, so the added complexity should also be considered.

Unlike TLS, WS-Security can offer end-to-end (message-level) security and it is more flexible when application-level proxy servers are involved. Still, the performance overhead is significant with the standard WS-Security implementation (see Table 6) and further work is required to improve its usability in resource-constrained devices.

**TABLE 6. WS-SECURITY AND TLS BENCHMARK RESULTS FOR 25 CONCURRENT REQUESTORS [133]**

Security Mechanism	Messages/sec	CPU load	Throughput (kB/s)
X.509 XML Signature & Encryption	352	99	2,403
WS Secure Conversation XML Signature & Encryption	798	98	5,679
SSL with HTTP Basic	2,918	95	3,181
None (message routing only)	5,088	96	5,419

Therefore, other than the baseline uSPBM entities communicating in plaintext, a variation of the proof-of-concept implementations was also developed which adopted mechanisms specified in WSS along with TLS (the latter being recommended in the DPWS specification).

Moreover, a symmetric authenticated encryption mechanism was also developed in used in some uSPBM implementations, to provide for a more efficient and lightweight security mechanism.

#### 4.3.2 SYMMETRIC VARIANT

Considering the resource-constrained nature of the devices where the uSPBM framework may be deployed as well as the need to minimize its performance impact in general, symmetric cryptographic primitives could be considered more appropriate.

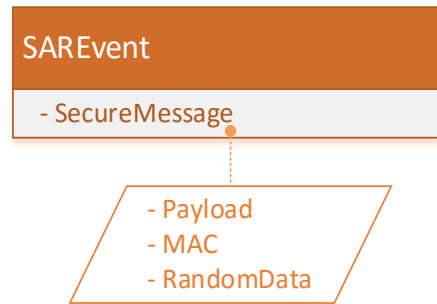
To this end, a security mechanism based on the AES/CCM [134] authenticated encryption algorithm was implemented and was deployed to protect both PEP-PDP as well as PDP-PIP/PAP communications, using pre-shared 256-bit keys. The utilization of the AES/CCM algorithm guarantees that the uSPBM-related messages 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, the operations are in this case 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 [134].

Thus, for example, the SAREvent operation now takes the form appearing in Figure 22. All other operations, i.e. PDPResponse and PIPOperation, are also adjusted accordingly.



**FIGURE 22. SECURE SAREVENT**

With regard to the symmetric keys, each node shares a pre-installed key with the PDP, which is installed to the nodes during installation. In more details, the PDP holds:

- A master key for communication with devices: **DMK**
- A master key for communication with infrastructure nodes (e.g. PAP/PIP): **IMK**

Moreover, the PDP shares a Shared Master Key (**SMK**) with each of the PEP-equipped devices it needs to communicate with. Each of these SMKs is generated using the DMK and the corresponding device's universally unique identifier (UUID), using the AES algorithm, as follows:

$$SMK = AES_{DMK}(Device\_UUID)$$

Equivalently, the PDP also shares a Shared Infrastructure Master Key (**SIMK**) with each of the infrastructure entities (e.g. PAP/PIP) that it needs to communicate with. This SIMK is derived using the IMK and the corresponding UUID of the infrastructure device, again using the AES algorithm, i.e.:

$$SIMK = AES_{IMK}(PAP\_UUID)$$

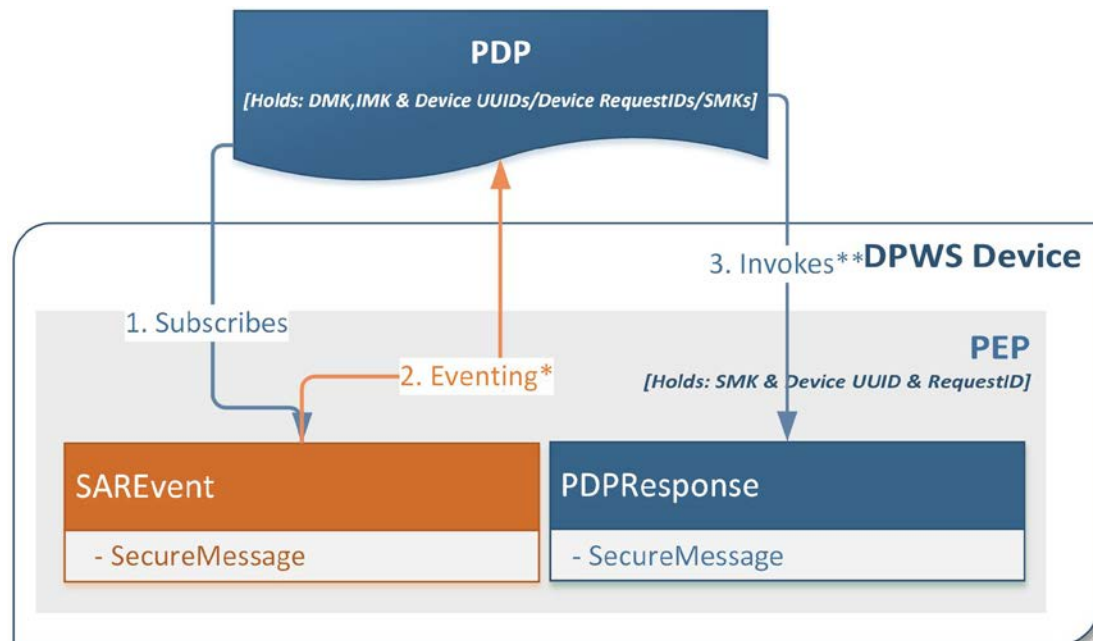
So, the PDP holds all the values needed to communicate with each device, namely the counter (i.e. RequestID) of the latest message exchanged and the key shared with the corresponding device.

The RequestID is not only used to guarantee freshness, but also to create the Nonce needed by the AES/CCM algorithm (the Nonce is the RequestID concatenated with random bytes):

$$Nonce = RequestID \parallel Random\_bytes$$

These random bytes must also be included in the message sent to the PDP (so that it can reproduce the Nonce from the shared/expected RequestID and those random bytes).

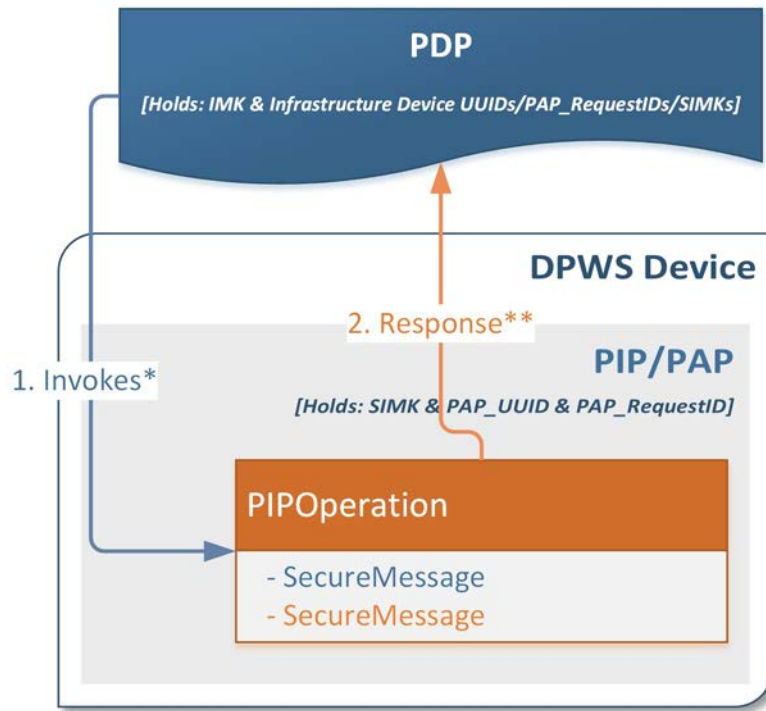
Thus, the PDP-to-PEP implementation of Figure 20 now takes the form appearing in Figure 23, and the PDP-to-PIP/PAP implementation of Figure 21 now takes the form appearing in Figure 24.



\* 2.Eventing = AES\_CCM(SMK, Nonce, SAROut || random\_bytes, EAD)

\*\* 3.Invokes = AES\_CCM(SMK, Nonce, PDPIIn || random\_bytes, EAD)

FIGURE 23. IMPLEMENTATION OF AES/CCM-PROTECTED PEP-PDP COMMUNICATION



\* 1.Invokes = AES\_CCM(SIMK, Nonce, PIPIn || random\_bytes, EAD)

\*\* 2.Response = AES\_CCM(SIMK, Nonce, PIPOut || random\_bytes, EAD)

FIGURE 24. IMPLEMENTATION OF AES/CCM-PROTECTED PDP-PIP/PAP COMMUNICATION

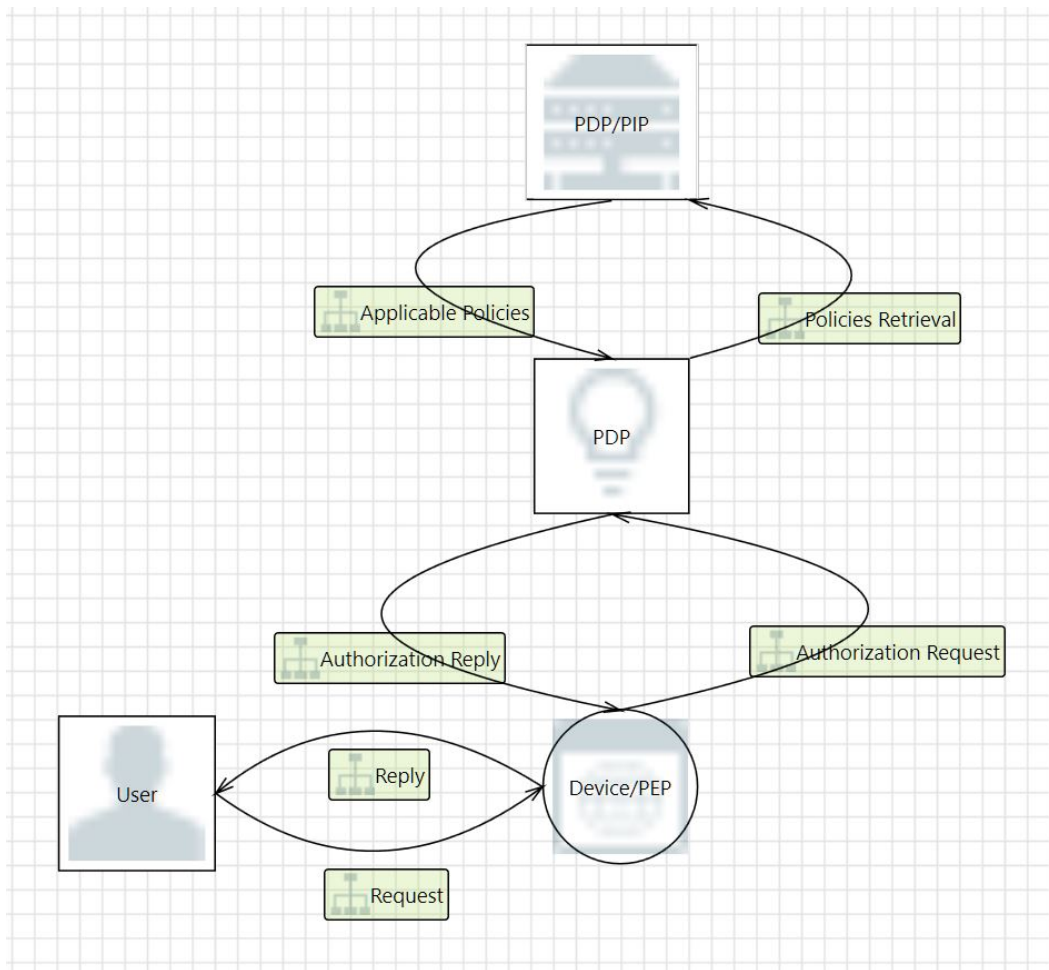
#### 4.3.3 SECURITY ANALYSIS

We assume that legitimate clients possess valid credentials (pre-shared keys or certificates, where applicable) and that they do not cheat intentionally. Still, compromised nodes (e.g. by malware) from local or remote networks cannot be excluded. Moreover, the development of the devices' provided services and the interactions with their clients should be judged in a case-by-case basis and are beyond the scope of this analysis.

In terms of generic threats, attackers could retrieve keys from compromised nodes. Thus, private keys should be stored securely (e.g. on a TPM module). During the discovery phase, if discovery messages use signatures, the integrity and identity of the client can be verified. Otherwise, a malicious entity inside the home network can determine the presence of services (but not use them, if protected by HTTPS). For cross-domain communications, interactions between the MQTT broker and its clients (uSPBM Proxies) can be protected via TLS, while application-level payload encryption can allow for end-to-end protection of published messages (safeguarding communications from compromised brokers).

We analyzed the core (i.e. the access-control related) interactions of uSPBM using Microsoft's Threat Modeling Tool (2016 version<sup>9</sup>), using STRIDE (Spoofing, Tampering, Repudiation, Information disclosure, Denial of service, Elevation of privilege) for threat risk modeling, as appearing in Figure 25.

<sup>9</sup> <https://www.microsoft.com/en-us/download/details.aspx?id=49168>



**FIGURE 25. STRIDE ANALYSIS OF USPB'S CORE ACCESS CONTROL INTERACTIONS**

Based on this analysis the following threats were identified:

- Improper data protection of PDP/PIP can allow an attacker to read information not intended for disclosure. (Information Disclosure / Tampering). To this end, it is important for authorization settings (including the security policies) to be stored securely and be regularly reviewed.
- Device/PEP may be able to impersonate the context of PDP in order to gain additional privilege. (Elevation of privilege). Thus, the devices must be protected from compromise (by malware etc.).
- Device/PEP may be able to impersonate the context of User in order to gain additional privilege. (Elevation of privilege). Again, this highlights the importance of protecting devices from compromise.
- An attacker can read or modify data transmitted over an authenticated flow from device to user. (Information Disclosure / Tampering). Thus, the provided services should feature mechanisms that protect the confidentiality and integrity of their interactions with the users.

## 4.4 INTERFACING WITH MIDDLEWARE & MANAGEMENT SYSTEMS

The OSGi Alliance is an open source standards organization that specified the OSGi standard [78], a module system and service platform for Java that implements a complete and dynamic component model. Applications offered by components are formed as bundles for deployment and can be remotely installed, started, stopped and uninstalled without requiring a reboot. To this end, we use Knopflerfish [135], the leading universal open source OSGi implementation.

In uSPBM's backend nodes, DPWS is integrated into OSGi by implementing operators that are controlled by the relevant system components via an OSGi interface (see Figure 26. Embedded devices specify their type and the provided services in DPWS. Each system component implements a component operator bundle that handles its underlying DPWS devices and their services. Thus, uSPBM's infrastructure entities can be deployed as OSGi bundles and can trivially be interfaced with other OSGi-based middleware entities and services, which are common in the literature (see "Background & Related Work" section).

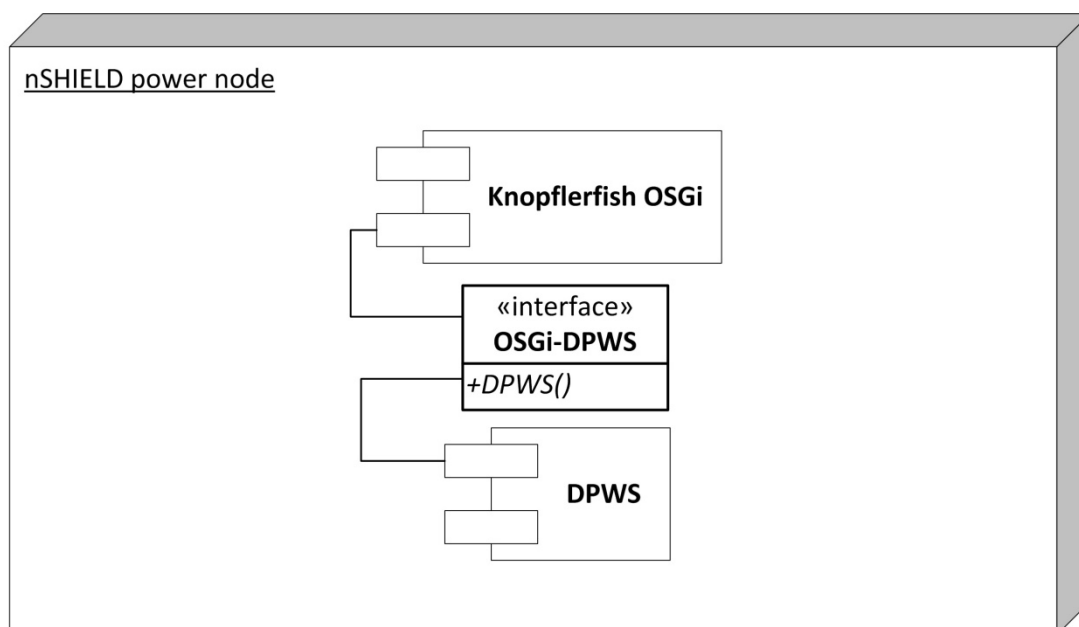


FIGURE 26. OSGi - DPWS INTERFACING

Possible features that the DPWS-OSGi interface could accommodate include:

- Allowing OSGi clients to use uSPBM device services and to deploy OSGi proxy services for DPWS services.
- Allowing uSPBM clients to use OSGi services.
- Event-based OSGi-uSPBM services communication (e.g. subscription to uSPBM events by OSGi bundles)
- OSGi-based uSPBM service creation & management

Moreover, other than the entities devoted to the uSPBM architecture itself, an extra entity was developed to act as a gateway to help interface uSPBM with various other middleware entities, such management agents (e.g. to monitor and manage the security status of the devices) or even more sophisticated AI management mechanisms, such as the one presented

in [136]. This entity is called Device Operator (DOP) and it is used to transmit and enforce the decisions of the management agent (e.g. a Security Agent) to the DPWS devices of uSPBM, and to allow the transmission of information (such as the current operating status) from said devices to the management agent. An uSPBM DPWS device deployed as an OSGi bundle and discovered via a DPWS client on the network can be seen in Figure 27.

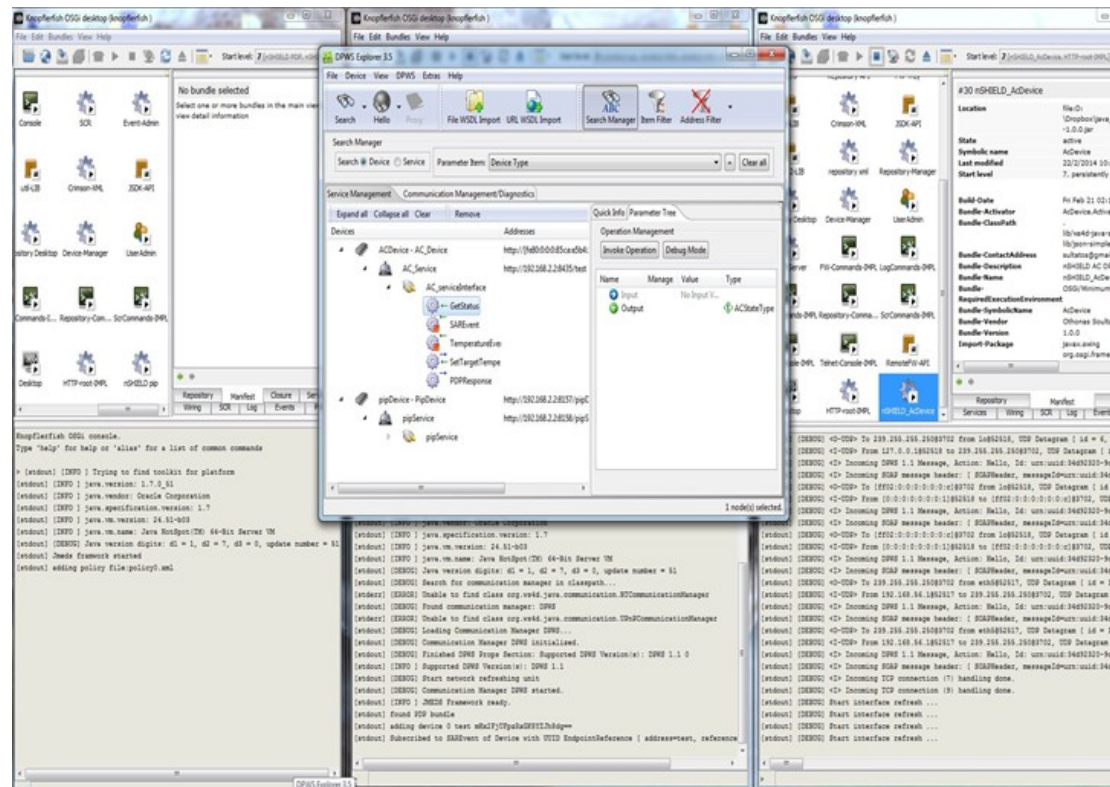


FIGURE 27. AN USPBM DEVICE DEPLOYED OVER OSGI AND DISCOVERED ON LOCAL NETWORK

The communication of the DOP with the rest of the middleware services and the management agent can take place via OSGi, namely over Knopflerfish, in cases where they'll be deployed on the same system. In case of remote deployment, the communication can be realized via DPWS, as is the case with the rest of the uSPBM entities. The DOP features the following operations:

- **St\_Event:** A WS-Eventing operation. All of the uSPBM end devices, upon initialization, search for the Device Operator and subscribe to St\_Event. As soon as a management agent issues a command that dictates a certain change in the state of uSPBM devices (e.g. to increase the length of their encryption keys, in order to raise the security level), the Device Operator fires an event, transmitting that information to the target end devices.
- **Update\_Operation:** As soon as each device performs a change in its operating (because of a management agent decision or otherwise), it invokes the Update\_Operation to let the DOP know of this change. The DOP can then inform the management agent of the new status of uSPBM's devices.



The "Device Operator" (DOp) is, therefore, responsible for setting and updating the operational status (e.g. the use of encryption or not) throughout the uSPBM framework. This functionality is depicted in Figure 28.

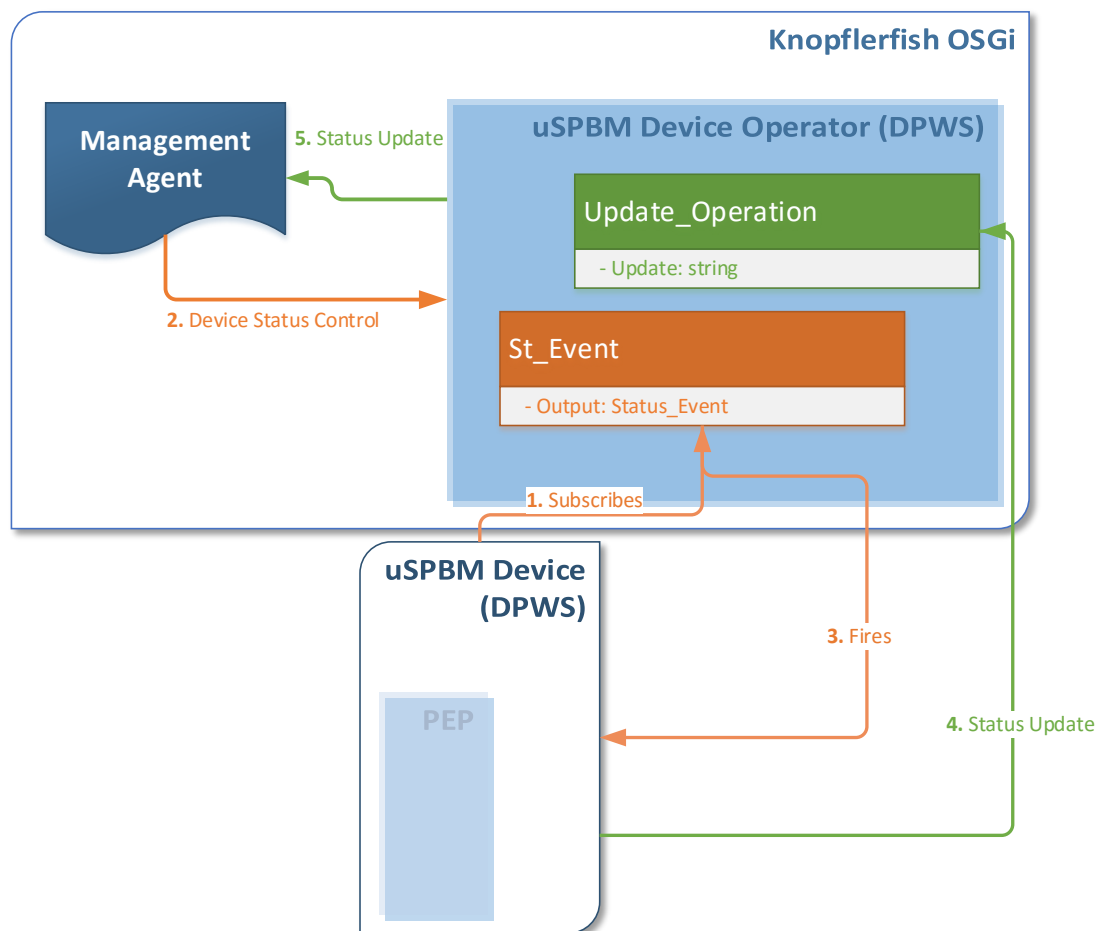


FIGURE 28. IMPLEMENTATION OF THE USPBM DEVICE OPERATOR

To briefly describe a simplified scenario:

- An uSPBM node (let's assume it's a smart camera) will feature the necessary Policy Enforcement Point (PEP) to make sure only authorized entities (i.e. those allowed by the current policy set) will be able to access its resources (rotate camera, turn it on/off etc.).
- In order to check said access requests, it communicates, when necessary, with the Policy Decision Point, which in turn communicates with the Policy Information Point to retrieve pertinent policies.
- Let us assume that the security of the policy messages exchanged between the uSPBM entities support 3 different security levels (e.g. no encryption, 256-bit AES encryption and 512-bit AES encryption).
- These mechanisms are enforced/controlled by the Device Operator bundle, which is running along with the rest of the middleware services (the backend PC running the Management Agent etc.).

- The uSPBM framework initializes with a pre-set security level. When the Management Agent (or any other control entity) decides that a higher/lower security level is needed, it relays this request to the "Device Operator" which in turns makes sure it's implemented on all of the uSPBM entities.

This simplified scenario can be seen in Figure 29.

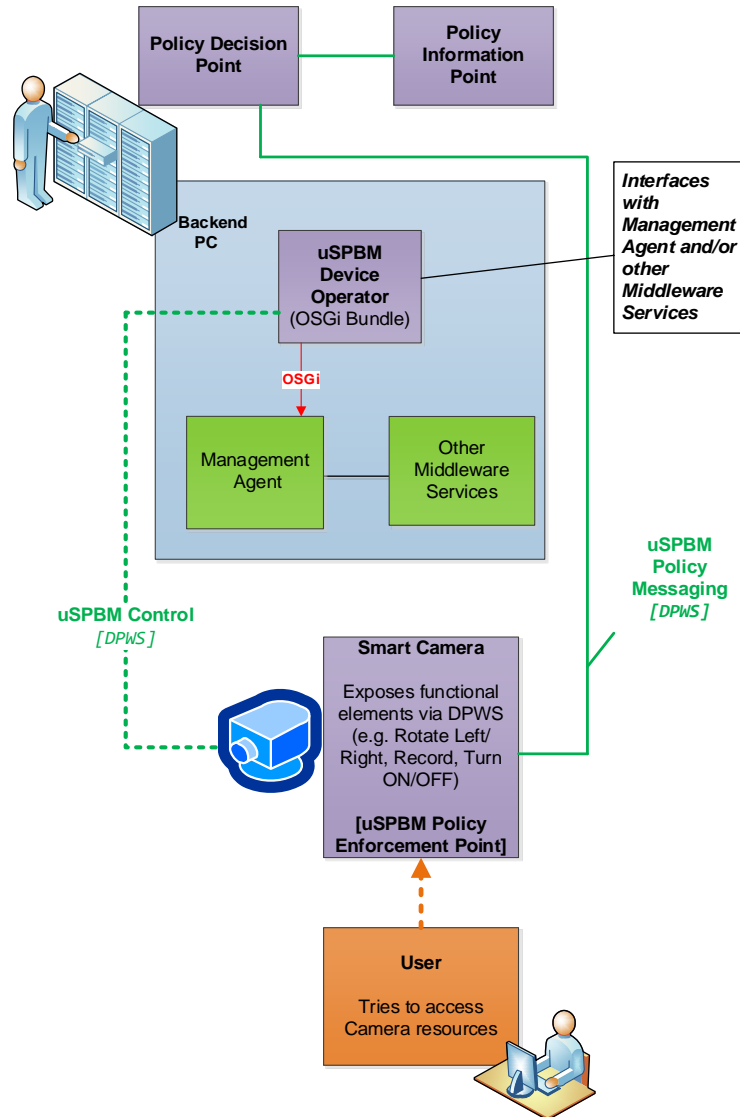
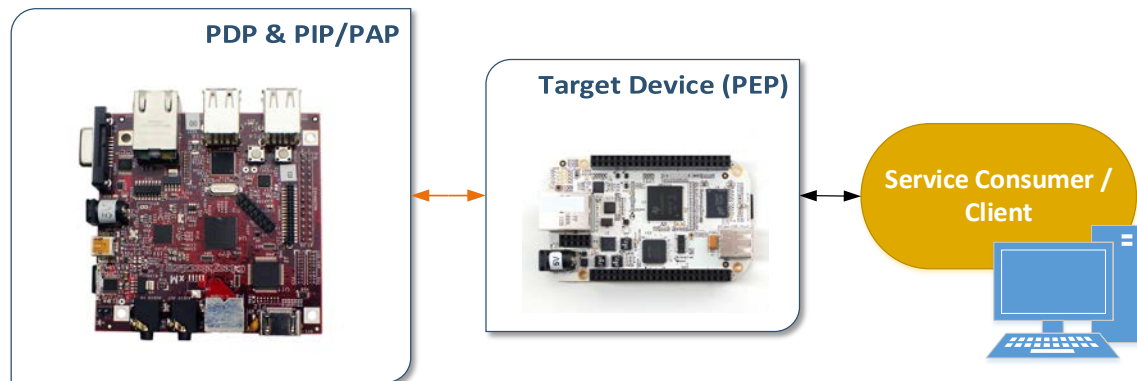


FIGURE 29. SIMPLE SCENARIO DEMONSTRATING DEVICE OPERATOR'S FUNCTIONALITY

## 4.5 PERFORMANCE EVALUATION OF CORE ENTITIES

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. This evaluation focuses on the main entities (i.e. the core authorization mechanisms) of the uSPBM framework. A more detailed evaluation, including other features such as management, authentication and cross-domain communications, can be found in later chapters, pertaining to specific use cases.

To evaluate the core components, the PEP-equipped DPWS devices were deployed on Beaglebone embedded 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 power platform, featuring a 1GHz ARM Cortex-A8 processor (throttled to run at 600MHz during testing), 512MB of RAM and a minimal Linux-based operating. The test-bed also featured a client application developed to query the uSPBM-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 bundles, which is an open source service platform following the OSGi specification. The uSPBM 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. The test setup appears in Figure 30.

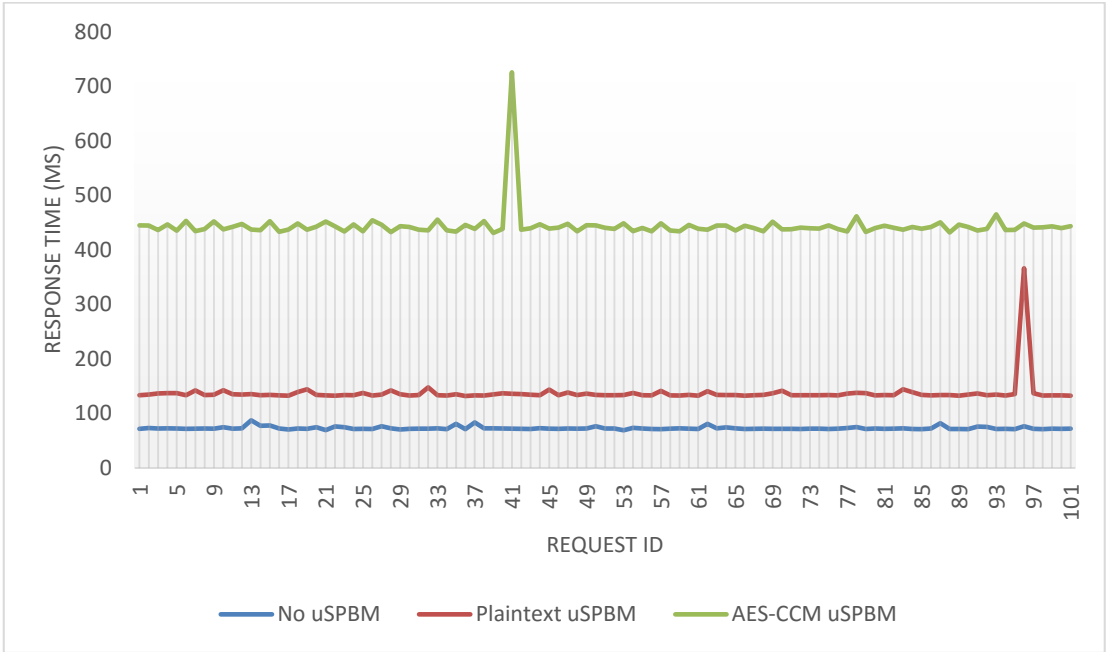


**FIGURE 30. PROOF-OF-CONCEPT TESTBED SETUP**

For the evaluation process, two functional operations were added to the test DPWS device deployed on the Beaglebone. So, a `GetStatusuSPBM` operation and a `GetStatus` operation were also implemented on the device, in addition to the PEP-related operations appearing in Figure 20. 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 uSPBM framework, as any extra delay when invoking the `GetStatusuSPBM` operation can be attributed to the access control 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).

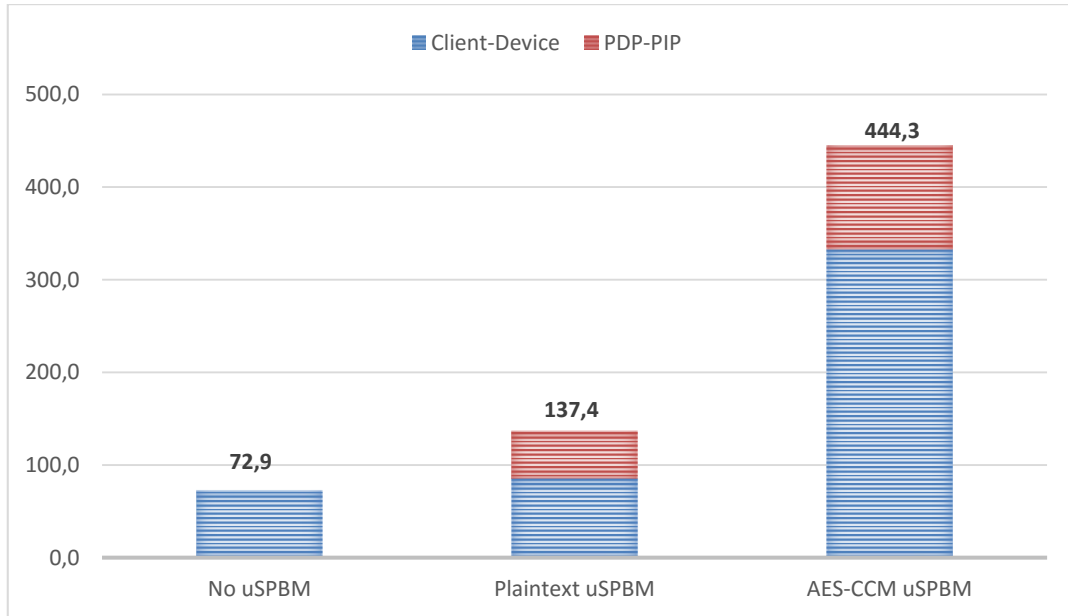
A total of 100 consecutive requests were issued from the client application to the DPWS device residing on the Beaglebone. The response time recorded by the test client trying to access the device's resources appear in Figure 31. "No uSPBM" refers to the invocation of the `GetStatus` operation, while the "Plaintext uSPBM" & "AES-CCM uSPBM" columns refer to the invocation of the `GetStatusuSPBM` operation, without and with AES/CCM encryption of the uSPBM 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.



**FIGURE 31. CLIENT-SIDE RESPONSE TIME (MS)**

A breakdown of the response times (averaged over 100 requests) can be seen in Figure 32, 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 7.

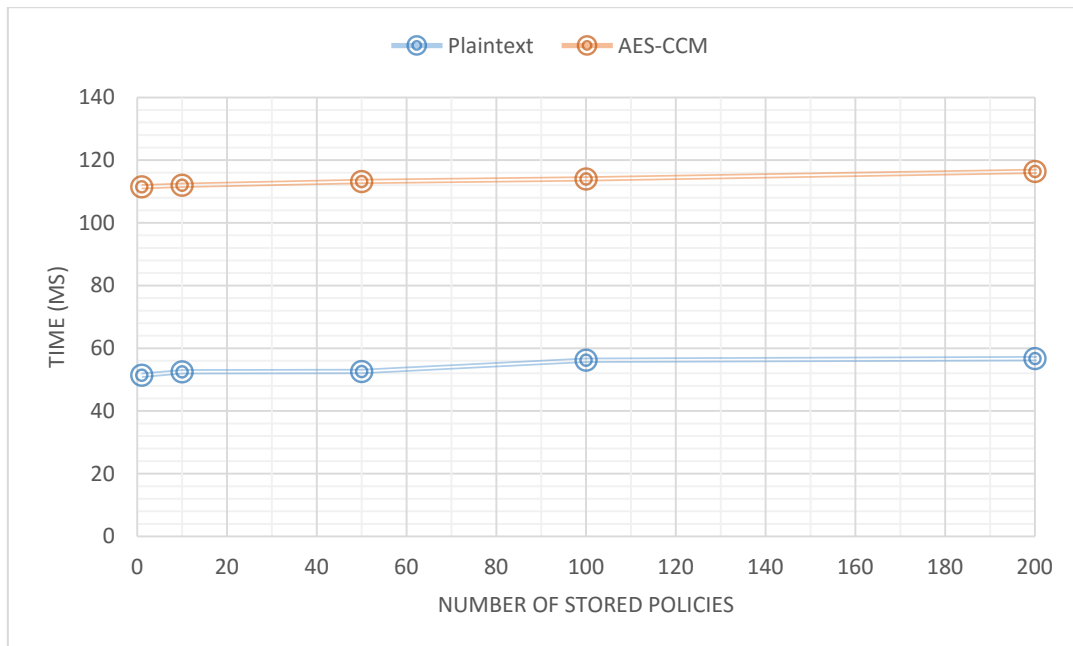


**FIGURE 32. RESPONSE TIME (MS) BREAKDOWN, AVERAGED OVER 100 REQUESTS**

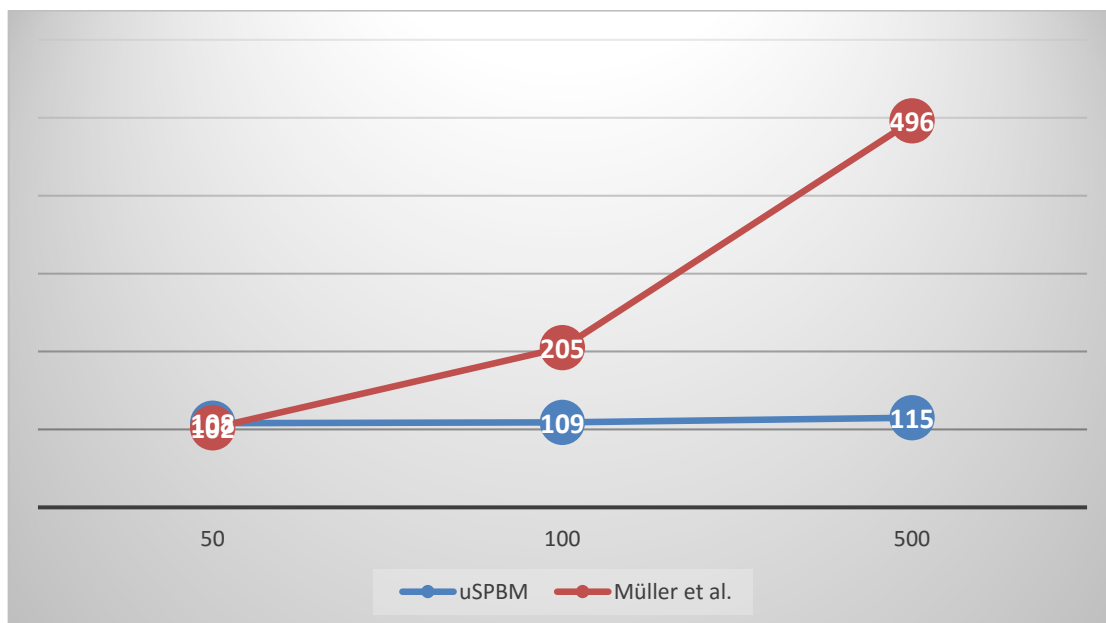
**TABLE 7. RESOURCE CONSUMPTION ON USPBM-PROTECTED DPWS DEVICES DURING BENCHMARKS**

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

Review of related work indicated that the number of stored policies can significantly affect the performance of the access control system, to the point where the response time overhead can become prohibitive in certain application scenarios [57]. 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. Figure 33 depicts the PDP-PIP communication, focusing on the time the PDP has to wait before it receives pertinent policies from the PIP, in scenarios where 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. The improvement over the approach of Muller et al. can be seen in Figure 34.

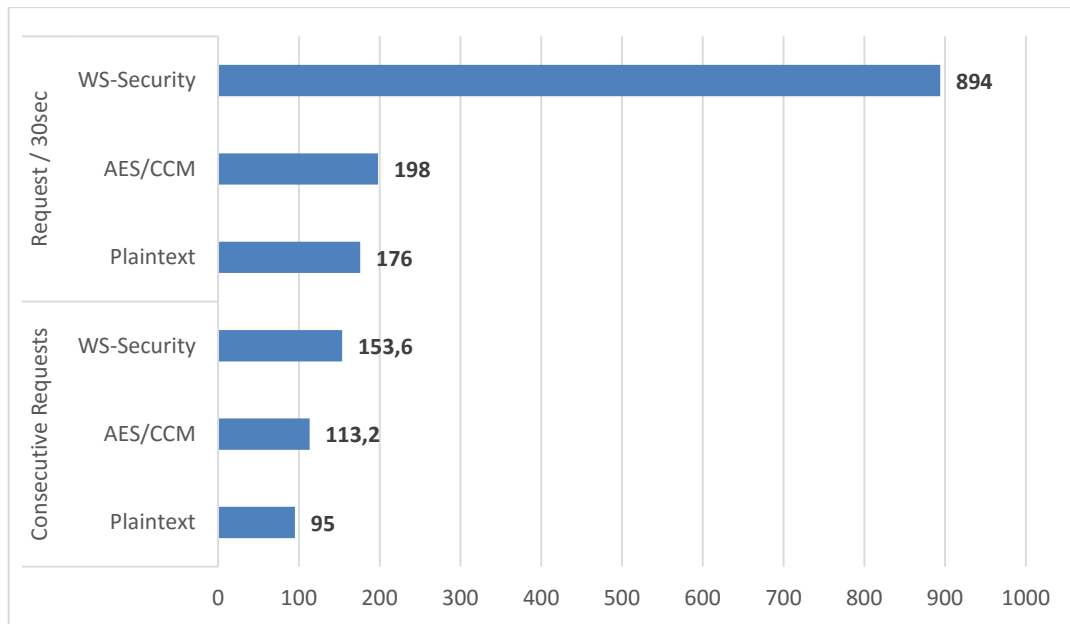


**FIGURE 33. PDP PROCESSING TIME. AVERAGE RESPONSE TIME (MS) DEPENDING ON NUMBER OF STORED POLICIES.**



**FIGURE 34. PDP PROCESSING TIME VS. NUMBER OF STORED POLICIES (IN MS). COMPARISON TO RELATED WORK.**

Tests were also conducted to determine the difference between the supported security mechanisms, namely the WS-Security mechanisms and the symmetric ones of AES/CCM. To this end, 50 policies were stored on the PIP and then 50 requests were sent from the client to the PDP. For this test, the PDP and PIP/PAP were all deployed on a common Beagleboard xM platform, with a client application on desktop PC. The requests were sent in two different forms, to test different use cases: consecutively (as in the previous tests) and once every 30 seconds. During these tests, the client-side response time was monitored. The results appear in Figure 35.



**FIGURE 35. THE EFFECT OF THE DIFFERENT SECURITY MECHANISMS UNDER TWO USE CASES (TIMES IN MS, AVERAGED OVER 50 REQUESTS).**

As is evident from these results, and as expected, there is a significant difference between WS-Security and AES/CCM modes. The differences are exacerbated in the non-consecutive requests, as there is no caching and, moreover, the socket between the two entities is closed. This is especially burdensome for the WS-Security asymmetric mechanisms, as the computationally intensive processes (handshake etc.) have to be repeated; thus the very significant performance degradation.

## 5. APPLICATIONS & EXTENSIONS

Smart device adoption is taking off in various domains, leading to the improvement of existing and introduction of new business sectors, all of which constitute key pieces of the puzzle that is the Internet of Things (IoT). This chapter presents such use cases of uSPBM, including its deployment over heterogeneous platforms and introducing subsets of the framework's entities that significantly enhance its capabilities and applicability to the various domains examined.

### 5.1 BODY SENSOR NETWORKS

Sensor nodes and actuators are becoming ubiquitous and research efforts focus on addressing the various issues stemming from resources constraints and other intrinsic characteristics typically associated with such devices and their applications. In the case of wearable nodes, and especially in the context of e-Health applications, the security issues are exacerbated by the direct interaction with the human body and the associated safety and privacy concerns.

The SOA approach of uSPBM constitutes an attractive solution for many types of networks, including those that consist of nodes with limited capabilities. Such a network is a body sensor network (BSN [137]) which comprises of a number of low-power implanted, wearable (on-body) or in close distance wireless sensors and actuators. The environmental and physiological sensors of a BSN provide vital information to medical staff, who can remotely monitor and possibly control user's medical treatment. For such an application there are many security requirements that need to be satisfied [138], including secure transmission of sensitive medical data to (remote) medical staff, unaltered instructions that reach patient's actuators, robust entity authentication and access control mechanisms.

The architecture presented in this section is a variation of the uSPBM mechanisms, simplified and adapted to the environment of a BSN, so that it can run on resource-constrained devices, thus facilitating secure and authorized access to BSN resources and services. The proposed scheme specifically considers the very limited resources of so-called nano nodes that are anticipated to be used in such an environment. A proof-of-concept implementation is developed and a preliminary performance evaluation is presented.

The proposed scenario facilitates the separate and scalable deployment of nodes based on the patient's needs. For example, a patient might initially have only a temperature sensor deployed and controlled, for instance, by his/her mobile phone, while subsequently a blood pressure sensor is also added where access to its resources and data is controlled by a proprietary device. Both these devices need to be accessible by authorized third parties, e.g. doctors, hospitals, national health system representatives, and therefore should adopt a unified set of policy rules, defined by the aforementioned stakeholders, including the patient, and maintained on a third system.

#### 5.1.1 MOTIVATION

In a typical BSN used for e-Health purposes, environmental and physiological sensors are deployed for gathering all the required information depending on medical staff's prescribed needs, such as blood pressure and body and room temperature. On top of that, actuators



controlled by authorized medical staff can also be deployed, such as an automatic insulin injection device used for remote treatment. Such sensitive actions, i.e. reading and issuing commands, need strict access control decisions before being authorized so that user's privacy and even safety are not jeopardized by unauthorized actions.

Consider the scenario where the patient has multiple let's say physiological sensors and actuators deployed, to monitor patient's 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 uSPBM fits perfectly as it provides the necessary fine-grained access control to resources. A doctor might deploy a new sensor on the patient's network without worrying about the current status of this network. The newly deployed sensor, let's say temperature, registers with patient's controlling device, e.g. mobile phone, and an access policy is defined for this sensor. The doctor or personnel in an emergency room, directly query the sensor for its readings, instead of having to find out which patient's device handles these requests and authorizes them (as it would be in a centralized scenario where the patient's mobile would handle all these requests on all sensors' behalf). The sensor, i.e. the Policy Enforcement Point in access control language, queries the Policy Decision Point, whether access should be granted or not. The PDP authenticates the requestor and contacts the Policy Information Point to get all applicable policies which it uses to decide upon granting or denying access to the requested resource.

The types of nodes, in terms of computing capabilities, which can be found in a BSN include the following:

- **Power Devices:** nodes with medium to high performance in terms of computing power and no particular resources restrictions. Although these nodes are typically used as sink nodes and/or gateways to other networks they can also have their own on-board sensors for collecting additional information. Example of a power node is a mobile phone, a laptop or a dedicated sink node.
- **Sensor/Nano Devices:** small devices with limited capabilities and resources, such as computational power, memory, storage space and energy, which in the case of nano nodes might be very restricted. These are typically the on-body or implanted nodes found in a BSN.

Access control is very important for protecting the sensitive resources of a BSN, which can affect human lives. Among the requirements that have to be satisfied are the following [138], [139]:

- **Data confidentiality:** Access to medical data should only be allowed to authorized parties, such as medical staff. Note that unauthorized disclosure of medical data while in transit is also a protection requirement.
- **Message authentication:** Commands issued to actuators must be authenticated to avoid unauthorized execution.

- **Availability:** Data must remain available to authorized entities, such as medical staff, while access to them must not be denied due to wrong decisions.

A central device may have the responsibility to collect data, manipulate, and add some logic during their processing can control the deployed sensors. Such a device can be a mobile phone or a dedicated sink node. If a value exceeds a threshold an alert can be sent to the user or the medical staff. In this case the central device can do all access control decisions on the nodes' behalf. The information is routed through a user's gateway, such as a mobile phone or a laptop and reaches the authorized medical staff.

In another scenario, access to these medical data is offered directly by the nodes as a service, assuming that the corresponding nodes have the capacity to accommodate such functionality. In such a scenario, the service requester, which can typically be any entity that can reach this node, interacts with the service provider, i.e. the node, to get the required medical data. Upon the request reaching user's environment, through a user's gateway, a question arises on whether access to the resource should be granted or not. Although this scenario provides more flexibility in terms of functionality, the deployed sensors are integrated on a low-power, resource-constrained devices which do not have the computational power to handle such requests. Moreover, these sensing devices are often expendable in e-Health applications, prohibiting the permanent deployment of complex (and thus expensive) platforms. Therefore, a solution has to be provided regarding the way that the node controls access to its resources. The framework provided in this thesis addresses this issue.

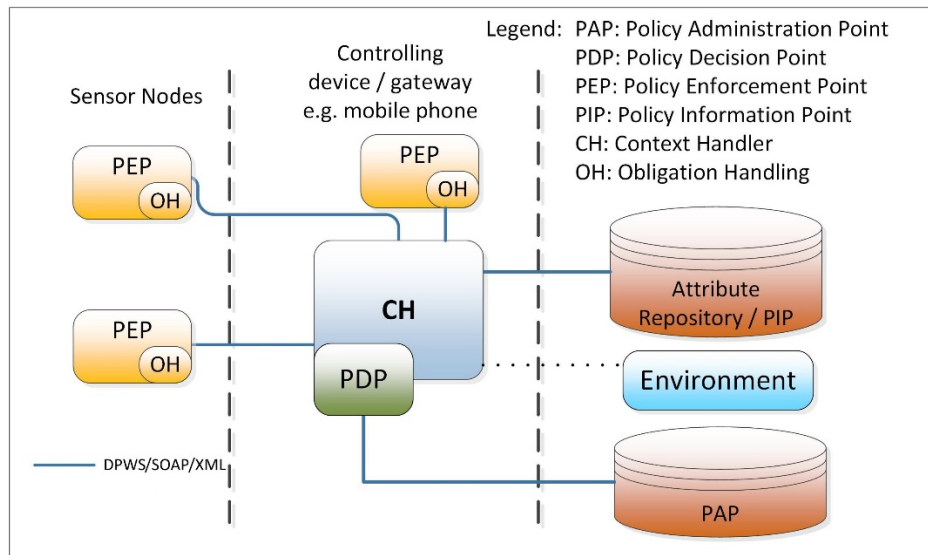
### 5.1.2 PROPOSED ARCHITECTURE

The framework's adaptation in this use case is based on the standardised XACML architecture [22], [84] to provide a cross-platform solution that can typically be deployed in various types of embedded systems while satisfying interoperability, an important requirement for next-generation pervasive computing devices. Thus, the main entities include:

- Policy Enforcement Point (PEP): It performs access control, by making decision requests and enforcing authorization decisions.
- Policy Administration Point (PAP): Creates and manages policies or policy sets.
- Policy Decision Point (PDP): It evaluates requests against applicable policies and renders an authorization decision.
- Policy Information Point (PIP): It acts as a source of attribute values.
- Context Handler: It orchestrates the communications among the stakeholders, converts, if necessary, messages between their native forms and the XACML canonical form, and collects all necessary information for the PDP.
- Environment: Provides additional information independent of a particular subject, resource or action. Refers to features related to the environment and can affect a PDP's decision, e.g. accident scene or hospital emergencies.

In the proposed architecture the sensor nodes and actuators, which have direct access to resources, expose their functional elements to the PEP. These nodes are micro/nano nodes and are not expected to have the capacity to accommodate additional functionality. The context handler and the PDP are likely to run on power nodes together with PEP functionality,

only if the node has resources to share through corresponding services. Note that there can be more than one power nodes in the user's network, e.g. a mobile phone, tablet and/or a dedicated device deployed with specific sensors. In this case only one of them will act as a PDP. The functional model of the proposed framework is depicted in Figure 36.



**FIGURE 36. FUNCTIONAL MODEL OF POLICY-BASED ACCESS CONTROL FOR HEALTHCARE ENVIRONMENTS**

All the above XACML components can run on a single system or, in a more distributed approach, on different systems based on their distinctive capabilities. The latter is the model that fits the scenario described above, i.e. that of a BSN comprising of a number of nodes.

#### 5.1.2.1 TYPICAL APPLICATION

As an example, consider the case of a patient who visits a doctor. A number of micro/nano nodes are present on or near the patient's body and each of these nodes hosts one or more services exposing the various features of their sensors and actuators. As soon as the available devices and their corresponding services (e.g. temperature or heart-rate service) are discovered, the doctor can request access to the node that is of particular interest, e.g. in order to extract the latest values from the temperature sensor attached to it. The doctor's request is intercepted by the node's PEP module which then forwards the request to the PDP, the latter running on a user's trusted device. The PDP has to consider all applicable policies from the PAP, enriched by any relevant information residing on the PIP, while additional ones might be added in real time regarding the specific access. For instance, a question can be displayed on the user's mobile phone regarding this access request giving the user the option to explicitly grant or deny access. Once all the required information has been collected, the PDP issues a decision which is sent back to the node's PEP. Based on that decision the PEP may or may not allow the doctor to access said node's data of interest. It should be noted that, on top of the decision taken on the request, the PDP might set one or more obligations for the PEP. An obligation is additional restrictions that should be taken into account when enforcing a decision, like the requirement to log any permitted access or to inform for unauthorized attempts. Moreover, prior to this communication the PAP should have set all applicable policies and policy sets for all targets in the network. These policies are made available to PDP for subsequent requests evaluations.

#### 5.1.2.2 *POLICY CONSIDERATIONS*

While in most situations full control on the policy administration should be granted to the patient there are many situations where this should be overridden, such as immediately after an accident, or if the patient suffers from a mental illness, where additional entities must gain access to the medical data. Therefore, among the healthcare access control requirements is the need for both medical staff and patients to be able to define their own policy requirements and restrictions [138] based on the need for treatment and their right to have full control over their health data and records respectively. On top of that, additional entities may request access to these resources, such as an insurance company which might want to verify the patient's medical condition upon request for expenses. All these stakeholders have the right and need to define their own rules and policies regarding access to medical data. In this case a Rule and/or Policy – Combining Algorithm has to be used to come to a conclusion regarding granting or denying access to the requester. The involvement of multiple stakeholders who want to define their own policies raises the need for maintaining multiple repositories (note that these are not necessarily under user's administration).

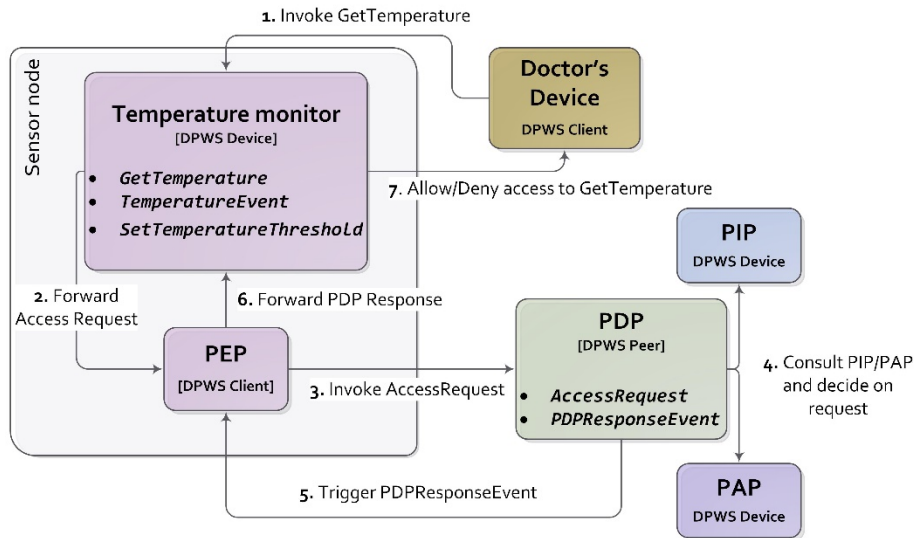
The delegation mechanism defined in XACML 3.0 can be a useful tool to a BSN. Delegation allows the system owner or administrator to delegate some rights regarding the administration of policies, such as to create additional policies. To support this functionality, policies that will be used to control policies also have to be set by the system owner and/or administrator which in our case might be the patient herself.

#### 5.1.3 IMPLEMENTATION APPROACH

For the proposed scheme to be operational each device's functional elements must be represented by an appropriate DPWS entity and its corresponding operations. Assuming a simple temperature sensor, for instance, a node is programmed as a DPWS device which hosts a temperature service featuring various operations:

- A "*GetTemperature*" operation which, when invoked, will return the patient's current temperature.
- A more complex "*TemperatureEvent*" operation which, by exploiting the WS-Eventing mechanism [95], allows a client device (e.g. doctor's device) to subscribe to the service and get temperature updates at set intervals as well as event notification messages when the temperature exceeds a certain threshold.
- An additional "*SetTemperatureThreshold*" operation which, when invoked, allows setting/updating the abovementioned warning threshold.

Similarly, the XACML-related elements of each node must be represented as DPWS devices, clients or peers (i.e. devices that function both as clients and servers). The approach adopted includes a DPWS client on the temperature sensor node described above. This client is then used to discover and use the PDP service implemented on a control/gateway node. The process followed when a user tries to access a sensor's functional elements (e.g. the temperature reading) is depicted in Figure 37.



**FIGURE 37. USPB BSN IMPLEMENTATION USING DPWS**

In more detail, assuming a doctor tries to access the temperature sensor's features (Step 1), the request is automatically forwarded to the device's PEP (Step 2). The PEP can then invoke the "AccessRequest" operation on the control node's PDP service (Step3), sending a properly formulated access request to the PDP. When the PDP is done evaluating (Step 4) the request based on subject's attributes and policy rules, it can, in turn, trigger its "PDPResponseEvent" (to which the sensor's PEP client subscribes during initialization), returning the authorization decision. This decision is then conveyed to the functional operation of the device, thus granting or denying access to the "GetTemperature" operation the doctor tried to invoke.

The PDP to PAP and PIP entities' functionality can, equivalently, be developed as DPWS devices and clients, exploiting the integrated discovery and subscription mechanisms, thus bypassing the need to use other protocols (e.g. LDAP).

#### 5.1.4 PROOF OF CONCEPT

A proof of concept implementation of the USPB scheme presented in this work was developed using Sun's XACML as a basis for the policy and access control mechanisms. The WS4D-JMEDS API was used for the creation of the necessary DPWS devices. The application developed for the BSN included the functional operations of the BSN as well as the Policy Enforcement Point that had to be deployed on each node. The implementation consists of the following modules:

- An application that runs on the sensors and which implements the access to the functional elements of the sensor (e.g. temperature reading) as well as the communication with the sink node. A security mechanism was also developed, based on the AES algorithm in CBC mode [140] and pre-shared secret keys, to guarantee that only the legitimate sink node/bridge can access the sensors. When connected to the bridge, sensors ignore all other connection requests. Moreover the security mechanism protects the messages from eavesdropping on the sensors'-sink node communications.

- A sink node application that bridges the BSN (Figure 38), which in this case operates over 802.15.4, to the standard network infrastructure. This application has to be deployed on a device equipped with dual 802.15.4 & Ethernet/wireless Ethernet functionality.
- The DPWS Provider module (Figure 39) which discovers available sensors (via the sink node), probes said sensors to discover their functionality and then maps this functionality to a corresponding DPWS device. As a result of this procedure, a DPWS device is created for each of the discovered sensors. This device also includes the necessary operations to realize the PEP functionality, as well as the conversion of all low level messages transmitted to and from the sensors to a DPWS compatible form. This procedure is transparent from the perspective of the client (BSN user). Therefore, a client can use DPWS discovery (Figure 40) to identify available sensors and their operations (functional elements), invoke the desired operation or the chosen sensor and, if allowed by the PEP (which will first check if the request is authorized), get the result of said invocation. The communication of the PEP(s) deployed by the invoker to the infrastructure entity which authorizes the requests (i.e. the PDP) must also be protected, as malicious tampering of the policy messages exchanged by the USPBM entities can compromise the access control efforts. To this end, a security mechanism based on the AES/CCM [134] authenticated encryption algorithm was implemented, using 128bit keys. Deployment of this mechanism guarantees that the USPBM-related messages exchanged between PEP and PDP (when the former seeks the authorization status of a specific client's request), are fully protected in terms of confidentiality, integrity and authenticity.

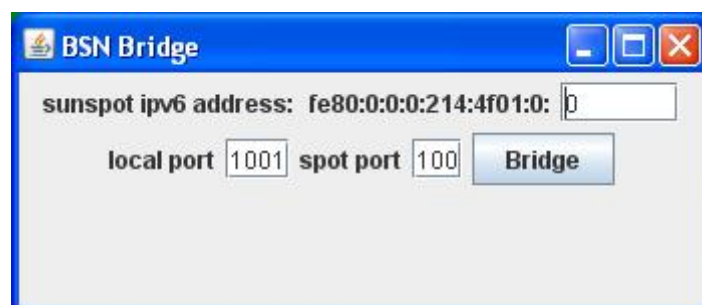


FIGURE 38. THE BSN BRIDGE

BSN DPWS Provider

Provider's IP Address to be binded 192.168.2.4

Spot Bridge IP Address (where spots are bridged) 192.168.2.6

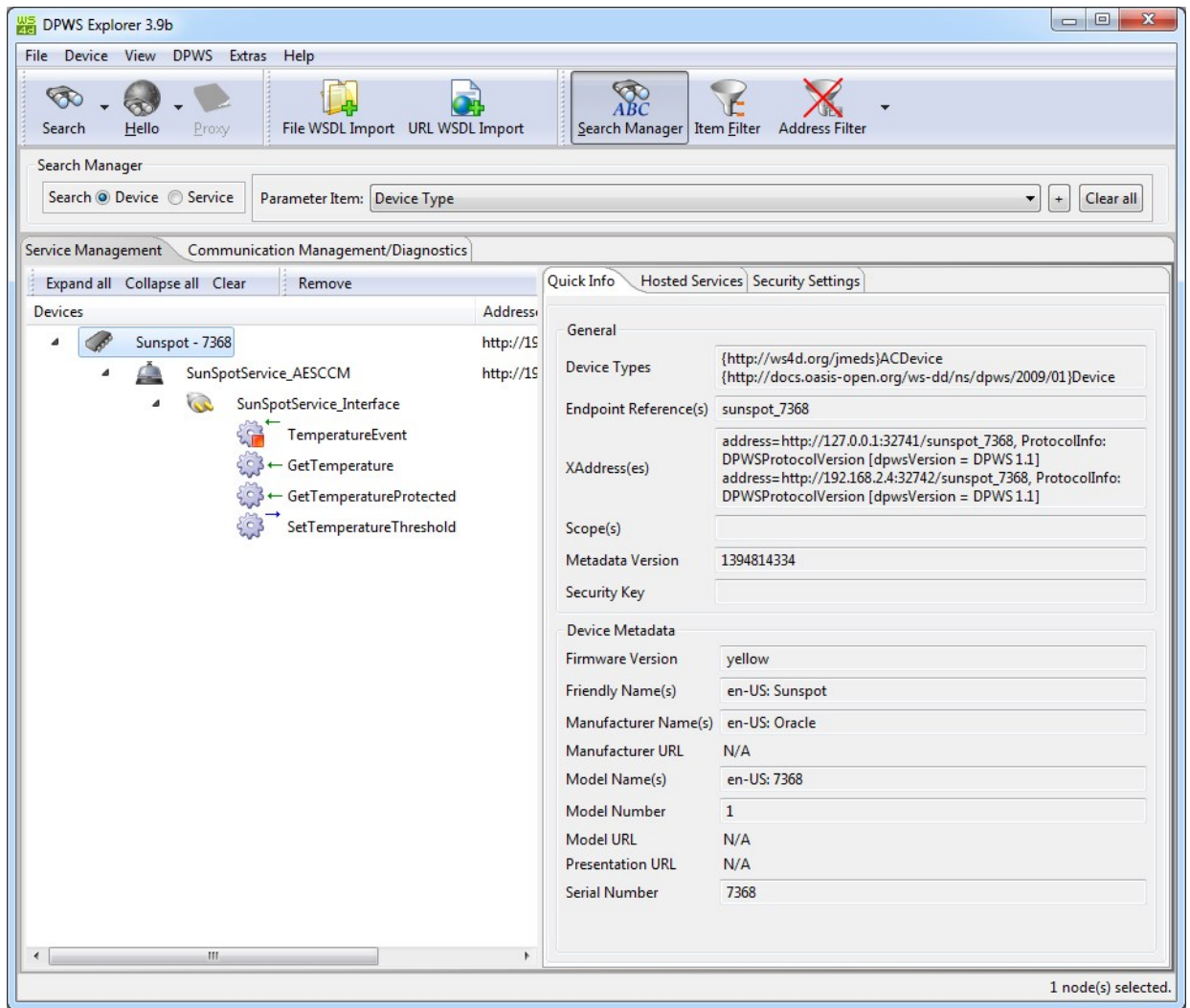
Mode of Operation ☐ Plain ☒ AES CCM

How Many Spots? 3

Port	Key	
2001	646ef5a30154e9f6deb	Stop
2002	e489b44e954ec0b9bca	Stop
2003	d93369ce542e8f2322d	Stop

Save&Exit 0 StartProfiling StopProf Start

FIGURE 39. THE BSN DPWS PROVIDER

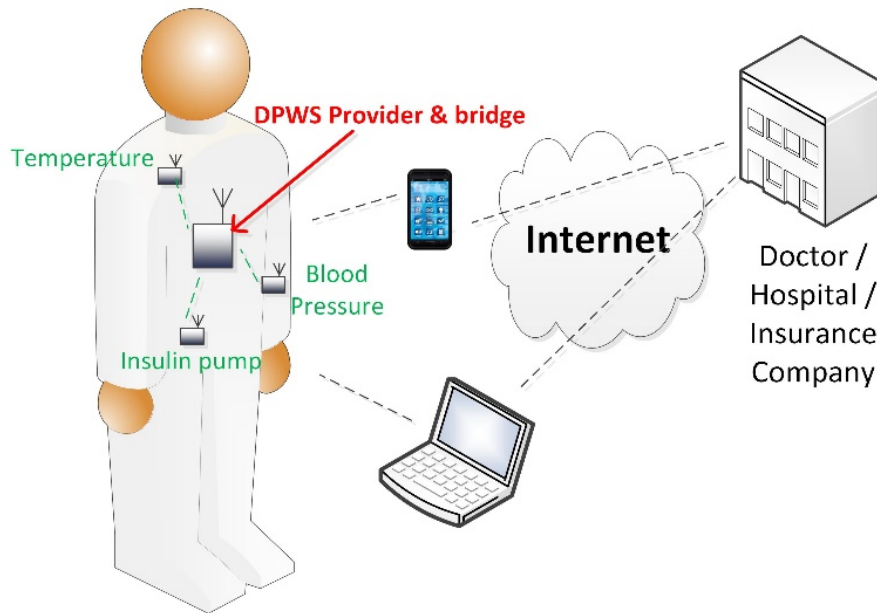


**FIGURE 40. DISCOVERING THE SENSORS AND THEIR HOSTED SERVICES ON THE NETWORK**

#### 5.1.4.1 PERFORMANCE EVALUATION

The performance of the proof of concept implementation was evaluated on a test-bed featuring a SunSPOT mote [108] running the sensing application. Another SunSPOT mote was connected to a personal computer acting as a sink node. The DPWS Provider application was deployed on the same computer system. In a real-world application the bridge/DPWS Provider functionality could be deployed on any smart device with dual 802.15.4 & Ethernet/wireless Ethernet connectivity, even a small embedded or wearable device, as depicted in Figure 41. The PDP/PIP/PAP application was running on a separate computer system which also stored the policy files. This system also featured a client application developed to query the sensors for benchmarking purposes. SunSPOTs communicate via the 802.15.4 radio, while the personal computers communicated via wired Ethernet.

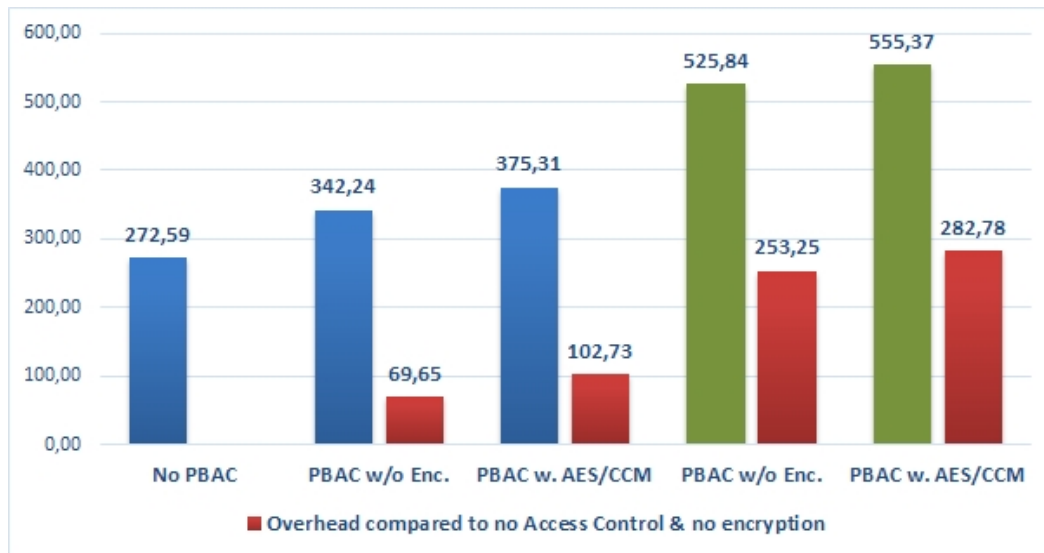




**FIGURE 41. PROPOSED DEPLOYMENT OF THE PROOF-OF-CONCEPT USPBM BSN APPLICATION**

A total of 50 consecutive requests were issued from the client application to the sensor. In order to evaluate the delay imposed by the proposed scheme, the sensor featured both a PEP-protected operation (*GetTemperature*) that the test client was allowed to invoke by the current policy set and an unprotected operation (*GetTemperatureUnprotected*) which could be invoked immediately (without going through the policy enforcement point for authorization). Aiming to also weigh the impact of the security mechanisms, the assessment included scenarios with and without encryption on both the SunSPOT-Provider link (plaintext vs. AES-CBC) and the PEP-to-PDP link (plaintext vs. AES/CCM).

The average response time for 50 requests, including the overhead when considering a totally unprotected (access control- and security-wise) operation as baseline, can be seen in Figure 42.



**FIGURE 42. AVERAGE RESPONSE TIME (IN MS) FOR 50 REQUESTS. COLUMNS IN BLUE DEPICT THE SCENARIO WHERE THERE IS NO SECURITY BETWEEN THE SENSOR AND THE PROVIDER, WHILE COLUMNS IN GREEN CORRESPOND TO THE SCENARIOS WHERE AES-CBC ENCRYPTION WAS USED TO PROTECT SAID LINK**

It should be noted that the bulk of the delay can be attributed to the communication between the SunSPOT and the Provider, as was evident from timing tests run concurrently on the client side and the Provider side. E.g. the results of such a test, run with AES-CBC protection on the SunSPOT messages and no protection on the PEP-PDP communication, indicated that out of the 527,3ms client-side delay (on average, for 50 requests) when invoking an unprotected (i.e. no uSPBM involved) operation, 449,45ms was the average time that the Provider had to wait until it got a reply from the SunSPOT. Therefore, the overhead of the DPWS communication between client and the Provider (i.e. the DPWS device that “mirrors” the sensor’s functionality) was 77,85ms.

Another interesting fact is that when changing the policy so that the invocation of the protected operation by our test client is denied, the response time is negligible, as the request is rejected by the PEP and is never forwarded to the sensor. In a test run of 50 such unauthorized requests, the average response time of the DPWS device was just 8,39ms.

### 5.1.5 SUMMARY

In this section we proposed an adaptation to uSPBM framework for controlling access in BSNs comprising of nodes with limited resources. In the proposed scheme emphasis was given on the limited resources of some nodes found in such networks. This assumption and the relevant provisions allow for a more flexible scheme and one which can be deployed in heterogeneous systems, assisting in its integration with the Internet of Things. The results of these efforts included a proof-of-concept implementation, which is presented in this work along with an initial performance assessment.

## 5.2 AUTHENTICATED ACCESS TO LLN-CONNECTED RESOURCES

This section presents the uSPBM extension that allows authorized entities to access the services provided by resource-limited nodes. The scheme provides flexibility in terms of the authentication mechanism used, that is to say that the service requester can be authenticated

using e.g. username/password, certificate, or other authentication methods. Among the main concerns of the proposed architecture are the nodes' protection from unjustifiable use of their resources and the need to be able to control access through a well-established set of policy rules that can change and adapt to new environmental parameters.

The work builds upon the XACML model for policy based access control infrastructures, proposing certain modifications to satisfy requirements stemming from the limited resources of nodes, and the adoption of lightweight SOA mechanisms, through the use of the DPWS, for entity interactions. Although mainly a framework, the main components of the proposed architecture have been implemented and results are provided here as a proof of concept.

The proposed approach allows leveraging work already carried out on XACML policy definitions, but also Web Services. With regard to the former, the "Cross-Enterprise Security and Privacy Authorization Profile of XACML v2.0 for Healthcare" [141] constitutes important background work, compatible and in-line with the scheme proposed in this thesis. This OASIS profile specifies the use of XACML to promote interoperability within the healthcare community by providing common semantics and vocabularies for interoperable policy request/response, policy lifecycle, and policy enforcement.

The benefits of adopting a SOA-based approach come in the form of increased usability and interoperability. While typical XACML deployments require the setup of complex infrastructures to enable entities' interaction and policy retrieval (e.g. via the Lightweight Directory Access Protocol, LDAP [28]), the proposed framework leverages the benefits of DPWS. This allows the deployment of devices aligned with the Web Services technologies, thus facilitating interoperability among services provided by resource-constrained devices, facilitating seamless discovery and interactions among entities, and allowing the deployment of the framework's entities to any platform, anywhere on the hospital or home network, with minimal involvement on behalf of the user.

### 5.2.1 MOTIVATION

Before moving into the presentation of the proposed architecture, it would be good to demonstrate through specific scenarios, the incentives behind this work that have also formulated the requirements defined below. The proposed scheme addresses the main need to be able to remotely access data collected by sensors and control actuators deployed in a LLN. The architecture utilizes service oriented technology to be able to provide services to remote authorized parties where access restrictions are imposed through policy rules. This typically means that access is not necessarily restricted to entities of a closed system. Such an architecture fits perfectly to a Body Sensor Network (BSN) deployment [137], [142], which actually inspired this work, and which we use here to demonstrate the architecture's applicability and the way that policy based access control SOAs are envisaged.

Let's assume that a patient has multiple medical sensors and/or actuators deployed to monitor and/or control his/her medical condition. Sensors and actuators typically reside on nodes with very limited processing power and capabilities, namely nano nodes. These can communicate and register with a mobile device that the user has in possession, such as a mobile phone or tablet. An application running on this mobile device actively monitors

sensor's readings and, if necessary or appropriately instructed, forwards these data to authorized medical staff. Alternatively, the data could be given to medical staff not as a result of an alert, but as a response to a request issued by this staff.

Now consider the case where in the context of telemedicine or in case of an accident as well as for many other medical reasons, other people not previously registered with the user's application, need to gain access to those readings and actuators. For example, a patient is involved in an accident, which is reported to the emergency services, and some readings have to be taken to validate his medical conditions while emergency services are on their way to the accident scene.

In this case, the requester, i.e. doctor, emergency services staff, will request remote access to the readings of these sensors or even issue commands to the actuators. Without loss of generality, we can claim that in the eHealth environment, as with any other environments, services might need to be accessed occasionally, depending on the patient's health condition, and on a need to know basis. Therefore, several questions arise that have to be addressed in the proposed architecture, mostly related to patient's privacy and life protection.

- Who is eligible to access this information?
- How do we authenticate a user that has not been registered with the specific service in the past?
- How is the legitimacy of his/her request evaluated?
- Who and how is going to decide about this requester's privileges?

In our scenario we consider that the medical staff can look in a central repository for the types of services provided by the patient and can request access to them. This is checked against applicable policies that the user in conjunction with medical staff and/or national insurance and/or insurance company and/or applicable law have defined. If access is granted the request is forwarded to the patient's device and the requested information is disclosed or access to the actuator is permitted. As a result, the requester will be able to have sensor readings, e.g. patient's heart rate, and/or act remotely, e.g. inject an altered dose of insulin.

Access control is very important for protecting the sensitive resources of a BSN, which can affect human lives. Among the requirements that have to be satisfied are the following [138], [139]:

- Data confidentiality: Access to medical data should only be allowed for authorized parties, such as medical staff. Note that unauthorized disclosure of medical data while in transit is also a protection requirement.
- Message authentication: Commands issued to actuators must be authenticated to avoid unauthorized execution.
- Availability: Data must remain available to authorized entities, such as medical staff, while access to them must not be denied due to wrong decisions.

IP based networking in LLNs changes the way that participating nodes can be accessed and their respective services can be consumed. For instance, there is no need for a dedicated application server that will intervene between a node and a remote party that wants to access

the node's resources [143]. However, one of the problems that these nodes face in such a deployment, is that they have limited resources which do not suffice for the deployment of strong protection mechanisms. Without those mechanisms however, nodes are exposed to direct access from the Internet without having the capacity to handle unlimited requests. Therefore, several issues arise regarding the protection of nodes resources, which have to be addressed. The main aim is to protect the limited resources of a node that implements a service oriented architecture, to provide access to data and mechanisms that the node has under control.

Within this context, the proposed architecture is designed to satisfy the following requirements:

- Provide services using of Service Oriented Architecture technologies;
- Provide fine-grained access control to nodes' resources;
- Authenticate remote entities wishing to access protected nodes resources;
- Control access to nodes' resources through well-defined policies;
- Protect sensitive nodes from unauthorized access and unnecessary consumption of valuable resources including network and energy;
- Secure the channel between the participating nodes to provide message confidentiality, integrity and authentication;
- Comply with existing standards to satisfy interoperability among the participating entities, such as between the identity provider chosen by the requester and the service orchestrator, regarding the exchange of authentication messages, assertions or user metadata and attributes.

In the following section we describe the proposed architecture that satisfies the above.

### 5.2.2 PROPOSED ARCHITECTURE

The architecture proposed in this work is an enhanced policy based access control scheme that seeks to provide flexibility regarding the chosen authentication mechanism while satisfying the aforementioned requirements, typically imposed by nodes' resource limitations. For this purpose, certain modifications to the OASIS standardized policy-based access control scheme are proposed to accommodate these needs.

The scheme utilizes and seeks compliance with the following technologies:

- An XACML-based architecture consisting of the main components already mentioned (i.e. PEP, PDP, PAP & PIP).
- SAML 2.0 specification to protect, transport, and request XACML schema instances and other information needed by an XACML implementation [109]. Note that although SAML can be used to convey authorization decision statements, this functionality in SAML is intentionally restricted compared to the more flexible XACML solution, hence the adoption of XACML and the use of SAML for encapsulating XACML messages.

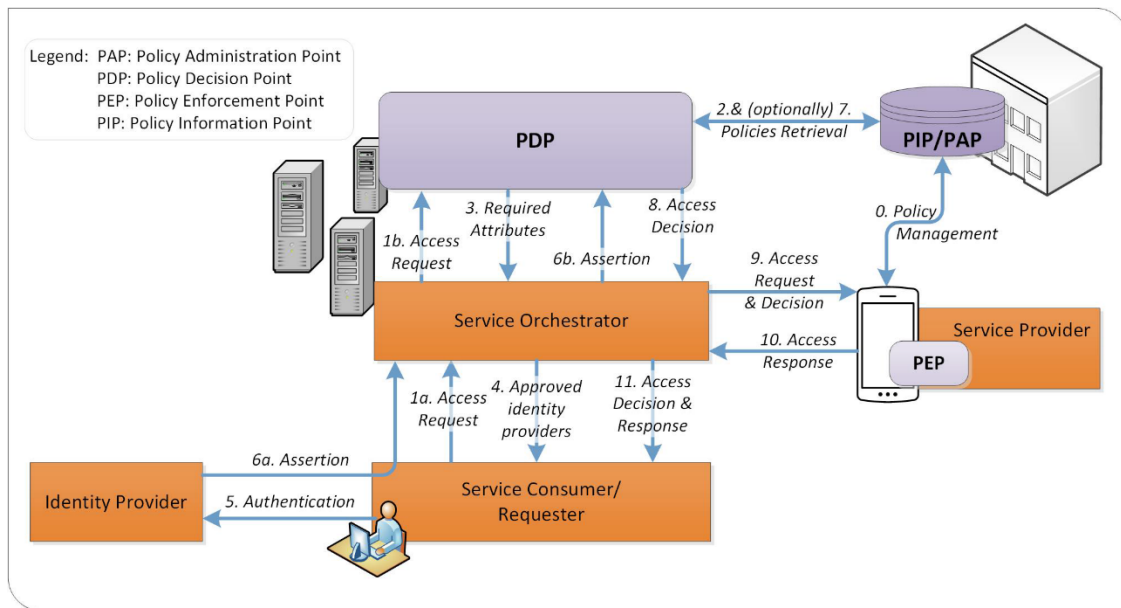
In the XACML data-flow model defined in the OASIS standard the PEP, via the context handler, is considered as the device that orchestrates the exchange of messages among the requester,

the PDP, the Attribute Authority and the Attribute Repository. According to the XACML specifications the PEP is considered as “part of a remote-access gateway, part of a Web server or part of an email user-agent, etc”. Therefore all initial requests, valid or not, are sent to the PEP which will act as a routing device between the requester and the back-end key entities that examine the requests and make decision based on policy rules and other parameters, such as the requester’s and/or resource’s attributes.

While this model is appropriate for typical application gateways, it cannot be considered as such for resource-constrained nodes that only have the capacity to accept requests from a limited number of clients. Beyond this threshold, valuable node resource consumption is not acceptable as it leads to battery drainage and service unavailability. In this context, resource-constrained devices have to participate in the decision making process only if absolutely necessary and only to authorized entities to save valuable resources. As such, they cannot assume the role of a PEP as this is defined in the XACML standard.

Moreover, the flow model currently defined by XACML, considers that the PIP has all the required attributes for the requester, and that the PDP gets all the information from the PIP, which might be queried twice for the required attributes, once from the PEP and once from the PDP. Use of specific PIP implies that services will only be provided to entities subscribed to the specific scheme, thus narrowing down flexibility. This is in contrast to a more flexible approach where services are offered to a broader group of users, subject to policy restrictions.

The proposed architecture is depicted in Figure 43. In this proposal we assume that nodes bearing sensor and actuators, expose their functionality as web services. This can either be done through the device that the node is attached to, e.g. a mobile device, or directly by the node, assuming that it is powerful enough to accommodate such functionality. All these nodes are part of a dispersed environment where there is not necessarily a single gateway or web server to assume the role of PEP as this is defined in the XACML standard. Besides that, the service owner might want to register these services with multiple servers. As a result, the PEP functionality cannot be assigned to a gateway but it should be on the device that exposes this functionality, e.g. the mobile device, a wearable node, etc. For a given PEP, one of these web servers is assumed to play the role of the orchestrator as described below.



**FIGURE 43. AUTHENTICATED ACCESS CONTROL.**

The core component of the proposed scheme is the Service Orchestrator (SO) which acts as a proxy for certain operations, such as relaying queries and messages exchanged among participating entities, yet not for handling the information the PEP exchanges with the requester.

Initially, the node, which assumes the role of a PEP, registers its services, defines the connection point to be the SO and sets the policy rules for its resources. This is accomplished once during the set-up phase. Following that, the data flow of the proposed architecture includes the following steps:

- A requester, who wants to access the service, formulates an appropriate request based on the advertised service rules, and sends it to the SO (step 1a). Note that this is in contrast to the XACML specifications which opted for sending the request directly to the PEP, introducing significant overhead that a limited-resources device cannot handle.
- The SO forwards the request to the PDP (step 1b) which, based on the requested target, fetches all applicable policies from the PAP (step 2) and informs the SO about the needed user attributes (step 3). As a result, the SO presents a list of approved Identity Providers (IdP) for the requester to authenticate (step 4).
- The requester chooses the appropriate IdP and the SO issues a (signed) authentication request (<AuthnRequest>) together with an attribute query (<AttributeQuery>) to the chosen IdP [109]. Upon successful authentication (step 5) the requester consents for the disclosure of certain attributes that the SO requires. Note that the IdP might be an entity that operates within the same environment as the SO. The actual authentication method used by the IdP is outside the scope of this work.
- The IdP formulates a proper assertion for the necessary attributes and sends it to the SO via the Requester (step 6a). As a result, the SO forwards the received assertion to the PDP (Step 6b [144]).

- The forwarded assertion allows the PDP to establish a security context by combining the supplied attributes with the applicable policy rules which the PDP obtained from the PAP (step 2). Note that additional policy rules, might be obtained at this point (step 7), based on the requester's attributes. The typical XACML decision making process can take place during this step.
- The access decision is sent to the SO (step 8). If the decision is to grant access, a signed or MAC-protected ticket is forwarded to the PEP together with details about the request (step 9). This is the first time that the node is contacted, and is only performed by an authorized party, hence not exposed to the outside world. If access is denied the decision is simply forwarded to the Requester. The Service Provider might also be informed on that based on appropriate pre-configurations.
- Now the PEP can respond to the service request through the SO (step 10). The SO can in turn send to the requester the Access Decision and the response to the Access Request. The Access Decision can be used as a token for re-accessing the same service without undergoing the authentication process.

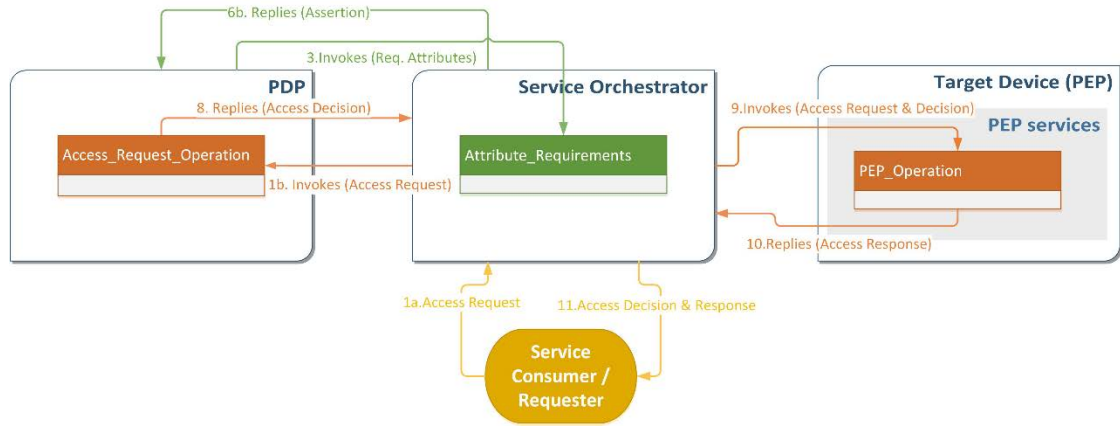
The framework can trivially be expanded to cater for the joint operation of two or more access control infrastructures (i.e. PDPs and corresponding PIPs/PAPs). This can be used as a means to consolidate the requirements of different stakeholders and their active policy sets. In such a case, the SO can query all the different PDPs and provide or deny access based on pre-defined simple rules (e.g. only in cases where all PDPs explicitly allow such access). So, for example, someone's request to access the patient's blood sugar levels will only be forwarded to the pertinent medical device if both the patient and the attending doctor have authorized the specific individual to perform such an action.

### 5.2.3 IMPLEMENTATION APPROACH

Sun's XACML [85] was used for this implementation as well. All of the framework's entities were implemented and their interfaces exposed using DPWS. This facilitates the discovery and description of the devices involved, also offering control and eventing mechanisms which assist in the communication of the necessary information among the entities. The DPWS API of choice is again the WS4D-JMEDS (Java-based) stack [145], as it is the most advanced and active work of the WS4D initiative [146], supporting almost all of the existing DPWS features and providing portability to a wide range of platforms. The approach adopted to protect the messaging of the proof of concept implementation is the use of WS-Security [124].

The exact implementation of the framework's entities and their communication interfaces depicted in Figure 44 are detailed below.





**FIGURE 44. DPWS-BASED IMPLEMENTATION OF THE AUTHENTICATION SCHEME.**

### 5.2.3.1 SERVICE ORCHESTRATOR TO POLICY DECISION POINT

The SO is implemented as a DPWS peer (i.e. both a client and a server). Other than the necessary mechanisms needed to interface with the approved identity providers (which will vary depending on the specific scenario/deployment examined), it also features an "Attribute\_Requirements" operation. Similarly, the PDP has an "Access\_Request\_Operation". The latter is invoked by the SO as soon as an access request arrives from a service consumer, relaying the request for evaluation. As soon as the XACML decision-making process is completed, the PDP replies to the invocation with its access decision. As detailed in the information flow above, prior to providing a decision, it may need to invoke the "Attribute\_Requirements" operation on the SO, in order to inform it of the needed user attributes, getting the proper assertion as an answer.

### 5.2.3.2 SERVICE ORCHESTRATOR TO POLICY ENFORCEMENT POINT

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), one extra operation must be present on each DPWS device, namely the "PEP\_Operation". The SO, acting as a client, invokes this operation providing the service consumer's access request along with the decision (pre-issued by the PDP) as input. If the decision accompanying the invocation is positive, the PEP replies to the SO with the resource (e.g. temperature reading) that the service consumer originally tried to access. This information is then relayed to the service consumer/requester. The above DPWS-based communication mechanisms are depicted Figure 44.

## 5.2.4 PERFORMANCE EVALUATION

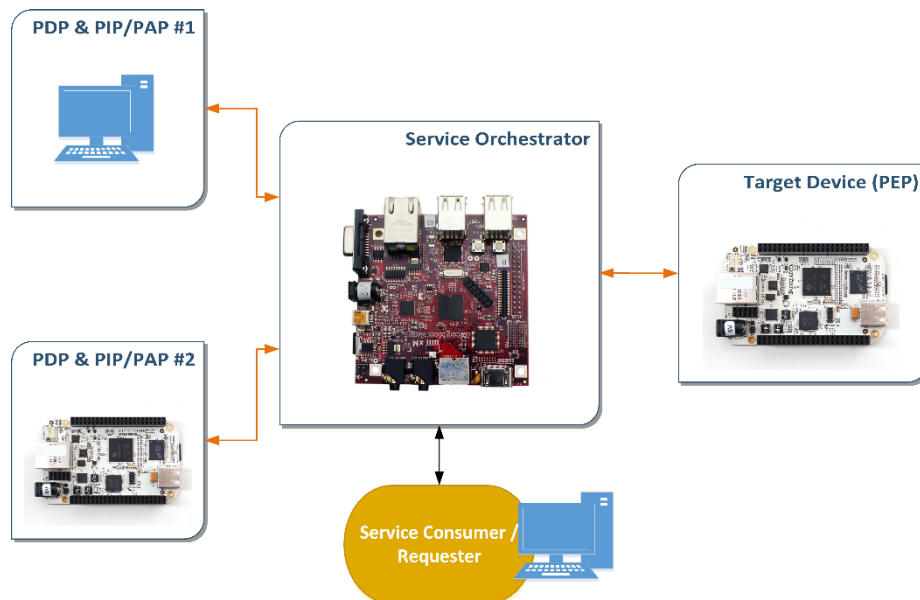
### 5.2.4.1 TEST-BED SETUP

The platform-agnostic nature of SOAs enables the proposed framework to be deployed, by design, on a variety of platforms and operating systems. However, in order to realistically assess the performance of the proposed framework, the developed entities had to be deployed on devices expected to be present in healthcare deployments. Therefore, the proposed framework was implemented and its performance was evaluated on a heterogeneous environment, featuring relatively resource-constrained embedded platforms as well as desktop computers.

The PEP-equipped target device (i.e. the device providing the actual service to be accessed) was on a Beaglebone [107], a low-cost credit-card-sized embedded device that runs a compact Linux-based operating system. It uses an ARM Cortex-A8 single core CPU running at 720MHz (throttled at 500MHz during testing) with 256MB DDR2 RAM. The test-bed for the Service Orchestrator was a similar but slightly more powerful and versatile Beagleboard-xM embedded platform [106], featuring an 1GHz ARM Cortex-A8 processor (throttled to run at 600MHz during testing) and 512MB DDR2 RAM, also running a minimal Linux-based operating system. The access control infrastructure entities, i.e. the PDP and PIP/PAP, were deployed on a desktop system (Core i5 CPU at 3.3GHz, 8GB DDR3 RAM). An identical desktop system was used to run the service consumer, a client application programmed to automatically invoke the resources exposed by the SO and record response times, for benchmarking purposes.

Tests also included a second scenario where an extra PDP and PIP/PAP were deployed on a more resource-constrained platform, namely a Beaglebone embedded device, like the one used for the target device (i.e. the PEP). The latter was used to investigate the performance impact when the SO has to query two different PDPs, each with its own policy set, to emulate the use case where e.g. a patient and a hospital each have their own access control infrastructure and policy requirements. In this scenario, the SO had to evaluate both responses and only allow the user to access the resources if both PDPs allowed such access.

The test-bed setup described above is depicted in Figure 45. Note that this setup is by no means the only option for the proposed framework's deployment. For instance, a Beaglebone was chosen for the SO to simply demonstrate the ability of the SO to be deployed even in a constrained environment of an embedded system. In a large-scale deployment one would expect the functionality of the SO to be deployed at an application server to ensure the system is able to serve a sufficient number of users.



**FIGURE 45. THE TEST-BED SETUP, FEATURING EMBEDDED DEVICES AND DESKTOP PCS. ORANGE LINES INDICATE COMMUNICATION WHERE WS-SECURITY IS OPTIONALLY ENABLED. ALSO DEPICTS THE EXTRA PDP & PIP/PAP INTRODUCED IN THE SECOND TEST SCENARIO.**

Aiming to also assess the performance impact in situations where the messages exchanged would have to be secured, an alternative proof-of-concept implementation was developed adopting the security mechanisms specified in WS-Security. These mechanisms safeguard the integrity and confidentiality of the policy messaging exchanged by the framework's entities.

The application profiling (i.e. CPU and memory utilization) was focused on the Service Orchestrator, which is the main entity of the proposed approach, and on devices which are expected to have resource limitations, i.e. the PEP-equipped target device. Moreover, the impact on user experience was also assessed, by recording client-side response times in all usage scenarios.

The steps related to the Identity Provider were omitted during testing, as these will vary depending on the Identity Provider that the user will choose and are deployment-specific, thus out of the scope of the framework presented in this work.

#### 5.2.4.2 RESULTS

A total of 100 consecutive requests were issued from the service consumer application to the SO residing on the Beagleboard-xM. The response time recorded by the test client trying to access the target device's resources appear in Figure 46. The WS-Security mechanisms impose a significant overhead to the response times, which is expected given the use of asymmetric cryptographic mechanisms. In contrast, the response times for the second scenario indicate that the introduction of a second instance of the PDP and PIP/PAP is not prohibitive, while allowing to consolidate the policy requirements of different stakeholders.

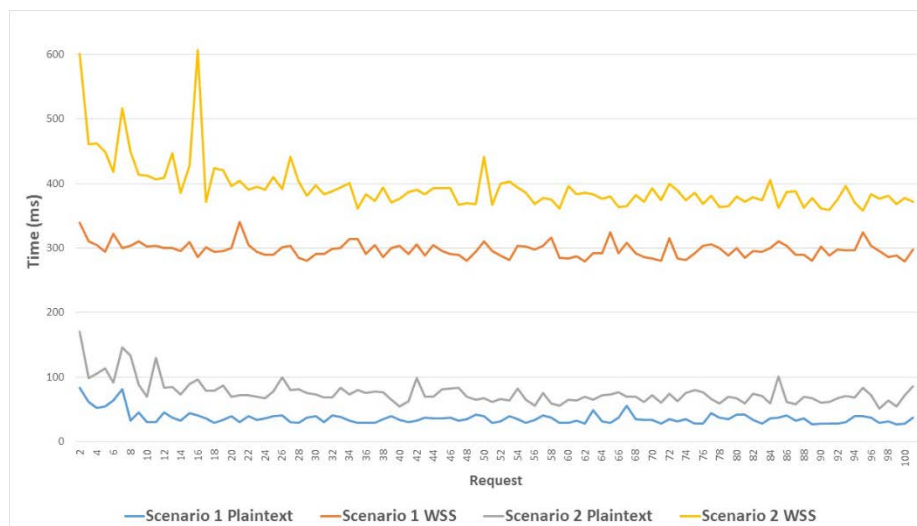
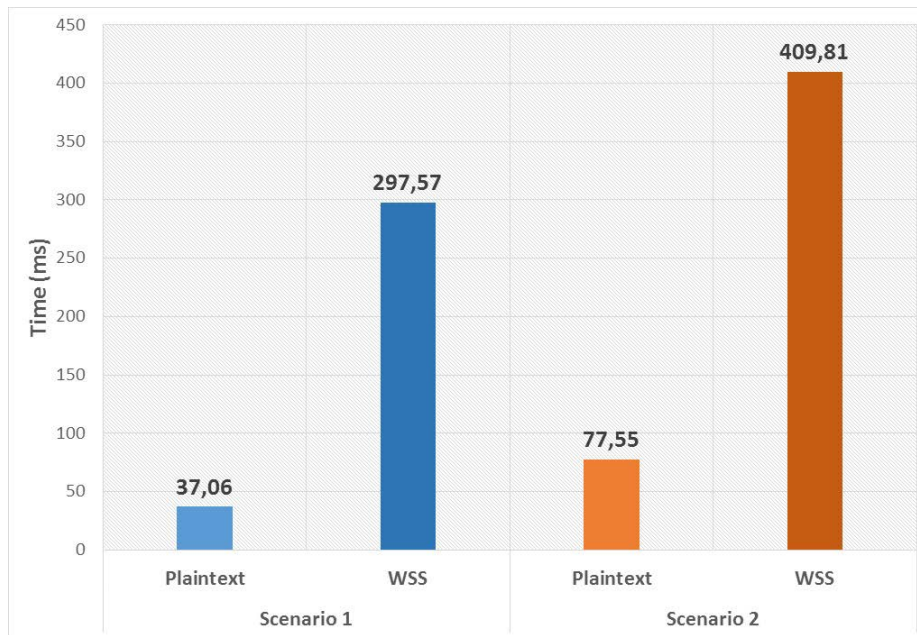


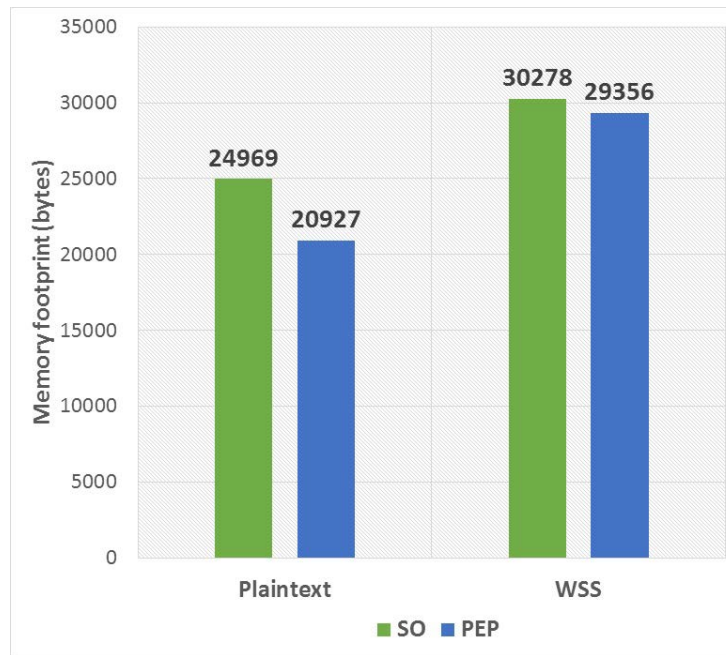
FIGURE 46. CLIENT-SIDE RESPONSE TIME FOR 100 REQUESTS TO THE SERVICE ORCHESTRATOR.



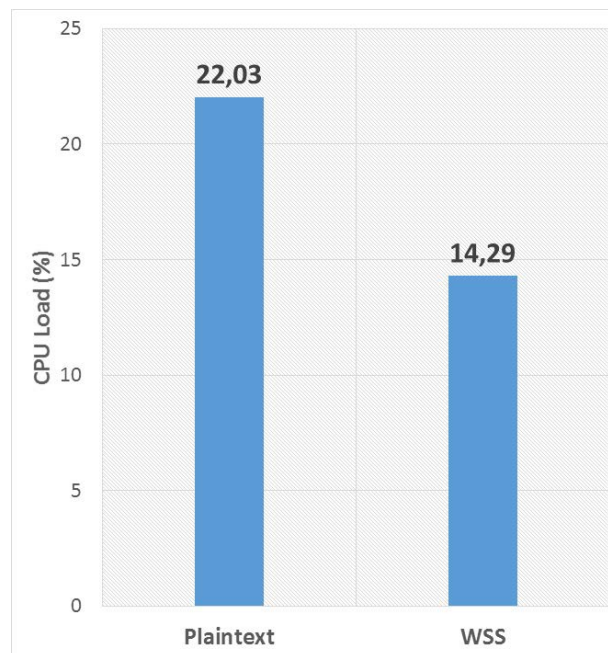
**FIGURE 47. SERVICE ORCHESTRATOR'S AVERAGE CPU LOAD (%).**

Profiling of the SO revealed a lightweight application, even under the load of consequent requests, or in the presence of two PDP and PIP/PAP instances. The average CPU load and memory consumption appear in Figure 47 and Figure 48 respectively. As the occupied memory remains constant irrespectively of the presence of one or two PDPs, the numbers for the second scenario are omitted. The use of WSS imposes a relatively small memory overhead, while the average CPU load drops, as the device has to wait more between requests, due to the network and processing overhead on other framework entities.

The same behaviour with regard to CPU load was also recorded on the target device (i.e. the device featuring the PEP), as is depicted in Figure 49. As in the case of the SO, introducing the WSS mechanisms increases the memory footprint (appearing along with SO values in Figure 48), but the latter, along with CPU load, are not significantly affected by the presence of multiple PDPs and the corresponding PIP/PAPs, thus the numbers of the second scenario are omitted from the corresponding figures.



**FIGURE 48. SERVICE ORCHESTRATOR'S AND TARGET DEVICE'S MEMORY UTILIZATION (IN BYTES) FOR SCENARIO 1.**



**FIGURE 49. TARGET DEVICE'S CPU LOAD (%) FOR SCENARIO 1.**

### 5.2.5 SECURITY CONSIDERATIONS

One of the main concerns in accessing services and issuing commands, is the protection of the data being exchanged among the participating entities. In the proposed scheme the service provider has a pre-established relationship with the SO, PDP and PAP. Note that all these three entities are only functional components and therefore the exact needs in secure channel establishment depend on the actual deployment choice and cannot be specified. In a

simplified approach, the SO, PDP and PAP can be part of the same entity and therefore a secure channel establishment using pre-shared keys is a viable and efficient option.

Regarding the underlying message security mechanisms, there are a number of proposed or standardized schemes that handle the protection of messages at various layers of the network stack. The WS-Security mechanisms adopted for the proof of concept implementation is typically used alongside DPWS, but its public-key security primitives can impose a significant performance overhead, as is evident from the performance evaluation presented in the previous section. Therefore, considering the resource-constrained nature of some devices, and the need to minimize performance impact in general, alternative cryptographic primitives can also be investigated for production environments.

It is expected that some of the framework's entities will be deployed on normal, relatively powerful nodes (personal computers or even servers). Thus, e.g. the link between the Requester and the SO could alternatively be protected using common methods, like TLS, the same way that the communication channel between the Requester and the IdP is anticipated to be protected, although the latter is outside the scope of this work. The cost of using TLS, however, between the Requester and the SO is that the secure channel breaks at the SO and the SO has to re-encrypt the communication using the security parameters set for the link between the SO and the service provider.

The actual authentication scenario could be further elaborated during deployment to match system owner's specific requirements and trust relationships with identity providers. Several options in such a deployment exist as they have been demonstrated in [147].

The proposed scheme provides the Service Provider the flexibility to change the orchestrator(s) it uses based on its needs. This also applies to applicable policy rules which the service provider can modify to match his/her requirements. As an example, consider the situation where the owner of the mobile device being used to offer these services, changes mobile operator. He/she simply has to change SO, to a platform operated by the new mobile operator, and register his/her policies with it. Use of the SO provides additional benefits which are related to the node's connectivity. The node can wake up occasionally to fetch any requests sent to the SO. This approach also helps save node's resources, as no requests are sent to the node unless the latter asks for it. If the service request was sent directly to the PEP, the corresponding device would have to always be online, otherwise the service would be unavailable.

#### 5.2.6 SUMMARY

As computing becomes ubiquitous, adopters aim to exploit the potential of pervasive systems, including LLN nodes bearing sensors and actuators, in order to introduce new types of services and address inveterate and emerging problems, healthcare being one of the most prominent application. Nevertheless, a key factor in the wide adoption and success of these new technologies is the effectiveness with which the various security and privacy concerns are tackled within the resource-constrained environment.

To this end, this section proposed an architecture for providing robust authenticated access control to heterogeneous resource-constrained devices. The scheme builds upon the standardized technologies, namely access control mechanisms based on XACML and SOA-based interfacing of its key entities. In contrast to typical XACML deployments, the core PEP functionality and the hosting resource-constrained device are efficiently relieved from the expensive computations that the XACML standard defines, without sacrificing any of the policy-based decision making process. The device is sheltered from direct user interaction, helping alleviate concerns that are typical to resource-constrained devices, like DoS attacks. Emphasis was given on the scheme's ability to serve users authorized by, typically, any authentication scheme, thus enabling the large-scale deployment of the solution to many environments.

As a proof of concept, the components of the proposed scheme were developed and deployed on a heterogeneous test-bed featuring desktop systems and typical embedded devices. The performance overhead imposed on the three most important endpoints, i.e. the client attempting to access the protected resources, the Service Orchestrator and the PEP, was analyzed and presented to demonstrate the feasibility of the suggested solution.

### 5.3 CROSS-DOMAIN SMART ENVIRONMENTS –THE XSACD VARIANT

In recent years, massive advancements in computing and communication technologies have led to what can only be described as a revolution in terms of how people perform the various tasks comprising their everyday lives; a revolution enabled by the ubiquitous presence of computing devices in all aspects of modern life. These major changes could not leave the residential environment unaffected, with smart homes gradually becoming a reality. In these cases, homes may feature sophisticated lighting (e.g. smart light bulbs), ambient environment controls (e.g. heating, ventilation and air conditioning via smart thermostats), appliances (smart -fridge, -oven, -washing machine, -coffee makers etc.), communication systems (including smart phones), entertainment (e.g. smart TVs), and home security (smart cameras, door and window controls etc.) devices. Moreover, the residential environment borders with other ubiquitous computing applications, like smart metering and e-health, as these will have to be integrated into the smart home ecosystem. Nevertheless, as said devices typically handle personal sensitive data and often feature direct interaction with the physical world, a key factor in the wider adoption and success of these new technologies will be the effectiveness with which the various security and privacy concerns are tackled. A necessary instrument in successfully addressing these issues is the presence of robust access control mechanisms and seamless management of devices.

To this end, we adopt uSPBM to the smart home / consumer environment, in the form of the Cross-domain Service Access Control for devices (XSACd) framework. By leveraging uSPBM's mechanisms, new devices can easily join existing networks and offer services protected by a predefined or dynamic policy set. Based on the policy rules set by the system owner, the proposed architecture provides fine-grained AC over the plethora of devices and services that may be found in smart home environments. Thus, XSACd assists in the use of the various smart devices aiming to enhance consumers' lives, while addressing their security concerns.

A key limitation to the use of DPWS across different domains, is that its device discovery is limited to local networks (as it is based on UDP multicast messages). To address this, XSACd also introduces proxies that, based on an external, Internet-based, broker, can enable the discovery and interaction with DPWS devices residing in other networks, in an automated manner (i.e. in a seamless way from the user's perspective).

### 5.3.1 MOTIVATION

In a typical ubiquitous-computing-enhanced residential setting, various smart devices are expected to be present on appliances (e.g. smart fridge) and automation-enabled structures (e.g. smart doors), also including environmental sensors and actuators. Moreover, these are typically complemented by purpose-built devices intended to organize, manage and enhance the functionality of the rest of the smart infrastructure, like energy monitors and control nodes (e.g. a computing system with touch-based input to allow seamless monitoring and interaction with the devices).

This heterogeneous assortment of devices will feature a variety of services, each with its own intrinsic characteristics (some being critical in terms of the residents' safety, others dealing with private sensitive data etc.), thus requiring a different protection profile. For example, all residents should be able to control the smart doors and windows of a house, but, perhaps, children should not be able to tamper with a subset of those (e.g. front door) at certain timeframes (e.g. during the night). In another scenario, visitors may have the rights to monitor the environmental sensors of the residence, but not to set the climate control at their will. Moreover, the residence owners may decide they feel alright with visitors checking the contents of their smart fridge, but they, expectedly, should not be able to add goods to the shopping lists. Assuming the presence of e-health devices in the smart home ecosystem, it is anticipated that the patient, her spouse and medical staff should be able to monitor the various readings and control the actuators that deliver the prescribed medicine, but only the latter group should have access to the service that controls the drug dosage. Moreover, it would be desirable to allow medical staff to operate on the devices remotely, to avoid unnecessary visits to the hospital. In cases where the residence is equipped with smart-metering devices, authorized power company staff should be the only ones able to adjust and/or reset the meters remotely (for billing purposes), but, nevertheless, the owners should be able to access the consumption readings as well.

Furthermore, a survey [20] on smart home users revealed that inflexibility (often forcing users to adopt solutions offered by a single manufacturer) and difficulties in achieving security constitute significant barriers to the broader adoption of pertinent technologies and devices.

From the above, and considering that, typically, the only pervasive protection mechanism present in home environments is the access to the wireless network, it is evident that strong and interoperable access control mechanisms are required to safeguard a variety of aspects pertaining to the operation of a smart home environment. Additionally, this should be achieved in a flexible, platform-agnostic manner, acting as an enabler instead of introducing new (or further exacerbate existing) obstacles to the adoption of "smart" devices and services.



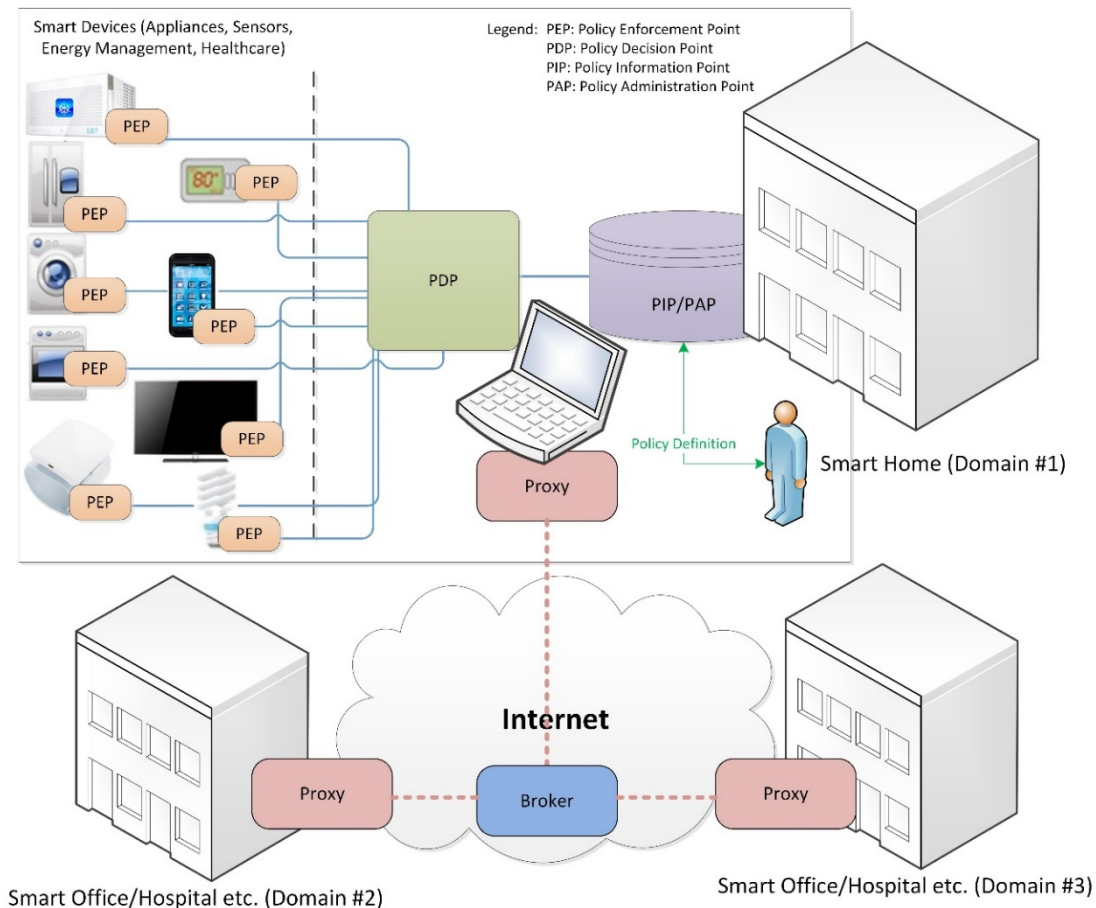
To this end, the presented framework is based on standardized mechanisms, which also allows leveraging work already carried out both in terms of Web Services as well as XACML policy definitions. DPWS can enable user-to-machine and M2M interactions in a unified manner, moving on from the current state of the field, where consumer electronics manufacturers offer a variety of proprietary protocols which are not interoperable and essentially lock-in consumers, forcing them to use a specific vendor/ecosystem. With regard to XACML, the scheme can trivially be expanded to cater for additional specific concerns, such as privacy issues and/or the handling of sensitive data (e.g. healthcare, as covered by the relevant OASIS Cross-Enterprise Security and Privacy Authorization profile for XACML v2.0 [141]).

### 5.3.2 PROPOSED ARCHITECTURE

In the proposed framework, the following key entities are present and can be deployed on different smart home nodes, depending on their role and resources:

- **Policy Enforcement Point (PEP):** Makes decision requests and enforces authorization decisions. This is expected to be present in every smart device (appliances, sensors, e-health de-vices, energy monitoring or smart metering devices etc.) which provides its resources to the end users, and which need to be protected by the active policy set.
- **Policy Decision Point (PDP):** Evaluates requests against applicable policies and renders an authorization decision. It is expected to be deployed on more feature-rich nodes, typically a personal computer or an embedded system that acts as a controlling node for the whole smart home infrastructure
- **Policy Administration Point (PAP) & Policy Information Point (PIP):** The former creates and manages policies or policy sets, while the latter acts as a source of attribute values. These two entities will typically be deployed on the same feature-rich node, facilitating direct interaction with end-users (e.g. home owners). A desktop computer or a laptop are good candidates for this role.
- **Cross-domain Proxy:** This entity is responsible for catching all discovery messages of the network, processing their contents and transmitting them to other networks, also transmitting to the local network all messages received from other domains.
- **Broker:** This is the main entity through which all cross-domain traffic is routed. The broker is responsible for distributing messages to all interested clients (i.e. the proxies) based on a message's topic. To this end, all proxies have to subscribe to the Broker.

As is evident from the above, and considering that nodes embedded in a smart home may not have the computing resources to accommodate expensive mechanisms, the core decision process is under-taken by more powerful nodes expected to operate within the node's trusted environment. Such an approach allows requests to be directly addressed to the node in question, while maintaining the capability to centrally manage and control access to these nodes. An overview of the architecture can be seen in Figure 50.



**FIGURE 50. SMART HOME ACCESS CONTROL ARCHITECTURE & CROSS-DOMAIN COMMUNICATION. MAIN ENTITIES.**

### 5.3.3 IMPLEMENTATION APPROACH

As with all uSPBM implementations, Sun's XACML engine [85] was the basis of the access control mechanisms, while WS4D-JMEDS [145] was used to expose the framework's entities via DPWS. Using the above API and exploiting the features of DPWS, the XACML functionality can be exposed to the home network (and the Internet, if needed), allowing the seamless discovery and communication of the framework's entities, regardless of the device where they may be deployed. In more detail, the XACML features are exposed as follows.

#### 5.3.3.1 PEP TO PDP IMPLEMENTATION

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 USPBM-specific operations.

More specifically, the first operation is the "SAR<sub>Event</sub>" (Service Access Request Event), referring to an operation following the WS-Eventing [95] specification to which devices can subscribe. When fired, the operation outputs "SAR<sub>Out</sub>" (Service Access Request Output), a message which includes all the information the PDP needs to have in order to evaluate a

request (i.e. Subject, Action and Resource). The second operation is "PDPResponse" (Policy Decision Point Response), which is invoked by the PDP to relay an answer to a pending access request.

### 5.3.3.2 PDP TO PIP/PAP IMPLEMENTATION

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). This extra operation is named "PIPOperation" (Policy Information Point Operation). It features an input for the request issued by the PDP (requesting all applicable policy rules), and an output containing all the pertinent information (i.e. policies and rules) that the PIP has identified.

### 5.3.4 EVENT SEQUENCE

The above DPWS operations and the sequence of events that take place when an access request is received for a protected resource are depicted in Figure 51.

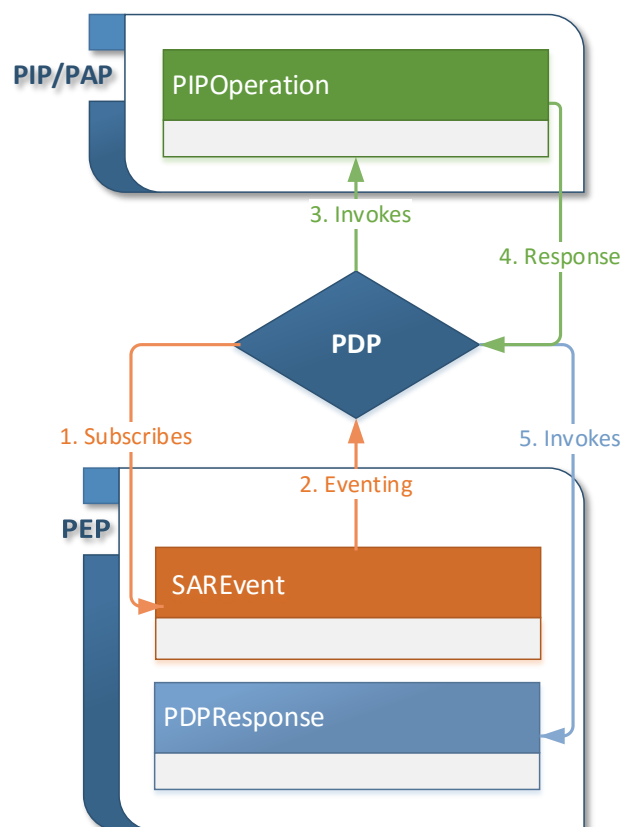


FIGURE 51. DPWS IMPLEMENTATION OF THE XACML MECHANISMS

In more detail, the PDP is implemented as a DPWS client, constantly monitoring the network for "Hello" messages transmitted by DPWS devices as they initialize. Whenever it discovers a PEP-equipped device, it automatically subscribes to its "SAREvent" operation.

Each time a user tries to access a resource on one of these AC-protected devices, the "SAREvent" is fired, notifying the PDP that a request has to be evaluated, and transmitting all the information required for XACML to make such a decision. This information includes the

Request ID (a counter of policy decision requests issued by the specific device), 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*).

The above data is used by the PDP to generate an XACML request, i.e. a request in a form that can be evaluated by the XACML decision engine. This process also involves the PDP invoking the "PIPOperation" of the PIP/PAP, in order to retrieve the applicable policies.

Based on the information included in the PEP request and the pertinent policies retrieved from the PIP/PAP, the PDP can then decide if the user's pending request is authorized or not. This decision is conveyed to the PEP by invoking its "PDPResponse" operation.

Finally, based on the decision received, the PEP either grants or denies access to the device's re-source that the user initially tried to access.

### 5.3.5 USPBM/XSACD CROSS-DOMAIN PROXY

On its own, DPWS does not support cross-domain communications, since device discovery is based on a UDP multicast protocol, WS-Discovery [94]. So, without the use of a purpose-built proxy, all the above entities must be on the same network in order to operate. Our proxy implementation is not based on a central server that catches all "hello" and "bye" messages as is typical for such proxies; this presumes that at every discovery proxy has a list of known devices and that proxies communicate each other to "share" devices, which is not really usable.

XSACd's proxy uses MQ Telemetry Transport (MQTT [24]), a publish-subscribe OASIS standard protocol, as the backbone of communication between networks. An MQTT Broker, deployed somewhere on the Internet, is responsible for handling and organizing all communications between the various domains and their corresponding proxies. In every local network a MQTT client is deployed which has two roles:

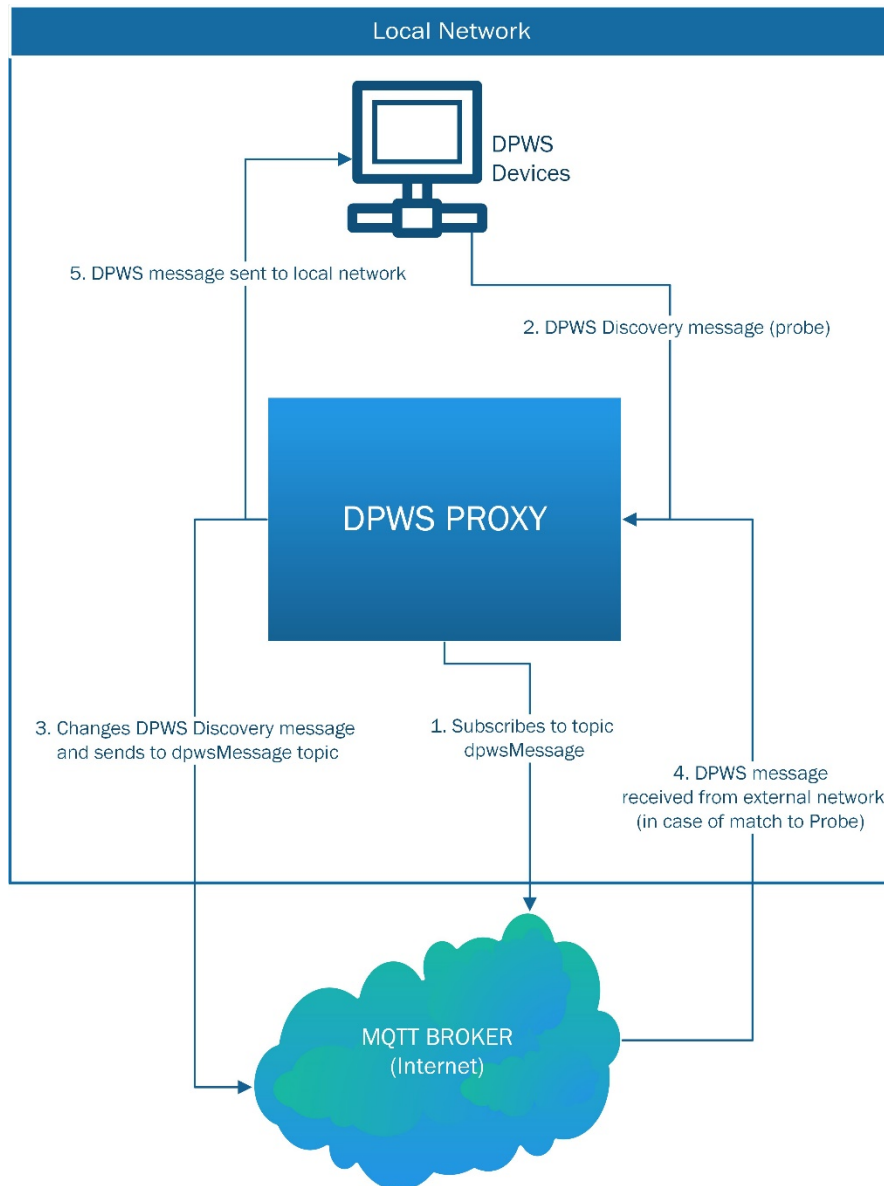
- catch all discovery messages of the network, handle the info and transmit them at the other DPWS networks and
- transmit to the current network all messages that are published from other networks.

Catching all messages from the network is done by adding a membership to a predefined address, namely 239.255.255.250:3702, which is the UDP broadcast and default port of DPWS discovery default settings. Then every SOAP xml message is parsed; if it is a "hello" message the IP of the message is changed from the 192.168.\*.\* ,which is normally used in a local network, to the external IP, also using NAT-PMP (if the network's router supports it) the local port is forwarded, so that external communication is achieved. Then the changed "hello" message is transmitted to all the other DPWS proxy subscribers. When a client from a local network searches (sends a probe) for a device, the proxy saves the ID of the message and also the information of the sender. That way, if a device is found that meets the search criteria (i.e. a probe match is received), it will only transmit the response to the local network that made the search. Moreover, the sender information is kept because, when a client sends

a probe message, it waits for the probe response at the same port. Moreover, as every message ID is saved, duplicate transmissions are eliminated.

It must also be mentioned that for multi-home communication to be fully supported, after the device discovery (facilitated by the DPWS proxy), the client asks the target device of its metadata, provided in the form of a WSDL [99] file. In order to have a successful communication, the WSDL file must contain the external IP of the device. This was not supported in any available DPWS implementation, so we modified our custom DPWS implementation in the Node.js programming environment [148], allowing us to track the `GetMetadata` message: if it originates from the local network, it uses the local IP (192.168.\*.\*) at the WSDL, otherwise it uses the router's external IP. Any standard MQTT Broker could be used to interact with the XSACd proxies. For the proof-of-concept implementation, we opted for the Mosquitto open source message broker [149].

Figure 52 depicts a simplified view of the above, along with the steps that take place during normal operation; in this example, when a device from the local network issues a probe match and gets a reply from a device deployed in another domain. All external traffic is routed through the local router in a typical home deployment; to this end, the proof-of-concept implementation automates port forwarding.



**FIGURE 52. XSACD CROSS-DOMAIN PROXY IMPLEMENTATION AND MAIN STEPS. SIMPLIFIED VIEW.**

A screenshot of a command line remote connection to the proxy appears in Figure 53.

```

ubuntu@arm:~/node/dpws_proxy$ node app.js
'Adding membership to DPWS'
'Connected to mqtt broker'
'Received message from DPWS device'
'<?xml version="1.0" encoding="UTF-8"?><s12:Envelope xmlns:dpws="http://docs.oasis-open.org/ws-dd/ns/dpws/2009/01" xmlns:s12="http://www.w3.org/2003/05/soap-envelope" xmlns:wsa="http://www.w3.org/2005/08/addressing" xmlns:wsd="http://docs.oasis-open.org/ws-dd/ns/discovery/2009/01"><s12:Header><wsa:Action>http://docs.oasis-open.org/ws-dd/ns/discovery/2009/01/Probe</wsa:Action><wsa:MessageID>urn:uuid:830f43e0-50d5-11e5-8031-42eb67a2c253</wsa:MessageID><wsa:To>urn:docs-oasis-open-org:ws-dd:ns:discovery:2009:01</wsa:To></s12:Header><s12:Body><wsd:Probe /></s12:Body></s12:Envelope>'
'Received message from DPWS device'
'<?xml version="1.0" encoding="UTF-8"?><s12:Envelope xmlns:dpws="http://docs.oasis-open.org/ws-dd/ns/dpws/2009/01" xmlns:s12="http://www.w3.org/2003/05/soap-envelope" xmlns:wsa="http://www.w3.org/2005/08/addressing" xmlns:wsd="http://docs.oasis-open.org/ws-dd/ns/discovery/2009/01"><s12:Header><wsa:Action>http://docs.oasis-open.org/ws-dd/ns/discovery/2009/01/Probe</wsa:Action><wsa:MessageID>urn:uuid:830f43e0-50d5-11e5-8031-42eb67a2c253</wsa:MessageID><wsa:To>urn:docs-oasis-open-org:ws-dd:ns:discovery:2009:01</wsa:To></s12:Header><s12:Body><wsd:Probe /></s12:Body></s12:Envelope>'
'Received MQTT Message'
'<?xml version="1.0" encoding="utf-8"?><s12:Envelope xmlns:dpws="http://docs.oasis-open.org/ws-dd/ns/dpws/2009/01" xmlns:s12="http://www.w3.org/2003/05/soap-envelope" xmlns:wsa="http://www.w3.org/2005/08/addressing" xmlns:wsd="http://docs.oasis-open.org/ws-dd/ns/discovery/2009/01"><s12:Header><wsa:Action>http://docs.oasis-open.org/ws-dd/ns/discovery/2009/01/Probe</wsa:Action><wsa:MessageID>urn:uuid:830f43e0-50d5-11e5-8031-42eb67a2c253</wsa:MessageID><wsa:To>urn:docs-oasis-open-org:ws-dd:ns:discovery:2009:01</wsa:To></s12:Header><s12:Body><wsd:Probe /></s12:Body></s12:Envelope>'
'Received MQTT Message'
'<?xml version="1.0" encoding="utf-8"?><s12:Envelope xmlns:dpws="http://docs.oasis-open.org/ws-dd/ns/dpws/2009/01" xmlns:s12="http://www.w3.org/2003/05/soap-envelope" xmlns:wsa="http://www.w3.org/2005/08/addressing" xmlns:wsd="http://docs.oasis-open.org/ws-dd/ns/discovery/2009/01"><s12:Header><wsa:Action>http://docs.oasis-open.org/ws-dd/ns/discovery/2009/01/Probe</wsa:Action><wsa:MessageID>urn:uuid:830f43e0-50d5-11e5-8031-42eb67a2c253</wsa:MessageID><wsa:To>urn:docs-oasis-open-org:ws-dd:ns:discovery:2009:01</wsa:To></s12:Header><s12:Body><wsd:Probe /></s12:Body></s12:Envelope>'

```

FIGURE 53. XSACD CROSS-DOMAIN PROXY. COMMAND LINE REMOTE CONNECTION TO THE EMBEDDED TEST PLATFORM

Compared to the proxy presented by Müller et al [57], the XSACd approach has various benefits. Scaling is an issue with the previous work, as in the case of N networks, each proxy must communicate with N-1 other proxies. So, for example, a proxy must send its probe match request to N-1 other networks, initiating an equal number of connections. In our approach, each proxy only needs to communicate with the MQTT Broker, which is responsible for disseminating all DPWS messages to the other registered proxies. Moreover, there is added complexity in that the proxy has to tamper with the metadata files, whereas in XSACd all devices properly formulate their metadata files by themselves. An additional complication is that they propose different types of proxies (client proxy & server proxy), depending on the network's role, whereas an XSACd proxy can accomplish both roles simultaneously. Finally, there is a significant gap in the performance and reliability in favor of the MQTT approach adopted in this work compared to the HTTP-based communications of the previous work, as is evident from the comparison benchmarks appearing in Table 8.

TABLE 8. PERFORMANCE & RELIABILITY COMPARISON OF MQTT AND HTTP ON TYPICAL MOBILE APPLICATION<sup>10</sup>

		3G		WiFi	
		HTTPS	MQTT w. SSL	HTTPS	MQTT w. SSL
Receive	msg/hour	1,708	160,278	3,628	263,314
	%battery/msg	0.01709	0.00010	0.00095	0.00002
	msg delivery	240 / 1024	1024 / 1024	524 / 1024	1024 / 1024
Send	msg/hour	1,926	21,685	5,229	23,184

<sup>10</sup> <http://stephendnicholas.com/archives/1217>

	%battery/msg	0.00975	0.00082	0.00104	0.00016
--	--------------	---------	---------	---------	---------

### 5.3.6 SECURITY CONSIDERATIONS

The effectiveness of any access control mechanism can easily be compromised unless appropriate security mechanisms are deployed to protect policy messaging. A malicious entity would otherwise be able to eavesdrop, replay or tamper with the access control messaging, overriding the offered protection to provide access to unauthorized entities or denying access to authorized ones. When feasible, deployments over trusted and/or secure networks (e.g. over a Virtual Private Network, VPN) can address most of these concerns, but an alternative mechanism has to be considered for deployments where these provisions are not realistic.

A detailed analysis of the protection of the access control –related communications can be found in chapter 4.3. In terms of protecting the communication of the proxies with the MQTT broker, there are various options that could be explored. As of version 3.1, MQTT supports the use of a username and password to secure the communication with the broker. There is also the option of using web sockets and secure web sockets. Moreover, since MQTT is based on TCP, one could always opt for the use of more typical socket security mechanisms, such as SSL/TLS, but this would add a notable amount of processing overhead. In this proof-of-concept we used a symmetric mechanism based on AES/CCM [134] to encrypt every payload sent from the publisher to the subscribers. Thus, a username and password is used to secure access to the MQTT broker and, moreover, the messages exchanged are encrypted, to protect them from eavesdropping and tampering (using the authenticated encryption mechanisms of AES/CCM).

### 5.3.7 PERFORMANCE EVALUATION

The use of platform-agnostic technologies (i.e. DPWS and Java) enables the proposed framework to be deployed, by design, on a variety of platforms and operating systems. However, in order to realistically assess the performance of the proposed framework and its impact on the target devices, the developed entities have to be deployed on devices expected to be present in smart home environments.

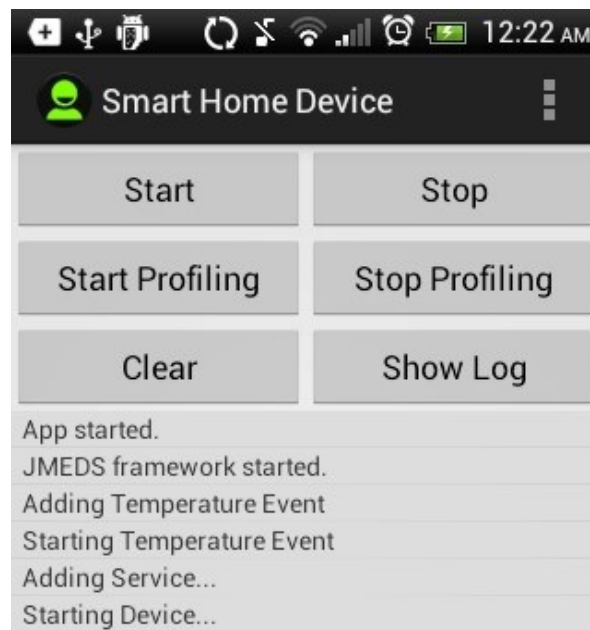
Therefore, the infrastructure entities, namely the PDP and PIP/PAP, were deployed on a laptop (quad core CPU at 2.6GHz, 4GB RAM), as a personal computer is typically available in home environments and is expected to act as a management hub through which the residents monitor and control their smart residence. A total of 50 policies were stored in the policy repository, which the authors considered a realistic approximation of the number of policies needed, considering the relatively limited number of devices expected to reside in a smart home. Tests were also carried out with 500 policies, to assess the impact more policies would have on the framework's performance

Regarding the target platforms – i.e. the platforms featuring the services that need to be protected – we chose to use relatively resource-constrained smart embedded devices (600MHz low power single core CPU, 512MB RAM) running the Android open source operating system for mobile devices. Such operating systems are already found in many smart



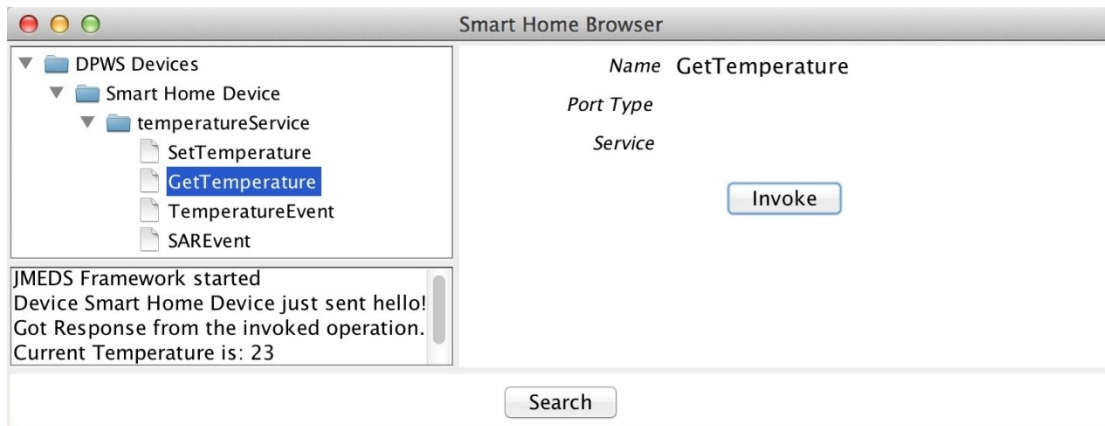
commercial appliances (e.g. smart fridges) offered by the various consumer vendors. Moreover, their adoption is expected to spread as more sophisticated home devices become available to end users; thus, the above platform can be considered a realistic choice for evaluating the performance of the proposed mechanisms.

The DPWS device deployed on the smart platform not only featured the access control related operations (as depicted in Figure 51) but also featured three simple operations, emulating part of the functionality of a smart appliance (e.g. smart fridge). Via the above operations, the user can get the current temperature, subscribe to a service that periodically informs of said temperature and also set the desired temperature when needed. A basic touch GUI was developed for this device, which can be seen in Figure 54.



**FIGURE 54. SCREEN CAPTURE OF THE (PEP-PROTECTED) DPWS TEST DEVICE DEPLOYED ON THE TOUCH-ENABLED SMART PLATFORM.**

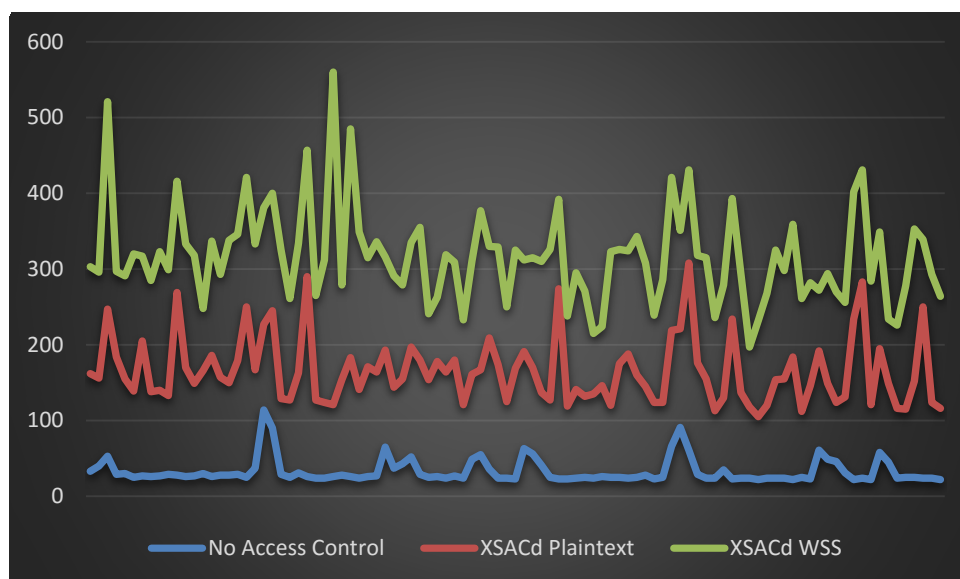
A client application was also developed for testing purposes; the “Smart Home Browser”. This application is deployable on various end devices (personal computers, smart phones or tablets) and allows users to discover and control the various DPWS-enabled smart appliances (to get the current contents of the smart fridge, to subscribe to the power consumption readings provided by the smart metering device etc.). A screenshot of the Smart Home Browser prototype implementation appears in Figure 55.



**FIGURE 55. THE "SMART HOME BROWSER"; AN APPLICATION DEVELOPED TO FACILITATE THE DISCOVERY OF DPWS DEVICES AND PROVIDE ACCESS TO THEIR HOSTED SERVICES.**

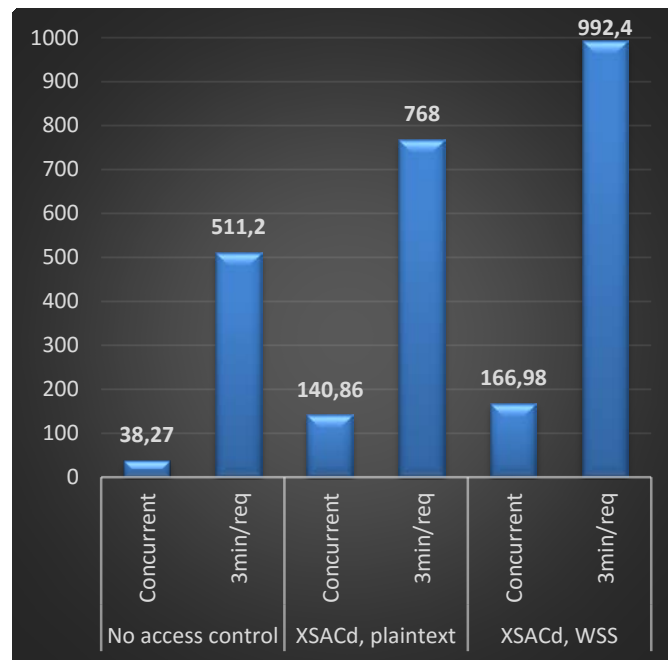
A command line-only variation of this client, programmed to automatically invoke operations and record response times, was developed for benchmarking purposes. This benchmark client was used to evaluate the performance of three setups: a simple DPWS device with no PEP implemented (i.e. with direct access to its services), a DPWS device protected by the presented access control entities communicating in plaintext, and a third setup with the entities' communications being protected via WS-Security. This allows us to separately assess the impact of the access control functionality and the impact of the security mechanisms that may be needed to protect the policy messaging in some deployments.

In addition to the client-side measurements, the CPU and memory utilization was also monitored on both the personal computer that hosted the PDP and PIP/PAP as well as on the PEP-equipped smart device. Furthermore, two different usage scenarios were investigated: In the first scenario, the client issued 100 concurrent requests to invoke the services, allowing the investigation of the performance under heavy load conditions. The results appear in Figure 56.



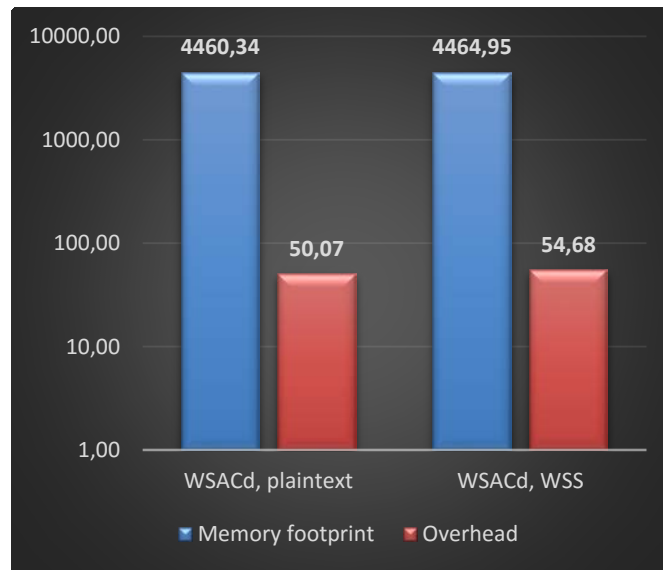
**FIGURE 56. CLIENT-SIDE RESPONSE TIME FOR 100 CONCURRENT REQUESTS (IN MS)**

Investigating a second use case scenario, we set the benchmark client to issue 20 requests, one every 3 minutes, emulating more realistic usage conditions in the context of a smart home environment. The results of the above assessments in terms of the average response time (i.e. the time the user has to wait before she receives the data she intended to access) are depicted in Figure 57. In both usage scenarios, the overhead of the access control mechanism are considered acceptable. The impact of the WSS protection is significant in cases of infrequent requests (as in the second scenario, where the connections close between the request timeouts and, thus, have to be reinitiated).



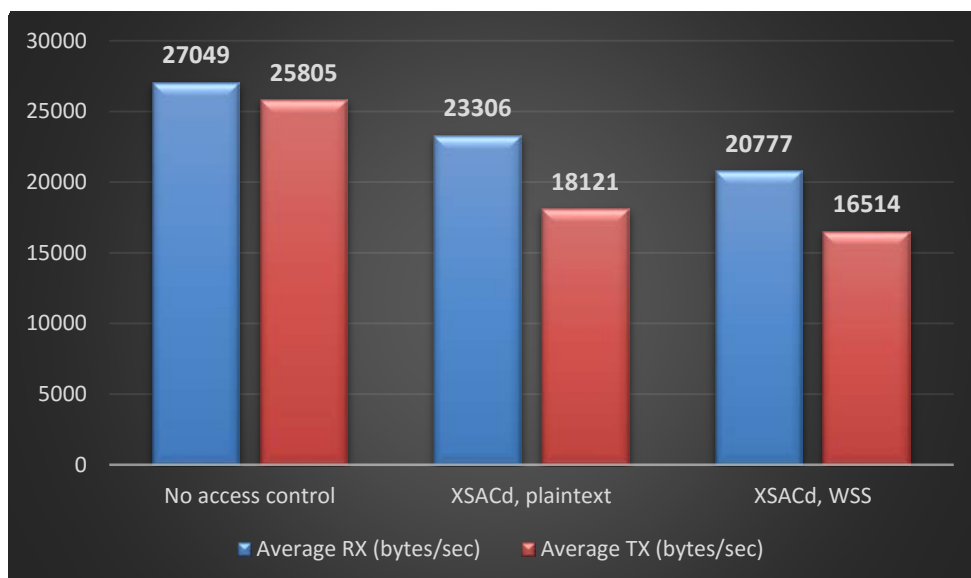
**FIGURE 57. AVERAGE CLIENT-SIDE RESPONSE TIME (IN MS) FOR THE INVESTIGATED DEPLOYMENTS AND USAGE SCENARIO.**

In terms of the resources consumed on the target, PEP-protected device, and focusing on the most demanding scenario (i.e. concurrent requests), profiling indicated a mild footprint during tests, even in the case of the relatively resource-constrained smart platform used in this setup. Average memory consumption is presented in Figure 58, where the overhead of the access control mechanisms appears trivial compared to the simpler DPWS device.



**FIGURE 58. MEMORY FOOTPRINT (IN KB, LOGARITHMIC SCALE) ON THE PEP-PROTECTED DEVICE, INCLUDING THE OVERHEAD COMPARED TO THE SIMPLE DPWS DEVICE WITH NO ACCESS CONTROL PROTECTION.**

The average CPU load was inversely proportional to the client response times (depicted in Figure 57); when the device has to wait for a reply from the framework (i.e. the PDP) before serving the client, its CPU load is expectedly lower. The recorded values were 11.6%, 9.3% and 8.4% for no access control, XSACd and XSACd with WSS respectively. The same ranking was also documented when monitoring average transmission (TX) and reception (RX) rates on the target device – see Figure 59. The most taxing scenario network-wise was that of the device with no access control, but in all cases the data rates were relatively low, with the lowest recorded value being 16.13kB (average TX of XSACd, WSS device) and the highest being 26.5kB/sec (average RX of DPWS device without access control).



**FIGURE 59. NETWORK THROUGHPUT ON TARGET DEVICE DURING TESTS.**

The number of stored policies may significantly affect the performance of the access control system, to the point where the response time overhead becomes prohibitive [57]. This was

taken under consideration during development and, thus, XSACd's PIP stores policies in the form of a hash table residing in memory. The effectiveness of this approach was validated during benchmarks: increasing the number of stored policies from 50 to 500 increased the response time by 7,37% in the first scenario (i.e. consecutive requests) and by just 1,7% in the more realistic second scenario (i.e. a request every 3 minutes).

Finally, to assess the load that the DPWS proxy might impose on the device it will be deployed on, the proof-of-concept implementations of the proxies were deployed on two Beagleboard-xM embedded devices (1GHz ARM Cortex-A8 processor, throttled at 600MHz during testing, 512MB DDR2 RAM, a minimal Linux-based operating system). Each of the platforms resided on a different geographical area and they communicated via the Internet. Even though these test-bed platforms were relatively resource-constrained, their CPU load during testing was minimal (below 10% when handling messages, otherwise idle). The memory footprint of the application is relatively stable at around 16MB, of which around 6,6MB are occupied by the various libraries used in its implementation.

The MQTT Broker used was just a standard implementation of Mosquitto [149], thus assessing its performance is beyond the scope of this work

### 5.3.8 SUMMARY

This section presented XSACd, an adaptation of uSPBM for smart homes and other smart environments, often residing across different networks. The intrinsic requirements of the smart home environment, its users and the often resource-constrained nature of its devices fundamentally affected the choice and implementation of the mechanism that form the basis of this work. Thus, XSACd's entities are platform-agnostic, lightweight and interact seamlessly, minimizing the home users' involvement in deploying, setting up and maintaining the system. The proxies responsible for the communication across different domains, require no interaction on behalf of the users, offering automated discovery and interactions with the target devices across the Internet.

## 5.4 SMART VEHICLES – THE RTVMF FRAMEWORK

Smart vehicles will be an important segment of the imminent Internet of Things (IoT) –enabled world, where computing devices will permeate our lives and will even be easy to design and create at home [150]. Modern vehicles already feature a number of embedded electronics that monitor and control their subsystems, to enhance passenger comfort and safety, achieve energy-efficient operation and maximize vehicle lifetime. Superior safety features can help avoid many accidents, and they are the focus of various governmental initiatives worldwide, which define stricter regulations (such as the COMMISSION DIRECTIVE 2008/89/EC enforcing daytime running lights in new vehicles). This push is expected to intensify, to benefit from Intelligent Transportation Systems (ITS), prompt emergency response and advanced features like early braking and road lane departure warnings, e.g. as indicated by the “Policy orientations on road safety 2011-20” European Union (EU) program [151]. Automotive legislation also necessitates the production of more eco-friendly vehicles, a target partly achieved by subsystems monitoring the vehicles' operation in real-time, triggering adjustments to engine parameters. Based on the above stimuli, the integrated electronics increase with every vehicle generation, and are expected to rise steeply with the introduction

of smart and, eventually, self-driving vehicles. This “intelligence” will also enable a variety of novel services that everyone will enjoy, from end-users (e.g. parents lending the family vehicle to their teenager) to private and public entities operating vehicle fleets (logistics, car-rental, governments, rescue services etc.).

To facilitate the deployment and operation of these services in a secure and interoperable manner, we merge uSPBM with an agent-based reasoning system that allows us to monitor various operational parameters, which are assessed using security, privacy and dependability –related metrics, enabling real-time monitoring and interaction with a smart vehicle or a smart vehicle fleet. This combination of technological building blocks, produces RtVMF, a smart vehicle management framework based on the integration of novel primitives with standardized technologies. The proposed approach can provide real-time management of vehicles, aggregating and quantifying various operating parameters (e.g. engine health) using a set of security, privacy and dependability metrics. The owner and other stakeholders can set policies of fair use (e.g. maximum speed), tracking the driver’s behavior, and informing emergency services when an accident occurs. Thus, it allows individuals and companies or public entities relying on vehicle fleets for day-to-day operations to minimize the risks associated with passenger safety, protect vehicle investments, enhance productivity, and reduce transportation and staff costs.

To validate the feasibility of our approach, a working prototype of the framework was developed and deployed on real vehicles. Moreover, we carried out a performance evaluation on a test-bed consisting of devices expected to be found on smart vehicles and a typical backend infrastructure (i.e. a command and control center), to investigate the performance overhead of RtVMF.

#### 5.4.1 MOTIVATION

A typical car may currently utilize over 80 built-in microprocessors, providing advanced safety systems, emission monitoring and in-car commodities [152] which aim to enhance passenger comfort and safety, also protecting the vehicle’s subsystems by providing early warning of failures and/or adjusting their operation accordingly. Typically, electronic control units (ECU) manage and interconnect the distinct systems [153], and the infotainment infrastructure provides enhanced facilities, like navigation, to passengers [154]. Newer vehicle generations will take this further, supporting communication with other vehicles, the city infrastructure and backend systems. Such prototype deployments are already under assessment in the EU and the United States (USA); e.g. the UMTRI Safety Pilot [155].

Real-time monitoring of the vehicle’s state and the driver’s behavior will allow public entities, logistics organizations and other businesses to minimize the vehicle investment risks and promote strategies for increasing productivity and safety while reducing transportation and staff costs. Government regulations are decisive motivators of pertinent research efforts. The United Kingdom aims to minimize road deaths in business-owned vehicles; starting in 2008, road death is considered an unlawful killing, enabling seizing of the company’s records and bringing prosecutions against directors who fail to enforce safe driving policies. Therefore, fleet management is now imperative for organizations owning a significant amount of vehicles.

The European Commission also defines new regulations for vehicle safety, such as the eCall system [156], which will become mandatory for every vehicle moving in the European Union by 2018. This emergency service dictates that, when an accident occurs, the vehicle should automatically relay essential information (its location, its direction and speed before the crash, number of passengers etc.) to appropriate Public Safety Answering Points (PSAP). By providing early notification and allowing efficient coordination of the emergency services, it is expected to decrease the response time to such incidents by 50% in rural and 40% in urban areas, drastically reducing the number of deaths and the severity of injuries for the thousands of people involved in road accidents every year [151].

However, security-related incidents are a tangible threat, as exploiting vulnerabilities in the infotainment infrastructure can allow the remote control of vehicle components; an attacker can control turn off the lights or even control the brakes while on the move [8]. As more vehicles feature seamless connection to Internet, urges engineers to consider security at the design phase. To this end, there is a need for systematic methodologies for developing secure and efficient vehicular embedded systems.

Researchers of the Ford motor company have presented a methodology for modeling automotive systems in terms of Security, Privacy, Usability and Reliability (SPUR [157]). Evaluated components are analyzed based on the offered SPUR functionality, assigning a qualitative value (low, medium or high) to each parameter. They apply this method on real system attributes, such as the valet key and the anti-lock braking system.

The EU-funded project nSHIELD [158] proposed two quantitative methodologies for building secure embedded systems in terms of Security, Privacy and Dependability (SPD): the attack surface metric and the multi-metric methodologies. The latter, a key RtVMF component, was demonstrated in social mobility applications, while the former was demonstrated in avionics and railway applications.

Finally, any novel components should be compatible with existing vehicles; it is estimated that there are over 500 million vehicles already roaming US and EU roads alone [159], [160], indicating a huge market for anyone involved in retrofitting such modules.

RtVMF aims to address the above issues. The principal concern is the enhancement of passenger safety and the framework can enable this in various ways: Mechanisms are included to allow the vehicle owner (e.g. a logistics company or a father lending his car to his son) to specify driving rules, such as maximum speed and/or the operating area (through geo-fencing). Furthermore, sensors can monitor vehicle health, informing stakeholders about engine malfunctions or emergencies in real-time. Thus, the adoption of RtVMF can allow any public or private organization with vehicle fleets to reduce the risks to their personnel and their vehicle investment, advancing productivity while reducing transportation and staff costs. Moreover, it can help achieve compliance with upcoming regulations, both for vehicle safety and green infrastructure management.

#### 5.4.2 THE RTVMF ARCHITECTURE

RtVMF consists of various components, comprising a low-cost multi-agent system for smart vehicles, also allowing the integration of smart city infrastructures into the monitoring and decision-making process, as depicted in Figure 60.

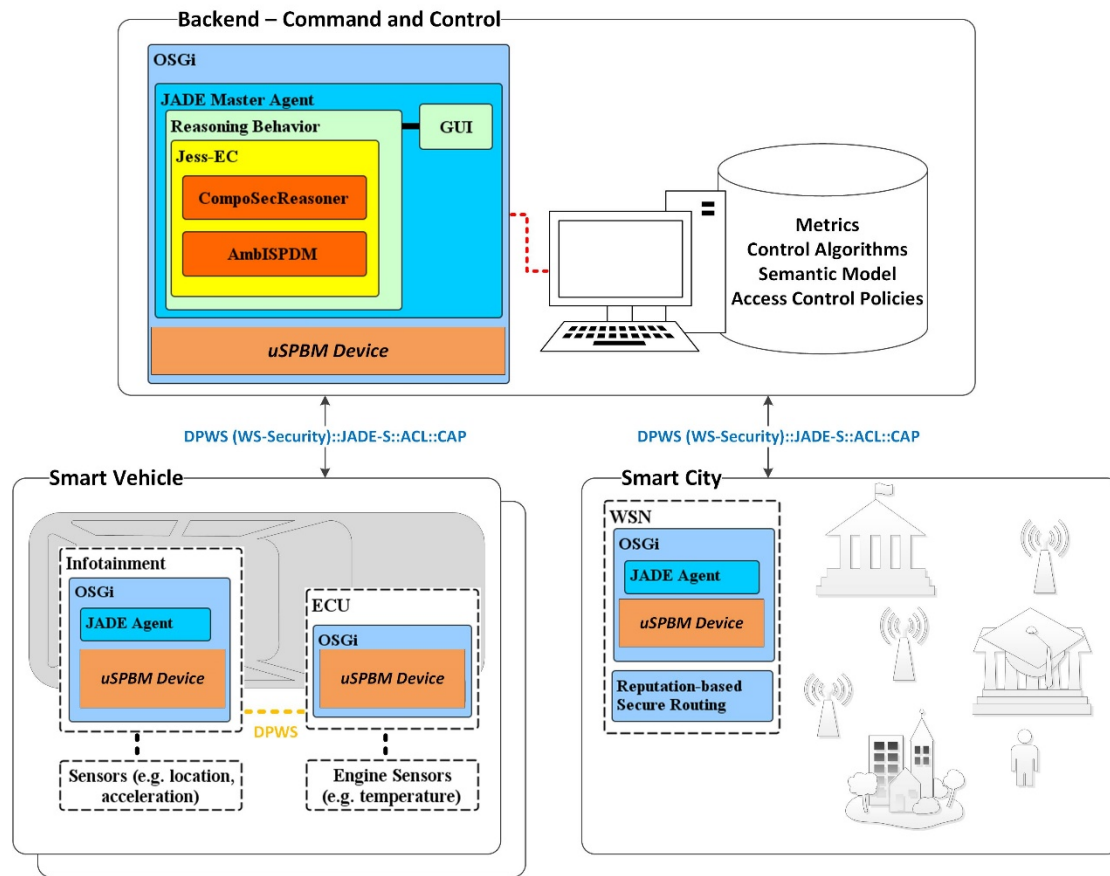


FIGURE 60. RTVMF ARCHITECTURE AND DEMONSTRATOR DEPLOYMENT

The following sections describe the main building blocks of RtVMF, which adopts a SOA approach for the underlying communications, providing seamless interaction between the framework's entities. All entities specify semantic information (e.g. their type and provided services) using DPWS, supporting device and service description, discovery, messaging and eventing. This allows the deployment of devices aligned with Web Services technologies, facilitating interoperability among services provided by resource-constrained device and providing seamless Machine-to-Machine (M2M) interactions.

#### 5.4.2.1 THE RTVMF AGENTS

The multi-agent system is implemented using the Java Agent Development framework (JADE), which supports all agent deployment standards defined by the Foundation for Intelligent Physical Agents (FIPA), including the Agent Communication Language (ACL). Event Calculus (EC) forms the basis of the reasoning process, using the Jess (Jess-EC) rule engine for the implementation. We model the ambient context in Jess-EC and develop reasoning services based on a formal theory that reasons about the composability and integration of the underlying devices and technologies, verifying the system's current SPD level. An ambient environment management theory is responsible for administrating the system in real time. The Common Alerting Protocol (CAP), also an OASIS standard, models the semantic



information exchanged between agents. Agents transform the CAP alerts into Jess-EC events (and vice-versa), triggering the reasoning process. The external communications rely on GSM. The RT-SPDM system [136] is a preliminary version of the above.

At the smart vehicle end, an RtVMF agent monitors the vehicle and is responsible for a minimal set of decisions (regarding security, privacy, dependability and safety), to avoid unnecessary backend communications, to safeguard continuity of operations when network connectivity is not available and to avoid unnecessarily transmitting sensitive information. The agent runs on the Infotainment system, communicating internally with the ECU and externally with other agents, acting as an intermediate layer between the vehicle and external entities.

The Command & Control (C&C) center is responsible for smart vehicle management. A master RtVMF agent, deployed at the backend, collects information from smart vehicles and infrastructure agents. The master agent has global knowledge of the system and carries out all computationally intensive tasks, implementing the fleet management strategy. An extension of the backend could use databases or cloud services to store vehicle and driver history records, maintaining log files and execution reports. Moreover, the positioning information of the smart vehicles can be display on a geographic information system (GIS) to provide location-based services (e.g. geo-fencing).

To implement the above functionality, we integrated DPWS along with the JADE agents into the Open Service Gateway initiative (OSGi) – a standardized middleware that constitutes a module system and service platform. ECU and supplementary vehicle devices model the provided functionality in DPWS and exchange information with the vehicle’s agent through OSGi.

#### 5.4.2.2 *SPD METRICS*

At the smart city infrastructure or service provider end, the master RtVMF agent monitors the vehicles, and communicates information to enforce SPD and safety plans, based on an ambient intelligence-based, decision making mechanism. We adopt the nSHIELD [158] multi-metric methodology for modelling and measuring the SPD features of each entity and the system as a whole.

The multi-metric methodology is based on an analysis of a system’s individual components and determines triple vectors representing the  $\langle S, P, D \rangle$  features, based on the evaluation of corresponding metrics. Each measurable parameter takes a value from a set, mapped in a range from 0 to 100. These values represent no protection to optimum protection respectively, based on a preceding security analysis. The individual SPDs are also composed to form the overall system SPD ( $SPD_{System}$ ). The system is configured at runtime to achieve a targeted SPD level ( $SPD_{Goal}$ ). The above enable the SPD analysis of individual elements and of their combination, allowing stakeholders to inspect each component and the composed system as a whole, evaluating the effectiveness of different configurations and associated mechanisms in terms of achieving the desired  $SPD_{Goal}$ . This approach eases human monitoring and operation of the system, provided the backend application development considers human-computer interaction principles.

More details on the multi-metric methodology appear in Garitano et al. [161], along with a smart vehicle use case. Its assessment includes three scenarios with nine different configurations. Metrics evaluate features like encryption and GPRS/SMS communication, enhancing security and inspecting driving behavior. The use case demonstrates the simplicity of the solution and its scalability, with an equally effective evaluation of simple and complex systems.

With regard to the dynamic SPD levels, ideal security, privacy and dependability settings may not always be possible, as congestion may hamper operation by congestion, missing mobile network coverage or other variables. So, for example, in situations where low latency is essential (e.g. V2V communications), a vehicle may resort to simpler cryptographic mechanisms, resulting in a lower Security level. In another case, the driver may decide not to report his exact location, but a larger area, to obstruct tracking of her movement and protect her privacy. This will have a positive effect on the Privacy level of the specific vehicle, as reported at the backend system. Variations in the Dependability level will be more common. The vehicle's health is monitored in real time (tire pressure, ECU warning messages etc.), directly affecting its Dependability level. E.g., when the car exceeds the planned travel mileage between services, the reported Dependability value will be lower.

#### 5.4.2.3 SECURITY

The protection of messages exchanged between RtVMF entities is of critical concern, as a violation of the framework's security could give an erroneous view with regard to the actual vehicle (or vehicle fleet) situation, cause unnecessary trouble for emergency response services or even endanger the safety of vehicle occupants.

To this end, the framework protects the communications of the various entities (managed vehicles, the backend, and users, such as vehicle owners) with the WS-Security standard. The latter, typically used alongside DPWS, provides end-to-end security, with message encryption to provide confidentiality, but also digital signatures to assure integrity and provide non-repudiation. Security tokens can also be attached to establish the sender's identity and, as various security token are supported (e.g. SAML, Kerberos, X.509), it can trivially be integrated into existing Vehicular PKI (VPKI) infrastructures. An added benefit is that entities without proper credentials cannot discover the services hosted by a smart vehicle agent, even if they have access to the same network.

While WS-Security safeguards all exchanged messages, to protect the ACL messages exchanged by agents, the implementation also includes the JADE-S add-on, which extends JADE with the Java Authentication and Authorization Service (JAAS), Java Cryptography Extension (JCE) and Java Secure Socket Extension (JSSE) mechanisms. This allows us to authorize agent actions against agent permissions residing in policy files containing information regarding the agent types and the valid actions that they can perform. Moreover, it is used to verify the integrity of the received reasoning data (e.g. when a vehicle informs of its updated SPD state), which is one of the most critical aspects in the framework's operation.

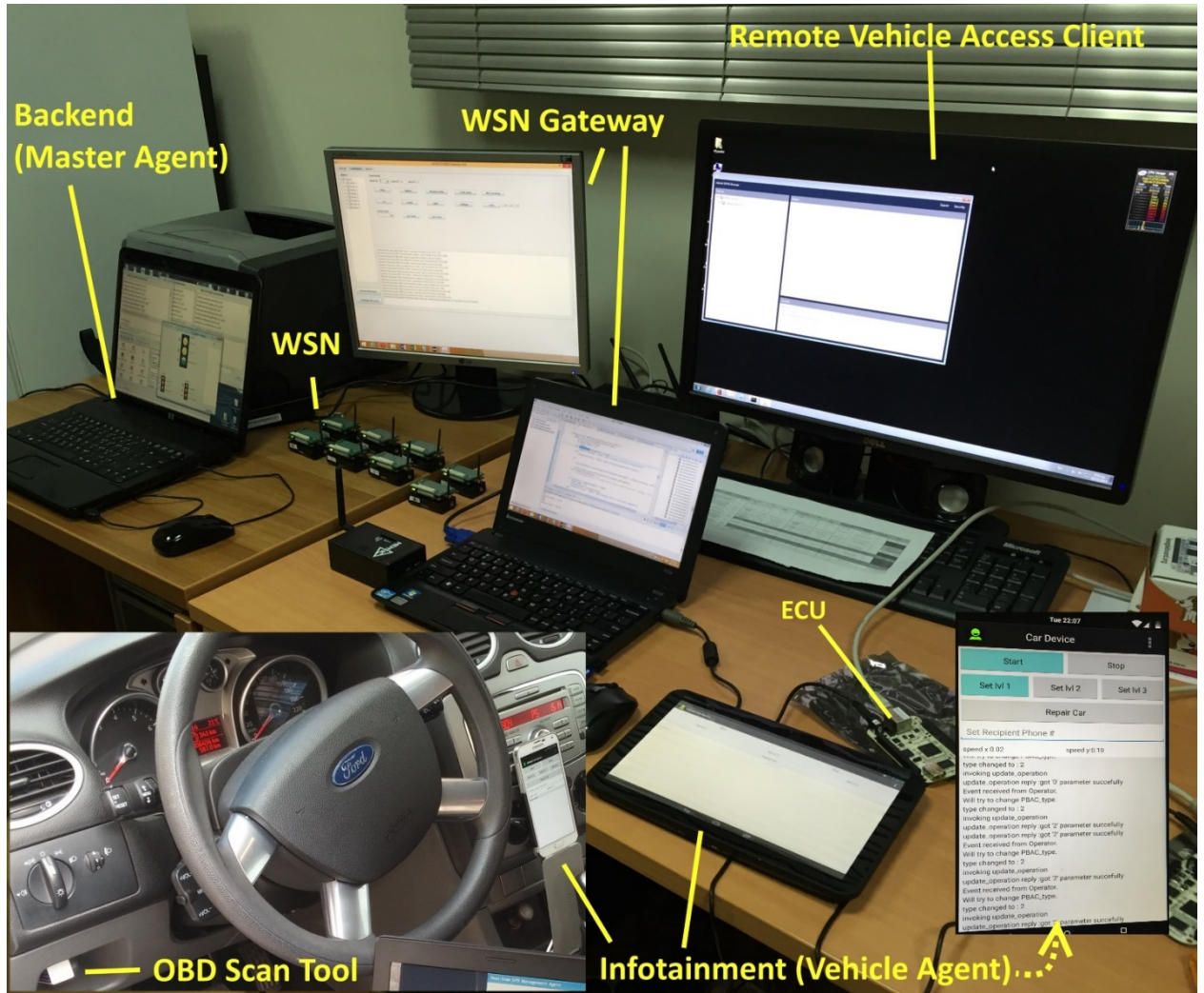
Moreover, the security features built into OSGi offer inner-platform security on both agents and managed devices. This allows us to control which bundles can be started/stopped, when,

by whom etc., based on pre-defined rules, thus limiting bundle functionality and providing an extra layer of protection, even in cases where a malicious entity manages to gain access to the software platform itself, aiming to disrupt the service.

RtVMF also integrates the strong access control provided by uSPBM, allowing its operators to manage access to the resources of smart vehicles in a centralized manner. The policy-based access control mechanisms of uSPBM exploit and extend the DPWS functionality of the framework's devices, to implement the necessary communications and entities (e.g. policy enforcement points on vehicles). Thus, a multifunctional RtVMF-enabled vehicle may feature various hosted services (a location service, a fuel consumption service etc.), access to which will be controlled based on the active policy set. Such an example could be a logistics company that allows drivers to track their designated vehicles' locations through their mobile phone: the company can simply modify the active policies to give a specific individual access to a vehicle's location service; after returning the vehicle, a policy reset can withdraw the driver's access privileges.

#### 5.4.3 PROOF OF CONCEPT

As a proof of concept, we retrofitted RtVMF onto an existing vehicle, using off-the-shelf components. The setup relies on the infotainment system – or the user's smart phone, if a smart infotainment system is not available, as was the case with the vehicle we used – for communication with the backend system. An Android-based application runs on the smart device, performing the vehicle's basic reasoning process, also communicating and receiving commands from the C&C. The application collects information regarding the car's sensors (e.g. fuel consumption) from the ECU via a widely available and affordable Bluetooth-enabled OBD scan tool (model ELM327 Bluetooth OBD-II, cost of ~10 Euros). The application collects additional data from sensors integrated into infotainment, namely acceleration and GPS position. The insert in Figure 61 depicts the above car setup.



**FIGURE 61. TEST-BED USED FOR DEMONSTRATION & PERFORMANCE EVALUATION. PROOF OF CONCEPT RTVMF RETROFITTING (INSERT)**

RtVMF can also incorporate a smart city's infrastructure and, as proof of this, we include a wireless sensor network (WSN) in the test setup, emulating e.g. environmental or traffic sensors spread across a smart road. The sensors communicate via a reputation-based secure routing protocol (presented in Hatzivasilis et al. [162]), enabling them to detect several types of sensor malfunctions and malicious attacks, guaranteeing network service continuation and resilience. The backend collects and processes information from the WSN's gateway, including it into the reasoning process.

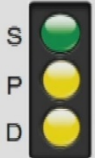
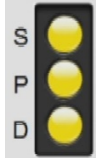
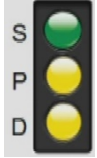
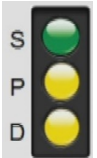

#### 5.4.3.1 DEMONSTRATION SCENARIO


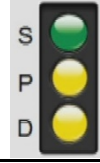
We demonstrate RtVMF using a smart city scenario that includes the WSN-equipped smart city infrastructure described above, a backend C&C and two smart vehicles with various sensing capabilities. The scenario architecture appears in Figure 60.

The use case scenario aims to demonstrate that the RtVMF backend is automatically informed of all changes in the various entities and their subcomponents, composing a new SPD state

based on these updated metrics. The set of countermeasures are able to mutually support each other when the system has to react to a particular incident (attack, emergency etc.), maintaining acceptable SPD levels, while conserving resources (if needed), also informing and aiding the response of the various stakeholders (vehicle owner, emergency services etc.). The scenario steps appear in detail in Table 9.

TABLE 9. SCENARIO STEPS

STEP	Events	Effect	Overall (S,P,D)	State Visualization
1	Power-on of all systems and discovery/registration	Initial State	<b>80,70,65</b>	
2	The WSN detects a Black Hole attack, which causes the Security level to decrease, indicating malicious activity.  MA is informed through WSN that an attack occurs and it sends a command to the vehicles to increase security.	Security level decreases	<b>60, 70, 65</b>	
3	Security level is increased on both Car 1 & Car 2, as they now use stronger encryption to better safeguard V2V and V2Backend communications in the context of a potentially compromised infrastructure.  MA is informed.	Security level increases	<b>85,70,65</b>	
4	The WSN has counteracted the Black Hole attack, reporting the change to the MA. The Security level is now higher, as the malicious activity has been countered.  The MA asks Cars 1 & 2 to return to normal state (to conserve resources)	Security level returns to initial state	<b>80, 70, 65</b>	
5	ECU of Vehicle 1 reports increased engine temperature, affecting its Dependability level.  MA is informed.	Dependability level decreases	<b>80, 70, 50</b>	

6	<p>Car 1 crashes (lowering SA1 Dependability); eCall message transmission. C&amp;C now has access to exact vehicle location, as dictated by access control policies (lowering SA1 Privacy). Encryption is disabled to facilitate emergency response (lowering SA1 Security).</p> <p>The MA is informed and sends email to C&amp;C operators/administrators.</p>	S & P & D levels decrease	50, 50, 40	
7	<p>Car 1 is repaired, returning to normal SPD state.</p> <p>SA1 reports new state to MA.</p>	S & P & D levels increase	80,70,65	

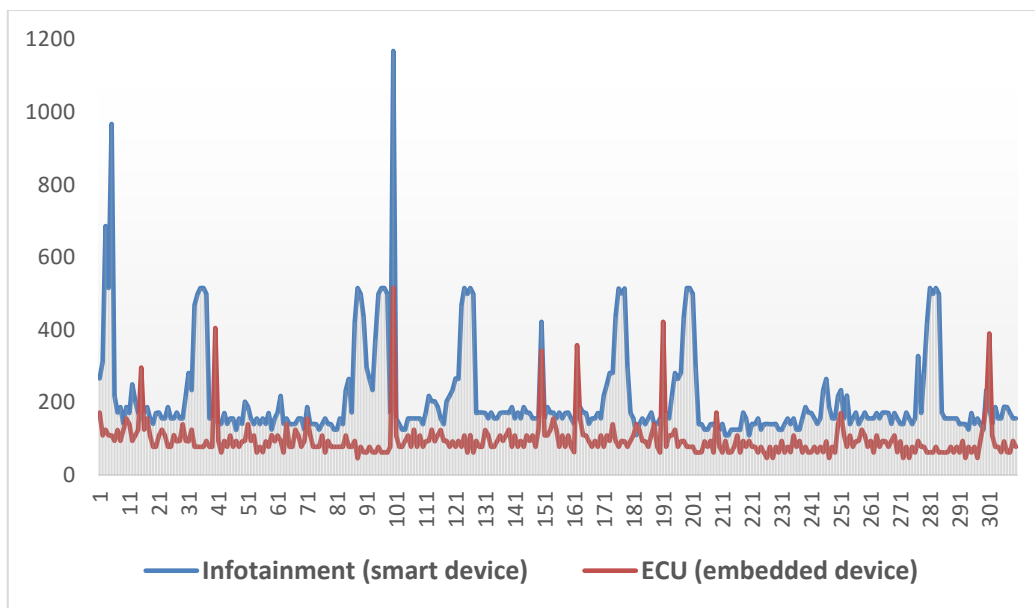
#### 5.4.3.2 PERFORMANCE EVALUATION

As a full-scale deployment was not feasible, an experimental test-bed was used to evaluate RtVMF. We deployed a Linux-based device, emulating the ECU, on a BeagleBone embedded device (720MHz ARM Cortex-A8 processor, 256MB RAM, Linux OS). The Android-based version of the agent was deployed on a tablet (1.9GHz quad-core processor, 2GB RAM, Android OS), emulating the vehicle's infotainment. The backend system run on a laptop PC, while a separate laptop was used as a gateway for the "smart city infrastructure" WSN which was based on IRIS motes (Atmel ATmega1281 16MHz CPU, 8kB RAM, 128Kb flash) running the reputation-based secure routing. Finally, a desktop PC was used as a client to the services provided by the vehicle (e.g. to access its location), for benchmarking purposes. This setup, also used to carry out all of the demonstration steps, appears in Figure 61.

We evaluated the performance of RtVMF focusing on the resource-constrained devices, as documenting all performance aspects of all entities involved in such complex framework would not be practical. The benchmark client issued consecutive requests to access the smart vehicle's location (from the infotainment device) and its engine temperature (from the ECU). Access control mechanisms protected both operations, thus, before replying, the devices communicated with the backend to verify that the client was authorized to access this data. Furthermore, at random intervals, the benchmark client would communicate with the master agent to trigger changes in the SPD state of the prototypes (e.g. to increase the key length used for encryption), thus evaluating the impact such changes can have on the responsiveness of the devices and the system as a whole.

The master agent is the most computationally demanding entity of RtVMF; its code size is 1.87MB, occupies 45MB RAM and needs 1.6 seconds, on average, to perform the reasoning process. Moving to other entities, we focused on the infotainment device (i.e. the android application) and the corresponding Linux-based application developed emulating the ECU. Their CPU load during tests was not that significant, with an average of 4.8% recorded on the Beaglebone and of 4.1% on the Android platform. Memory consumption was also acceptable,

averaging 33.46MB on the Beaglebone and 26MB on the Android device. The most interesting performance parameter is the delay experienced by the backend operator attempting to retrieve the smart vehicle's status and the latency of the system when changing between SPD states; also confirmed by relevant pilot programs, where stakeholders highlighted the need to have real-time or near real-time access to vehicles' operating parameters [155]. These results appear in Figure 62, which shows the response time (recorded client-side) for each of the requests issued concurrently to both the vehicle's infotainment (tablet) and its ECU (Beaglebone). The Linux-based embedded device is more responsive, with an average response time of 93.16ms compared to 198.7ms for the infotainment. This is reasonable, as there are more processes running at the background on the Android tablet, including the vehicle agent (which also features a GUI, further affecting its responsiveness). Nevertheless, both devices demonstrated acceptable response times. Also evident in the graph are the spikes recorded when the devices had to change SPD states (i.e. had to process the incoming master agent request, change their operating parameters accordingly and inform the master agent once the changes were in place). These SPD state changes do introduce significant delays, but they are not a concern, as they should be rare during normal operation (when the framework needs to react to some attack, when a vehicle crashes etc.).



**FIGURE 62. RESPONSE TIME (IN MS) PER REQUEST, FOR BOTH TEST PLATFORMS**

While the above results validate the proof-of-concept, realistic, larger-scale deployments should not be problematic either: the only potential bottleneck is at the backend, and especially the reasoning process at the master agent. Nevertheless, Jess-EC provides an efficient method for pattern matching. The computational complexity is linear in the working memory size and is of the order of  $O(RFP)$ , where  $R$  is the number of rules,  $P$  is the average number of patterns per rule left-hand-side, and  $F$  is the number of facts on the working memory. In our case, the implemented management theory consists of around 50 rules.  $P$  is kept small (in order of one to three) as each entity is assigned a unique identifier and the scenario events affect specifically declared components. Thus, only  $F$  affects the scalability of the reasoning process. Every new vehicle requires about 10 facts to be modelled, resulting in

low overhead (in the nanosecond range) to integrate its status to the system's total and less than 40 extra bytes of memory for the master agent.

#### 5.4.4 SUMMARY

This section presented RtVMF, a smart vehicle management framework based on uSPBM which provides real-time monitoring of vehicle fleets, including secure, dependable and privacy-aware monitoring of vehicles' operating parameters. The framework can maintain vehicle fleets in good condition by constant monitoring of vehicle health and drivers' compliance to good-driving practices, allowing prompt reporting to various events (e.g. an emergency), adapting accordingly, and also informing and aiding the response of emergency services.

Government and emergency services could monitor their fleet improving everyday workflow and vehicle assignments, fully exploiting vehicle investments. Private sector entities operating in transportation could also benefit by adopting the proposed framework. Bus and taxi companies could reduce the transportation time, designing and adopting better strategies to satisfy their passengers' expectations, e.g. by better scheduling trips, thus providing quick and timely service.

Moreover, it can help achieve compliance with governmental regulations but also helps build solid business cases. These involve retrofitting the large number of existing vehicles, but also enabling novel services. Such services may include enhanced road insurance services, adjusting fees based on user driving behavior, mileage, vehicle condition etc., while safeguarding users' privacy by, e.g., ensuring their preferences are not violated through strong policy enforcement mechanisms.



## 6. CONCLUSIONS & FUTURE WORK

Ubiquitous devices in the form of resource-constrained interconnected embedded systems are constantly gaining popularity. Such devices nowadays offer real-time accessibility and have the capability to provide various services to remote parties. 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, networks of devices manage well-defined services and process information that require protection from malicious entities attempting to perform unauthorized access and modifications. The information they manage might be safety critical, sensitive, or confidential, and therefore constitute an attractive target. Their protection necessitates the deployment of robust mechanisms that will, among others, control access to participating nodes' resources and prevent data breaches. This is not a trivial task, considering the resource limitations of some of the nodes that prohibit the deployment of computationally 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 thesis addresses the above requirements and defines a policy-based management mechanism based on eXtensible Access control Markup Language (XACML, [22]) policies. It provides serving nodes the ability to control access to their resources based on policy constraints set by the system owner. Considering that managed nodes 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 service oriented nature of the proposed scheme is accomplished by the use of the Devices Profile for Web Services (DPWS, [23]), 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. The implementations of these compact web services are based on a novel set of libraries, Node.DPWS, which outperform the existing platform-agnostic tools available to developers.

The proposed architecture can be considered a generic policy-based management model that can be easily adapted to meet certain requirements of various ubiquitous 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).

Combining XACML with DPWS, we aim to tackle two of the most important IoT barriers, as recognized by end-users, researchers and business stakeholders alike: lack of interoperability and the associated difficulty of device management, along with the lack of fine-grained, context-aware access control. With regard to the latter, uSPBM does indeed offer fine-grained, context-aware authentication & authorization for users, devices & services, while the former (i.e. interoperability) is guaranteed through the standardized communication technologies adopted. This also allows us to exploit pre-existing work on these technologies (e.g. specifications to adapt XACML policies to different domains and the successful use of SOAs in a variety of areas). Facilitated by the underlying design choices, uSPBM's entities are platform-agnostic and interact seamlessly, minimizing involvement in deploying, setting up and maintaining the system.

The framework is also extremely flexible and modular, operating with a different subset of modules needed for each scenario (e.g. some management or authentication features may be needed in one scenario, but not in another). The flexibility in terms of deployment options is enhanced by the presence of cross-domain proxies, which allow us to overcome the limitations of DPWS and provide seamless interactions between different domains and networks. Moreover, each entity type may be assigned more than one role, in scenarios where this is feasible, thus allowing the system to offload computationally demanding tasks to other infrastructure components.

Emphasis is also given on the security of the communicated request/response and service provision messages, as a critical issue is the protection of policy messages exchanged among the framework's entities. The mechanisms deployed for this task are closely dependent on the application requirements (e.g. the need for point-to-point or message-level confidentiality), as well as the potential support for more advanced security characteristics, such as node trust-sharing schemes and security context awareness. From a cryptographic point of view a scheme may seem far more efficient but the overall overhead may actually be quite significant and render the scheme impractical for large-sized networks. To this end, different security mechanisms have been analyzed and implemented, both symmetric and asymmetric in nature, each with its intrinsic characteristics. In any case, before making a decision, several parameters need to be considered in order to adapt the proposed framework to a specific application and its requirements.

Finally, efforts were made to produce proof-of-concept implementations of each of the framework's entities, to validate the feasibility of the proposed approach in various application domains (smart homes, e-health, smart vehicles) and on a number of heterogeneous hardware platforms (PCs/laptops, embedded devices, smartphones/tablets and wireless sensor devices)

There are a variety of aspects to explore in order to improve the framework's features and its applicability to a variety of domains. Focusing on the authorization language, additional work should 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 ubiquitous computing deployments and their intrinsic requirements. While there are already specifications for adapting XACML to various areas (e.g. healthcare [141]) and specific requirements (e.g. privacy [163]), there are a variety of application areas brought forward by the IoT (agriculture, industry, smart homes etc.) that could also be explored. In this context, semantics are often utilized to provide context-awareness [164] and spatio-temporal factors have to be considered, due to the constant mobility of users and their devices [165], [166] with application-specific requirements (e.g. healthcare [74]). These efforts could lead to the definition of corresponding XACML specifications, potentially extending XACML policies to consolidate the requirements introduced in the new IoT reality.

In terms of the framework's communications, more lightweight DPWS implementations could be developed. This would be especially useful for scenarios involving extremely resource-constrained devices (e.g. sensors, as in [167]). Moreover, this would help further evaluate the proposed framework on alternative sensor platforms, to include devices less capable than the SunSPOTs. This work will have to be carried out concurrently with the investigation of lightweight cryptographic primitives appropriate for said devices and the communication mediums they typically use.

Some of the newer IoT-oriented protocols could also be investigated for the framework's underlying communications, comparing them to DPWS. Potential alternative standardized protocols include CoAP [168], MQTT [24] (already used for uSPBM's proxy communications) and XMPP [169], among others, also considering the alternative security mechanisms available to each protocol. The results of these efforts could also lead to the adoption of a hybrid, custom protocol, combining the advantages of the various already existing standardized solutions; even though such an approach would inevitably compromise the interoperability with existing standards.

Regardless of the application area where uSPBM will be deployed, stakeholders (e.g. business or homeowners) using the framework will be responsible for defining some parameters of the active policy set, depending on their requirements and preferences. Thus, an important aspect to be investigated is the provision of user-friendly interfaces for specifying access control policies, e.g. using a GUI with easy to use drop-down menus and tick boxes or having the user answer simple questions, automatically translating the user input to policies.

Finally, uSPBM will have to be tested on a larger scale, to confirm its effectiveness in actual IoT-scale deployments. This will be necessary in order to collect valuable feedback to improve key areas of uSPBM's operation, such as its scalability, and, most importantly, its ease of deployment, use and maintenance.

Along with the above, further research is required for establishing secure mechanisms tailored for ESs, in order to address potential threats to their secure operation, including those exacerbated by the intrinsic characteristics of the devices and their application fields, especially in the case of critical systems' applications. What is more, given the widespread

adoption of smart devices in our everyday lives (e.g. smart vehicles, smart houses, smart clothing), it is important to deal effectively with the inherent security concerns. The challenge lies with the researchers to put more effort in the above matters and come up with appropriate solutions, thus helping realize the promise of pervasive computing and the Internet of Things (IoT).

## 7. REFERENCES

- [1] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications," *IEEE Commun. Surv. Tutorials*, vol. 17, no. 4, pp. 2347–2376, Jan. 2015.
- [2] Intel, "A guide to the Internet of Things," 2015. [Online]. Available: <https://www-ssl.intel.com/content/www/us/en/internet-of-things/infographics/guide-to-iot.html>.
- [3] Cisco, "The Internet of Everything, Global Private Sector Economic Analysis," 2013. [Online]. Available: <https://www.cisco.com/web/about/business-insights/docs/ioe-economy-faq.pdf>.
- [4] K. Mayes and K. Markantonakis, "Information Security Best Practices," in *Secure Smart Embedded Devices, Platforms and Applications*, K. Markantonakis and K. Mayes, Eds. New York, NY: Springer New York, 2014, pp. 119–144.
- [5] Cisco, "The Internet of Everything (IoE) Value Index," 2013. [Online]. Available: [http://www.cisco.com/web/about/ac79/docs/innov/IoE-Value-Index\\_External.pdf](http://www.cisco.com/web/about/ac79/docs/innov/IoE-Value-Index_External.pdf).
- [6] Accenture, "Igniting Growth in the Consumer Technology," 2015. [Online]. Available: [https://www.accenture.com/\\_acnmedia/PDF-3/Accenture-Igniting-Growth-in-Consumer-Technology.pdf](https://www.accenture.com/_acnmedia/PDF-3/Accenture-Igniting-Growth-in-Consumer-Technology.pdf).
- [7] A. G. I. and J. V. Vidal, "Lights off! The Darkness of the Smart Meters," in *BlackHat Europe*, 2014.
- [8] K. Koscher, A. Czeskis, F. Roesner, S. Patel, T. Kohno, S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, and S. Savage, "Experimental security analysis of a modern automobile," in *IEEE Symposium on Security and Privacy (SP)*, 2010, pp. 447–462.
- [9] T. Bonaci, J. Herron, T. Yusuf, J. Yan, T. Kohno, and H. J. Chizeck, "To Make a Robot Secure: An Experimental Analysis of Cyber Security Threats Against Teleoperated Surgical Robots," *CoRR*, vol. abs/1504.04339, 2015.
- [10] J. Radcliffe, "Hacking Medical Devices for Fun and Insulin: Breaking the Human SCADA System," in *BlackHat USA*, 2011.
- [11] S. Erven and M. Collao, "Medical Devices: Pwnage and Honeypot," in *Derbycon Security Conference, Louisville, Kentucky, USA*, 2015.
- [12] S. Erven and A. Brand, "Medical Device Security: An Infectious Disease," in *Thotcon 2015, Chicago, IL, USA*, 2015.
- [13] A. Cui and S. J. Stolfo, "A quantitative analysis of the insecurity of embedded network devices," in *Proceedings of the 26th Annual Computer Security Applications Conference on - ACSAC '10*, 2010, p. 97.
- [14] HP, "Internet of Things Research Study," 2014. [Online]. Available: <http://h20195.www2.hp.com/V2/GetDocument>.
- [15] OWASP, "Internet of Things Top Ten Project," *Open Web Application Security Project*

- (OWASP). [Online]. Available: [https://www.owasp.org/index.php/OWASP\\_Internet\\_of\\_Things\\_Top\\_Ten\\_Project](https://www.owasp.org/index.php/OWASP_Internet_of_Things_Top_Ten_Project).
- [16] HP, "Internet of Things Security Study: Home Security Systems Report," 2015. [Online]. Available: <http://goo.gl/AzOcWp>.
- [17] A. Jacobsson, M. Boldt, and B. Carlsson, "A risk analysis of a smart home automation system," *Futur. Gener. Comput. Syst.*, Sep. 2015.
- [18] M. L. Mazurek, B. Salmon, R. Shay, K. Vaniea, L. Bauer, L. F. Cranor, G. R. Ganger, M. K. Reiter, J. P. Arsenault, J. Bresee, N. Gupta, I. Ion, C. Johns, D. Lee, Y. Liang, and J. Olsen, "Access control for home data sharing," in *Proceedings of the 28th international conference on Human factors in computing systems - CHI '10*, 2010, p. 645.
- [19] D. Barnard-Wills, L. Marinos, and S. Portesi, "European Union Agency for Network and Information Security (ENISA): Threat Landscape and Good Practice Guide for Smart Home and Converged Media," 2014.
- [20] A. Brush, B. Lee, and R. Mahajan, "Home automation in the wild: challenges and opportunities," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 2011, pp. 2115–2124.
- [21] ENISA, "Security and Resilience in eHealth Infrastructures and Services," 2015. [Online]. Available: [https://www.enisa.europa.eu/activities/Resilience-and-CIIP/critical-infrastructure-and-services/ehealth\\_sec/security-and-resilience-in-ehealth-infrastructures-and-services](https://www.enisa.europa.eu/activities/Resilience-and-CIIP/critical-infrastructure-and-services/ehealth_sec/security-and-resilience-in-ehealth-infrastructures-and-services).
- [22] B. Parducci, H. Lockhart, and E. Rissanen, "eXtensible Access Control Markup Language (XACML) Version 3.0," *OASIS Standard*, 2013. [Online]. Available: <http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-cs-01-en.pdf>.
- [23] D. Driscoll, A. Mensch, T. Nixon, and A. Regnier, "Devices profile for web services, version 1.1," *OASIS*, 2009. [Online]. Available: <http://docs.oasis-open.org/ws-dd/dpws/wsdd-dpws-1.1-spec.pdf>.
- [24] A. Banks and R. Gupta, "MQTT Version 3.1.1," *OASIS Standard*, 2014. [Online]. Available: <http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/mqtt-v3.1.1.pdf>.
- [25] C. Röcker, M. Ziefle, and A. Holzinger, "From Computer Innovation to Human Integration: Current Trends and Challenges for Pervasive HealthTechnologies," 2014, pp. 1–17.
- [26] H. A. Kielland Aanesen and J. Borras, "eHealth: The future service model for home and community health care," in *2013 7th IEEE International Conference on Digital Ecosystems and Technologies (DEST)*, 2013, pp. 172–177.
- [27] S. Tennina, E. Kartsakli, A. Lalos, A. Antonopoulos, P.-V. Mekikis, M. Di Renzo, Y. Zacchia Lun, F. Graziosi, L. Alonso, and C. Verikoukis, "WSN4QoL: Wireless Sensor Networks for quality of life," in *2013 IEEE 15th International Conference on e-Health Networking, Applications and Services (Healthcom 2013)*, 2013, pp. 277–279.
- [28] J. Sermersheim, "Lightweight Directory Access Protocol (LDAP): The Protocol," *The Internet Engineering Task Force*. pp. 1–69, 2006.
- [29] P. Rashidi and A. Mihailidis, "A survey on ambient-assisted living tools for older adults,"

- IEEE J. Biomed. Heal. Informatics*, vol. 17, no. 3, pp. 579–590, 2013.
- [30] G. Mulligan, “The 6LoWPAN architecture,” in *Proceedings of the 4th workshop on Embedded networked sensors - EmNets '07*, 2007, p. 78.
  - [31] Z. Alliance, “Zigbee Specification,” *Zigbee Alliance website*, pp. 1–604, 2008.
  - [32] Z-Wave, “About Z-Wave,” *z-wave.com*, 2015. [Online]. Available: [http://www.z-wave.com/z-wave\\_benefits](http://www.z-wave.com/z-wave_benefits).
  - [33] S. Bluetooth, “Specification of the Bluetooth system,” *Core, version*, vol. 1, pp. 2005–10, 2005.
  - [34] B. P. Crow, I. Widjaja, L. G. Kim, and P. T. Sakai, “IEEE 802.11 Wireless Local Area Networks,” *IEEE Commun. Mag.*, vol. 35, no. 9, pp. 116–126, 1997.
  - [35] H. Bohn, A. Bobek, and F. Golatowski, “SIRENA - Service Infrastructure for Real-time Embedded Networked Devices: A service oriented framework for different domains,” in *International Conference on Networking, International Conference on Systems and International Conference on Mobile Communications and Learning Technologies (ICNICONSMCL'06)*, 2006, vol. 43, no. 1, pp. 43–43.
  - [36] F. Jammes, A. Mensch, and H. Smit, “Service-oriented device communications using the devices profile for Web services,” in *Proceedings - 21st International Conference on Advanced Information Networking and Applications Workshops/Symposia, AINAW'07*, 2007, vol. 2, pp. 947–955.
  - [37] “Service-Oriented Device & Delivery Architecture (SODA).” [Online]. Available: <https://itea3.org/project/SODA.html>.
  - [38] “Service-Oriented Cross-layer Infrastructure for Distributed Smart Embedded Devices (SOCRADES).” [Online]. Available: [http:](http://)
  - [39] S. Karnouskos, D. Savio, P. Spiess, D. Guinard, V. Trifa, and O. Baecker, “Real-world Service Interaction with Enterprise Systems in Dynamic Manufacturing Environments,” in *Artificial Intelligence Techniques for Networked Manufacturing Enterprises Management SE - 14*, L. Benyoucef and B. Grabot, Eds. Springer London, 2010, pp. 423–457.
  - [40] “MORE Project.” [Online]. Available: <http://www.ist-more.org/>.
  - [41] J. Schmutzler, U. Bieker, and C. Wietfeld, “Network-Centric Middleware Supporting Dynamic Web Service Deployment on Heterogeneous Embedded Systems,” in *14th International Conference on Concurrent Enterprising*, 2008.
  - [42] IMC-AESOP, “IMC-AESOP Project Overview,” *Project Description*, 2011. [Online]. Available: <http://imc-aesop.eu/html/ProjDes.html>.
  - [43] “Web of Objects (WOO), ITEA 2 Project.” [Online]. Available: <http://www.web-of-objects.com/>.
  - [44] P. Spieß and S. Karnouskos, “Maximizing the business value of networked embedded systems through process-level integration into enterprise software,” in *2007 2nd International Conference on Pervasive Computing and Applications, ICPCA'07*, 2007, pp. 536–541.

- [45] S. Mitropoulos and C. Douligeris, "The impact of Service-Oriented Architecture (SOA) technologies in global market-oriented enterprises," *Int. J. Appl. Syst. Stud.*, vol. 4, no. 1/2, p. 106, 2011.
- [46] C. Y. Leong, A. R. Ramli, and T. Perumal, "A rule-based framework for heterogeneous subsystems management in smart home environment," *IEEE Trans. Consum. Electron.*, vol. 55, no. 3, pp. 1208–1213, Aug. 2009.
- [47] T. Perumal, A. Ramli, and C. Leong, "Interoperability framework for smart home systems," *IEEE Trans. Consum. Electron.*, vol. 57, no. 4, pp. 1607–1611, Nov. 2011.
- [48] A. Sleman and R. Moeller, "SOA distributed operating system for managing embedded devices in home and building automation," *IEEE Trans. Consum. Electron.*, vol. 57, no. 2, pp. 945–952, May 2011.
- [49] T. Bray, "RFC7159: The JavaScript Object Notation (JSON) Data Interchange Format," *Internet Engineering Task Force (IETF) Request for Comments*, 2014. [Online]. Available: <http://tools.ietf.org/html/rfc7159>.
- [50] J. Bourcier, C. Escoffier, and P. Lalanda, "Implementing Home-Control Applications on Service Platform," in *2007 4th IEEE Consumer Communications and Networking Conference*, 2007, pp. 925–929.
- [51] R. R. Igorevich, P. Park, J. Choi, and D. Min, "iVision based Context-Aware Smart Home system," in *The 1st IEEE Global Conference on Consumer Electronics 2012*, 2012, pp. 542–546.
- [52] V. Venkatesh, V. Vaithayana, P. Raj, K. Gopalan, and R. Amirtharaj, "A Smart Train Using the DPWS-based Sensor Integration," *Res. J. Inf. Technol.*, vol. 5, no. 3, pp. 352–362, Mar. 2013.
- [53] T. Cucinotta, A. Mancina, G. F. Anastasi, G. Lipari, L. Mangeruca, R. Checcozzo, and F. Rusina, "A Real-Time Service-Oriented Architecture for Industrial Automation," *IEEE Trans. Ind. Informatics*, vol. 5, no. 3, pp. 267–277, Aug. 2009.
- [54] S. Pöhlse, S. Schlichting, M. Strähle, F. Franz, and C. Werner, "A DPWS-Based Architecture for Medical Device Interoperability," in *World Congress on Medical Physics and Biomedical Engineering, September 7 - 12, 2009, Munich, Germany SE - 22*, vol. 25/5, O. Dössel and W. Schlegel, Eds. Springer Berlin Heidelberg, 2009, pp. 82–85.
- [55] C. Manifavas, K. Fysarakis, K. Rantos, K. Kagiambakis, and I. Papaefstathiou, "Policy-Based Access Control for Body Sensor Networks," in *Information Security Theory and Practice. Securing the Internet of Things SE - 11*, vol. 8501, D. Naccache and D. Sauveron, Eds. Springer Berlin Heidelberg, 2014, pp. 150–159.
- [56] K. Fysarakis, I. Papaefstathiou, C. Manifavas, K. Rantos, and O. Sultatos, "Policy-based access control for DPWS-enabled ubiquitous devices," in *Proceedings of the 2014 IEEE Emerging Technology and Factory Automation (ETFA)*, 2014, pp. 1–8.
- [57] A. Müller, H. Kinkelin, S. K. Ghai, and G. Carle, "A Secure Service Infrastructure for Interconnecting Future Home Networks based on DPWS and XACML Categories and Subject Descriptors The Devices Profile for Web Services," pp. 31–36.
- [58] O. Dohndorf, J. Kruger, H. Krumm, C. Fiehe, A. Litvina, I. Luck, and F.-J. Stewing, "Towards the Web of Things: Using DPWS to bridge isolated OSGi platforms," in *2010*



- 8th IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops)*, 2010, pp. 720–725.
- [59] H. Wang, B. Sheng, and Q. Li, "Elliptic curve cryptography-based access control in sensor networks," *Int. J. Secur. Networks*, vol. 1, no. 3/4, p. 127, 2006.
  - [60] M. L. Das, "Two-factor user authentication in wireless sensor networks," *IEEE Trans. Wirel. Commun.*, vol. 8, no. 3, pp. 1086–1090, Mar. 2009.
  - [61] 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.*, vol. 10, no. 10, pp. 3472–3481, Oct. 2011.
  - [62] Y. Zhou, Y. Zhang, and Y. Fang, "Access control in wireless sensor networks," *Ad Hoc Networks*, vol. 5, no. 1, pp. 3–13, Jan. 2007.
  - [63] Y. Faye, I. Niang, and T. Noel, "A survey of access control schemes in wireless sensor networks," *Proc. World Acad. Sci. Eng. Tech*, no. Laboratory LID, pp. 814–823, 2011.
  - [64] S. Yu, K. Ren, and W. Lou, "FDAC: Toward Fine-Grained Distributed Data Access Control in Wireless Sensor Networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 22, no. 4, pp. 352–362, 2011.
  - [65] J. Maerien, S. Michiels, C. Huygens, D. Hughes, and W. Joosen, "Access Control in Multi-party Wireless Sensor Networks," in *Wireless Sensor Networks SE - 3*, vol. 7772, P. Demeester, I. Moerman, and A. Terzis, Eds. Springer Berlin Heidelberg, 2013, pp. 34–49.
  - [66] 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*, 2012. .
  - [67] S. L. Keoh, K. Twidle, N. Pryce, A. E. Schaeffer-Filho, E. Lupu, N. Dulay, M. Sloman, S. Heeps, S. Strowes, J. Sventek, and E. Katsiri, "Policy-based Management for Body-Sensor Networks," in *4th International Workshop on Wearable and Implantable Body Sensor Networks (BSN 2007)*, no. March, Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 92–98.
  - [68] J.-E. Lee, S.-H. Cha, J.-O. Lee, S.-J. Kang, and K.-H. Cho, "A Policy-Based Management Framework for Self-managed Wireless Sensor Networks," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2006, vol. 4238, pp. 43–52.
  - [69] N. Matthys and W. Joosen, "Towards policy-based management of sensor networks," in *Proceedings of the 3rd international workshop on Middleware for sensor networks - MidSens '08*, 2008, pp. 13–18.
  - [70] W. Zhang and H. Xu, "A Policy Based Wireless Sensor Network Management Architecture," in *2010 Third International Conference on Intelligent Networks and Intelligent Systems*, 2010, pp. 552–555.
  - [71] T. Bourdenas and M. Sloman, "Starfish," in *Proceedings of the 2010 ICSE Workshop on Software Engineering for Adaptive and Self-Managing Systems - SEAMS '10*, 2010, pp. 75–83.

- [72] A. A. A. El-Aziz and A. Kannan, "Access control for healthcare data using extended XACML-SRBAC model," in *2012 International Conference on Computer Communication and Informatics*, 2012, pp. 1–4.
- [73] Y. Zhu, S. Keoh, M. Sloman, and E. Lupu, "A lightweight policy system for body sensor networks," *IEEE Trans. Netw. Serv. Manag.*, vol. 6, no. 3, pp. 137–148, Sep. 2009.
- [74] C. Santos-Pereira, A. B. Augusto, R. Cruz-Correia, and M. E. Correia, "A secure RBAC mobile agent access control model for healthcare institutions," *Proc. 26th IEEE Int. Symp. Comput. Med. Syst.*, pp. 349–354, Jun. 2013.
- [75] A. Faravelon, S. Chollet, C. Verdier, and A. Front, "Enforcing privacy as access control in a pervasive context," in *2012 IEEE Consumer Communications and Networking Conference (CCNC)*, 2012, pp. 380–384.
- [76] M. Jung, G. Kienesberger, W. Granzer, M. Unger, and W. Kastner, "Privacy enabled web service access control using SAML and XACML for home automation gateways," *2011 Int. Conf. Internet Technol. Secur. Trans.*, no. December, pp. 584–591, 2011.
- [77] J. E. Kim, G. Boulos, J. Yackovich, T. Barth, C. Beckel, and D. Mosse, "Seamless Integration of Heterogeneous Devices and Access Control in Smart Homes," in *2012 Eighth International Conference on Intelligent Environments*, 2012, pp. 206–213.
- [78] "Open Services Gateway Initiative (OSGi)." [Online]. Available: <http://www.osgi.org/>.
- [79] E.-A. Cho, C.-J. Moon, and D.-K. Baik, "Home gateway operating model using reference monitor for enhanced user comfort and privacy," *IEEE Trans. Consum. Electron.*, vol. 54, no. 2, pp. 494–500, May 2008.
- [80] P. Busnel, P. El-Khoury, S. Giroux, and K. Li, "An xacml-based security pattern to achieve socio-technical confidentiality in smart homes," *Int. J. Smart Home*, vol. 3, no. 1, pp. 17–26, 2009.
- [81] L. Seitz, G. Selander, and C. Gehrmann, "Authorization framework for the Internet-of-Things," in *2013 IEEE 14th International Symposium on a World of Wireless, Mobile and Multimedia Networks, WoWMoM 2013*, 2013.
- [82] P. Fremantle, B. Aziz, J. Kopecky, and P. Scott, "Federated Identity and Access Management for the Internet of Things," in *2014 International Workshop on Secure Internet of Things*, 2014, pp. 10–17.
- [83] T. Perumal, a. Ramli, and C. Leong, "Design and implementation of SOAP-based residential management for smart home systems," *IEEE Trans. Consum. Electron.*, vol. 54, no. 2, pp. 453–459, May 2008.
- [84] A. Westerinen, J. Schnizlein, J. Strassner, M. Scherling, B. Quinn, S. Herzog, A. Huynh, M. Carlson, J. Perry, and S. Waldbusser, "Terminology for Policy-Based Management," *RFC 3198*, 2001. [Online]. Available: <http://www.ietf.org/rfc/rfc3198.txt>.
- [85] "Sun Microsystems Laboratories, XACML." [Online]. Available: <http://sunxacml.sourceforge.net>.
- [86] "PicketBox XACML." [Online]. Available: <https://community.jboss.org/wiki/PicketBoxXACMLJBossXACML>.

- [87] "Holistic Enterprise-Ready Application Security Architecture Framework (Heras AF ) XACML." [Online]. Available: <http://www.herasaf.org/heras-af-xacml.html>.
- [88] "Enterprise Java XACML." [Online]. Available: <http://code.google.com/p/enterprise-java-xacml/>.
- [89] D. Durham, "RFC 2748 (The COPS (Common Open Policy Service) Protocol)," *Req. comments*, pp. 1–38, 2000.
- [90] R. Presuhn, "Version 2 of the Protocol Operations for the Simple Network Management Protocol (SNMP)," *RFC*, 2002.
- [91] S. Murer and C. Hagen, "15 Years of Service Oriented Architecture at Credit Suisse," pp. 1–30, 2013.
- [92] N. Serrano, J. Hernantes, and G. Gallardo, "Service-Oriented Architecture and Legacy Systems," *IEEE Softw.*, vol. 31, no. 5, pp. 15–19, 2014.
- [93] T. Nixon, A. Regnier, and R. Jeyaraman, "SOAP-over-UDP Version 1.1," *OASIS*, 2009. [Online]. Available: <http://docs.oasis-open.org/ws-dd/soapoverudp/1.1/os/wsdd-soapoverudp-1.1-spec-os.pdf>.
- [94] 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>.
- [95] 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.
- [96] D. Booth, H. Haas, F. McCabe, E. Newcomer, M. Champion, C. Ferris, and D. Orchard, "Web Services Architecture," *Group*, vol. 22, pp. 19–26, 2004.
- [97] M. D. E. Guttman, C. Perkins, J. Veizades, "RFC 2608: Service Location Protocol, Version 2," pp. 1–54, 1999.
- [98] A. Donoho, J. Costa-requena, T. Mcgee, A. Messer, A. Fiddian-green, and J. Fuller, "UPnP™ Device Architecture 1.1," *Architecture*, pp. 1–136, 2008.
- [99] E. Christensen, F. Curbera, G. Meredith, and S. Weerawarana, "Web Services Description Language (WSDL) 1.1," *International Business*, no. March, pp. 1–40, 2001.
- [100] T. Nixon, "UPnP Forum and DPWS Standardization Status," 2008. [Online]. Available: [http://download.microsoft.com/download/f/0/5/f05a42ce-575b-4c60-82d6-208d3754b2d6/UPnP\\_DPWS\\_RS08.pptx](http://download.microsoft.com/download/f/0/5/f05a42ce-575b-4c60-82d6-208d3754b2d6/UPnP_DPWS_RS08.pptx).
- [101] S. Tilkov and S. Vinoski, "Node.js: Using JavaScript to build high-performance network programs," *IEEE Internet Comput.*, vol. 14, pp. 80–83, 2010.
- [102] I. K. Chaniotis, K.-I. D. Kyriakou, and N. D. Tselikas, "Is Node.js a viable option for building modern web applications? A performance evaluation study," *Computing*, Mar. 2014.
- [103] A. Ojamaa and D. Karl, "Security Assessment of Node . js Platform," in *Information Systems Security*, 2012, pp. 35–43.

- [104] "Projects, Applications, and Companies Using Node." [Online]. Available: <https://github.com/joyent/node/wiki/Projects,-Applications,-and-Companies-Using-Node>.
- [105] R. McGraw, "A Survey of Access Control Models," *U.S. National Institute of Standards and Technology (NIST) Privilege (Access) Management Workshop*, 2009. .
- [106] "BeagleBoard-xM System Reference Manual, Rev. C." [Online]. Available: [http://beagleboard.org/static/BBxMSRM\\_latest.pdf](http://beagleboard.org/static/BBxMSRM_latest.pdf).
- [107] "BeagleBone System Reference Manual, RevA3\_1.0." [Online]. Available: [http://beagleboard.org/static/beaglebone/a3/Docs/Hardware/BONE\\_SRM.pdf](http://beagleboard.org/static/beaglebone/a3/Docs/Hardware/BONE_SRM.pdf).
- [108] R. B. Smith, "SPOTWorld and the Sun SPOT," *2007 6th Int. Symp. Inf. Process. Sens. Networks*, 2007.
- [109] 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>.
- [110] K. Rantos, A. Papanikolaou, and C. Manifavas, "IPsec over IEEE 802.15.4 for low power and lossy networks," in *Proceedings of the 11th ACM international symposium on Mobility management and wireless access - MobiWac '13*, 2013, pp. 59–64.
- [111] K. Rantos, A. Papanikolaou, C. Manifavas, and I. Papaefstathiou, "IPv6 security for low power and lossy networks," in *2013 IFIP Wireless Days (WD)*, 2013, pp. 1–8.
- [112] S. Raza, S. Duquennoy, T. Chung, D. Yazar, T. Voigt, U. Roedig, and A. H. M. Chung, "Securing communication in 6LoWPAN with compressed IPsec," in *2011 International Conference on Distributed Computing in Sensor Systems and Workshops DCOSS*, 2011, pp. 1–8.
- [113] N. Sastry and D. Wagner, "Security considerations for IEEE 802.15.4 networks," in *Proceedings of the 2004 ACM workshop on Wireless security - WiSe '04*, 2004, p. 32.
- [114] S. Raza, S. Duquennoy, J. Höglund, U. Roedig, and T. Voigt, "Secure communication for the Internet of Things-a comparison of link-layer security and IPsec for 6LoWPAN," *Secur. Commun. Networks*, vol. 7, no. 12, pp. 2654–2668, Dec. 2014.
- [115] T. Dierks and E. Rescorla, "RFC 5246 - The Transport Layer Security (TLS) Protocol Version 1.2," *IETF*, 2008. [Online]. Available: <http://tools.ietf.org/rfc/rfc5246.txt>.
- [116] E. Rescorla and N. Modadugu, "Datagram Transport Layer Security," *RFC 4347*, 2012. [Online]. Available: <http://tools.ietf.org/rfc/rfc6347.txt>.
- [117] D. E. Boyle and T. Newe, "On the implementation and evaluation of an elliptic curve based cryptosystem for Java enabled Wireless Sensor Networks," *Sensors Actuators, A Phys.*, vol. 156, no. 2, pp. 394–405, 2009.
- [118] W. Jung, S. Hong, M. Ha, Y.-J. Kim, and D. Kim, "SSL-Based Lightweight Security of IP-Based Wireless Sensor Networks," in *2009 International Conference on Advanced Information Networking and Applications Workshops*, 2009, pp. 1112–1117.
- [119] S. Fouladgar, B. Mainaud, K. Masmoudi, and H. Afifi, "Tiny 3-TLS: A trust delegation protocol for wireless sensor networks," in *Lecture Notes in Computer Science (including*

- subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics*), 2006, vol. 4357 LNCS, pp. 32–42.
- [120] V. Gupta and M. Wurm, “The Energy Cost of SSL in Deeply Embedded Systems,” Sun Microsystems, Inc., Mountain View, CA, USA, 2008.
  - [121] V. Gupta, M. Wurm, Y. Zhu, M. Millard, S. Fung, N. Gura, H. Eberle, and S. Chang Shantz, “Sizzle: A standards-based end-to-end security architecture for the embedded Internet,” in *Pervasive and Mobile Computing*, 2005, vol. 1, no. 4, pp. 425–445.
  - [122] F. Li, H. Zhang, and T. Takagi, “Efficient Signcryption for Heterogeneous Systems,” *IEEE Syst. J.*, vol. 7, no. 3, pp. 420–429, Sep. 2013.
  - [123] S. Nair, O. Al Ibrahim, and S. Abraham, “State Machine-Based Security Fusion for Resource-Constrained Environments,” *IEEE Syst. J.*, vol. 7, no. 3, pp. 430–441, Sep. 2013.
  - [124] 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>.
  - [125] R. Monzilo, C. Kaler, A. Nadalin, P. Hallam-Baker, and C. Milono, “Web Services Security SAML Token Profile Version 1.1.1,” *OASIS Standard*, 2012. [Online]. Available: <http://docs.oasis-open.org/wss-m/wss/v1.1.1/wss-SAMLTOKENProfile-v1.1.1.pdf>.
  - [126] R. Monzilo, C. Kaler, A. Nadalin, P. Hallam-Baker, and C. Milono, “Web Services Security Kerberos Token Profile Version 1.1.1,” *OASIS Standard*, 2012. [Online]. Available: <http://docs.oasis-open.org/wss-m/wss/v1.1.1/os/wss-KerberosTokenProfile-v1.1.1-os.pdf>.
  - [127] A. Nadalin, C. Kaler, R. Monzillo, and P. Hallam-Baker, “Web Service Security X.509 Certificate Token Profile 1.1,” *Oasis Stand. Specif.*, pp. 1–22, 2006.
  - [128] K. Lawrence, C. Kaler, T. Demartini, A. Nadalin, R. Monzillo, and P. Hallam-baker, “Web Services Security Rights Expression Language (REL) Token Profile 1.1,” *Language (Baltim)*, pp. 1–27, 2006.
  - [129] S. Cantor, J. Kemp, R. Philpott, and E. Maler, “Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0,” *OASIS*, 2005. [Online]. Available: <http://docs.oasis-open.org/security/saml/v2.0/saml-core-2.0-os.pdf>.
  - [130] J. Zhang and V. Varadharajan, “Wireless sensor network key management survey and taxonomy,” *Journal of Network and Computer Applications*, vol. 33, no. 2. pp. 63–75, 2010.
  - [131] K. Lawrence, C. Kaler, A. Nadalin, M. Goodner, M. Gudgin, A. Barbir, and H. Granqvist, “WS-SecureConversation 1.4,” *OASIS*, 2009. [Online]. Available: <http://docs.oasis-open.org/ws-sx/ws-secureconversation/v1.4/ws-secureconversation.pdf>.
  - [132] H. Liu, S. Pallickara, and G. Fox, “Performance of web services security,” in *13th Annual Mardi Gras Conference*, 2005, pp. 72–78.
  - [133] F. Lascelles and A. Flint, “WS-Security Performance,” *Websphere Journal*, 2006. [Online]. Available: <http://websphere.sys-con.com/node/204424>.

- [134] D. Whiting, R. Housley, and N. Ferguson, "Counter with CBC-MAC (CCM)," 2003. [Online]. Available: <http://tools.ietf.org/rfc/rfc3610.txt>.
- [135] "Knopflerfish - Open Source OSGi service platform." [Online]. Available: <http://www.knopflerfish.org/>.
- [136] K. Fysarakis, G. Hatzivasilis, I. Askoxylakis, and C. Manifavas, "RT-SPDM: Real-Time Security, Privacy and Dependability Management of Heterogeneous Systems," 2015, pp. 619–630.
- [137] G. Yang and M. Yacoub, *Body sensor networks*, vol. 6, no. 3. Springer London, 2006.
- [138] B. Alhaqbani and C. Fidge, "Access control requirements for processing electronic health records," in *Proceedings of the 2007 international conference on Business process management*, 2007, pp. 371–382.
- [139] P. Ray and J. Wimalasiri, "The need for technical solutions for maintaining the privacy of EHR.," in *Proceedings of the International Conference of IEEE Engineering in Medicine and Biology Society*, 2006, vol. 1, no. 1557–170X (Print) LA - eng PT - Journal Article SB - IM, pp. 4686–4689.
- [140] W. F. Ehrsam, C. H. W. Meyer, J. L. Smith, and W. L. Tuchman, "Message verification and transmission error detection by block chaining," U.S. Patent 4,074,066, 1978.
- [141] D. DeCouteau, M. Davis, and D. Staggs, "OASIS Cross-Enterprise Security and Privacy Authorization (XSPA) Profile of XACML v2.0 for Healthcare Version 1.0," *OASIS Stand. Specif.*, pp. 1–21, 2009.
- [142] B. Lo and G. Yang, "Body Sensor Networks: Infrastructure for Life Science Sensing Research," in *2006 IEEE/NLM Life Science Systems and Applications Workshop*, 2006, pp. 1–2.
- [143] W. Colitti, K. Steenhaut, and N. De Caro, "Integrating wireless sensor networks with the web," *Conf. Inf. Process. Sens. Networks*, pp. 2–6, 2011.
- [144] S. Cantor, J. Hodges, F. Hirsch, R. Philpott, R. S. A. Security, J. Hughes, A. Origin, H. Lockhart, B. E. A. Systems, M. Beach, R. Metz, B. A. Hamilton, R. Randall, T. Wisniewski, I. Reid, P. Austel, R. L. B. Morgan, P. C. Davis, J. Kemp, P. Madsen, A. Anderson, and S. Microsystems, "Profiles for the OASIS Security Assertion Markup Language ( SAML )," *Language (Baltim).*, vol. 16, no. 3, p. 66, 2005.
- [145] "WS4D-JMEDS DPWS Stack." [Online]. Available: <http://sourceforge.net/projects/ws4d-javame/>.
- [146] 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.
- [147] "STORK 2.0, Secure Identity Across Borders Linked 2.0," *D4.1 First version of process flows*, 2012. [Online]. Available: <https://www.eid-stork2.eu/>.
- [148] K. Fysarakis, D. Mylonakis, C. Manifavas, and I. Papaefstathiou, "Node.DPWS: Efficient Web Services for the IoT," *IEEE Softw.*, pp. 1–1, 2015.
- [149] "Mosquitto Open Source MQTT v3.1/v3.1.1 Broker." [Online]. Available:

<http://mosquitto.org>.

- [150] S. Hodges, N. Villar, J. Scott, and A. Schmidt, "A new era for ubicomp development," *IEEE Pervasive Comput.*, vol. 11, no. 1, pp. 5–9, 2012.
- [151] "Road safety study for the interim evaluation of Policy Orientations on Road Safety 2011-2020," *European Commission, Jeanne Breen Consulting*, 2015. [Online]. Available: [http://ec.europa.eu/transport/road\\_safety/pdf/study\\_final\\_report\\_february\\_2015\\_final.pdf](http://ec.europa.eu/transport/road_safety/pdf/study_final_report_february_2015_final.pdf).
- [152] J. A. Cook, I. V. Kolmanovsky, D. McNamara, E. C. Nelson, and K. V. Prasad, "Control, computing and communications: Technologies for the twenty-first century model T," *Proc. IEEE*, vol. 95, no. 2, pp. 334–355, 2007.
- [153] A. Doshi, B. T. Morris, and M. M. Trivedi, "On-road prediction of driver's intent with multimodal sensory cues," *IEEE Pervasive Comput.*, vol. 10, no. 3, pp. 22–34, 2011.
- [154] K. Lee, J. Flinn, T. J. Giuli, B. Noble, and C. Peplin, "AMC," in *Proceeding of the 11th annual international conference on Mobile systems, applications, and services - MobiSys '13*, 2013, p. 1.
- [155] "Connected Vehicle Infrastructure Deployment Considerations: Lessons Learned from Safety Pilot and Other Connected Vehicle Test Programs," *U.S. Department of Transportation*, 2014. [Online]. Available: [http://www.roadsbridges.com/sites/default/files/Deployment\\_Considerations\\_report\\_06\\_02\\_2014\\_v1.pdf](http://www.roadsbridges.com/sites/default/files/Deployment_Considerations_report_06_02_2014_v1.pdf).
- [156] "Telematic applications: eCall HGV/GV, additional data concept specification," *Economic Commission for Europe*, 2011. [Online]. Available: <http://www.unece.org/fileadmin/DAM/trans/doc/2011/dgwp15ac1/INF.30e.pdf>.
- [157] T. J. Giuli, D. Watson, and K. V. Prasad, "The last inch at 70 miles per hour," *IEEE Pervasive Comput.*, vol. 5, no. 4, pp. 20–27, 2006.
- [158] "Build secure embedded systems with nSHIELD," 2014.
- [159] U. D. of Transport, "National Transportation Statistics." [Online]. Available: [http://www.rita.dot.gov/bts/sites/rita.dot.gov.bts/files/publications/national\\_transportation\\_statistics/html/table\\_01\\_11.html](http://www.rita.dot.gov/bts/sites/rita.dot.gov.bts/files/publications/national_transportation_statistics/html/table_01_11.html).
- [160] "2015-2016 Automobile Industry Pocket Guide," *European Automobile Manufacturers' Association*. .
- [161] I. Garitano, S. Fayyad, and J. Noll, "Multi-Metrics Approach for Security, Privacy and Dependability in Embedded Systems," *Wirel. Pers. Commun.*, vol. 81, no. 4, pp. 1359–1376, Apr. 2015.
- [162] G. Hatzivasilis, I. Papaefstathiou, and C. Manifavas, "ModConTR: A modular and configurable trust and reputation-based system for secure routing in ad-hoc networks," in *2014 IEEE/ACS 11th International Conference on Computer Systems and Applications (AICCSA)*, 2014, pp. 56–63.
- [163] E. Rissanen, "XACML v3.0 Privacy Policy Profile Version 1.0," *OASIS*. [Online]. Available: <http://docs.oasis-open.org/xacml/3.0/privacy/v1.0/xacml-3.0-privacy-v1.0.html>.

- [164] A. Huertas Celdran, F. J. Garcia Clemente, M. Gil Perez, and G. Martinez Perez, "SeCoMan: A Semantic-Aware Policy Framework for Developing Privacy-Preserving and Context-Aware Smart Applications," *IEEE Syst. J.*, pp. 1–14, 2014.
- [165] R. Abdunabi, M. Al-Lail, I. Ray, and R. B. France, "Specification, Validation, and Enforcement of a Generalized Spatio-Temporal Role-Based Access Control Model," *IEEE Syst. J.*, vol. 7, no. 3, pp. 501–515, Sep. 2013.
- [166] E. Georgakakis, S. A. Nikolidakis, D. D. Vergados, and C. Douligeris, "Spatio temporal emergency role based access control (STEM-RBAC): A time and location aware role based access control model with a break the glass mechanism," in *2011 IEEE Symposium on Computers and Communications (ISCC)*, 2011, pp. 764–770.
- [167] 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.
- [168] Z. Shelby, K. Hartke, and C. Bormann, "The Constrained Application Protocol (CoAP)," *IETF RFC 7252*, 2014. [Online]. Available: <https://tools.ietf.org/html/rfc7252>.
- [169] P. Saint-Andre, "Extensible Messaging and Presence Protocol (XMPP) : Core," *Internet Engineering Task Force*. pp. 1–212, 2011.
- [170] K. Markantonakis, R. N. Akram, and M. G. Msgna, "Secure and Trusted Application Execution on Embedded Devices," 2015, pp. 3–24.
- [171] S. Alam, M. M. R. Chowdhury, and J. Noll, "Interoperability of Security-Enabled {Internet of Things}," *Wirel. Pers. Commun.*, vol. 61, no. 3, pp. 567–586, 2011.
- [172] M. Daněš, J. Kadlec, R. Bartosinski, and L. Kohout, "Increasing the level of abstraction in {FPGA}-based designs," in *International Conference on Field Programmable Logic and Applications (FPL 2008)*, 2008, pp. 5–10.
- [173] M. Msgna, K. Markantonakis, and K. Mayes, "The B-Side of Side Channel Leakage: Control Flow Security in Embedded Systems," 2013, pp. 288–304.
- [174] M. Msgna, K. Markantonakis, D. Naccache, and K. Mayes, "Verifying Software Integrity in Embedded Systems: A Side Channel Approach," pp. 261–280, 2014.
- [175] "Trusted Platform Module. ISO/IEC 11889-1:2009." [Online]. Available: [http://www.trustedcomputinggroup.org/resources/tpm\\_main\\_specification](http://www.trustedcomputinggroup.org/resources/tpm_main_specification).
- [176] A. Kurs, A. Karalis, R. Moffatt, J. D. Joannopoulos, P. Fisher, and M. Soljačić, "Wireless power transfer via strongly coupled magnetic resonances," *Science (80-. )*, vol. 317, no. 5834, pp. 83–86, 2007.
- [177] A. Karalis, J. D. Joannopoulos, and M. Soljačić, "Efficient wireless non-radiative mid-range energy transfer," *Ann. Phys. (N. Y.)*, vol. 323, no. 1, pp. 34–48, 2008.
- [178] M. Naedele, "An access control protocol for embedded devices," in *4th International IEEE Conference on Industrial Informatics (INDIN '06)*, 2006, pp. 565–569.
- [179] C. Szilagyi and P. Koopman, "Low cost multicast authentication via validity voting in time-triggered embedded control networks," in *5th Workshop on Embedded Systems Security (WESS '10)*, 2010, pp. 10:1–10:10.



- [180] C. Szilagyi and P. Koopman, "Flexible multicast authentication for time-triggered embedded control network applications," in *IEEE/IFIP International Conference on Dependable Systems & Networks (DSN '09)*, 2009, pp. 165–174.
- [181] K. Rantos, A. Papanikolaou, K. Fysarakis, and C. Manifavas, "Secure policy-based management solutions in heterogeneous embedded systems networks," in *2012 International Conference on Telecommunications and Multimedia (TEMU)*, 2012, pp. 227–232.
- [182] G. Carl, G. Kesidis, R. R. Brooks, and S. Rai, "Denial-of-service attack detection techniques," *IEEE Internet Comput.*, vol. 10, no. 1, 2006.
- [183] D. R. Raymond and S. F. Midkiff, "Denial-of-Service in Wireless Sensor Networks: Attacks and Defenses," *IEEE Pervasive Comput.*, vol. 7, no. 1, pp. 74–81, Jan. 2008.
- [184] R. Smith, "PhlashDance, Discovering permanent denial of service attacks against embedded systems." 2008.
- [185] K. Stefanidis and D. N. Serpanos, "Implementing filtering and traceback mechanism for packet-marking IP-traceback schemes against DDoS attacks," in *2008 4th International IEEE Conference Intelligent Systems*, 2008, vol. 2, pp. 14–28–14–33.
- [186] I. Aad, J.-P. Hubaux, and E. W. W. Knightly, "Impact of Denial of Service Attacks on Ad Hoc Networks," *IEEE/ACM Trans. Netw.*, vol. 16, no. 4, pp. 791–802, Aug. 2008.
- [187] R. N. Akram, K. Markantonakis, R. Holloway, S. Kariyawasam, S. Ayub, A. Seeam, and R. Atkinson, "Challenges of security and trust in Avionics Wireless Networks," in *2015 IEEE/AIAA 34th Digital Avionics Systems Conference (DASC)*, 2015, pp. 4B1–1–4B1–12.
- [188] M. R. Doomun and K. M. S. Soyjaudah, "Analytical comparison of cryptographic techniques for resource-constrained wireless security," *Int. J. Netw. Secur.*, vol. 9, no. 1, pp. 82–94, 2009.
- [189] B. Preneel, "Research challenges in lightweight cryptography." 2009.
- [190] T. Eisenbarth, S. Kumar, C. Paar, A. Poschmann, and L. Uhsadel, "A Survey of Lightweight-Cryptography Implementations," *IEEE Des. Test Comput.*, vol. 24, no. 6, pp. 522–533, Nov. 2007.
- [191] R. Roman, C. Alcaraz, and J. Lopez, "A Survey of Cryptographic Primitives and Implementations for Hardware-Constrained Sensor Network Nodes," *Mob. Networks Appl.*, vol. 12, no. 4, pp. 231–244, Oct. 2007.
- [192] C. Paar, A. Poschmann, and M. Robshaw, "New designs in lightweight symmetric encryption," *RFID Secur.*, vol. III, pp. 349–371, 2009.
- [193] P. Kitsos, N. Sklavos, M. Parousi, and A. N. Skodras, "A comparative study of hardware architectures for lightweight block ciphers," *Comput. Electr. Eng.*, vol. 38, no. 1, pp. 148–160, Jan. 2012.
- [194] A. Bogdanov, L. R. Knudsen, G. Leander, C. Paar, A. Poschmann, M. J. B. Robshaw, Y. Seurin, and C. Vikkelsøe, "PRESENT: An Ultra-Lightweight Block Cipher," in *Cryptographic Hardware and Embedded Systems - CHES 2007*, vol. 4727, P. Paillier and I. Verbauwhede, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 450–466.

- [195] K. Shibutani, T. Isobe, H. Hiwatari, A. Mitsuda, T. Akishita, and T. Shirai, "Piccolo: An Ultra-Lightweight Blockcipher," *Cryptogr. Hardw. Embed. Syst. CHES 2011, Springer, LNCS*, vol. 6917, pp. 342–357, 2011.
- [196] C. De Cannière, "Trivium: A Stream Cipher Construction Inspired by Block Cipher Design Principles," in *Information Security*, vol. 507932, 2006, pp. 171–186.
- [197] D. Watanabe, K. Ideguchi, J. Kitahara, K. Muto, and H. Furuichi, "Enocoro-80: A Hardware Oriented Stream Cipher," in *3rd International Conference on Availability, Reliability and Security (ARES 08)*, 2008, pp. 1294–1300.
- [198] M. Feldhofer, J. Wolkerstorfer, and V. Rijmen, "AES implementation on a grain of sand," *IEE Proc. Inf. Secur.*, vol. 152, no. 1, pp. 13–20, 2005.
- [199] D. Engels, M.-J. O. Saarinen, P. Schweitzer, and E. M. Smith, "The Hummingbird-2 Lightweight Authenticated Encryption Algorithm," in *7th Workshop of {RFID} Security and Privacy {{RFIDSec '11}}*, vol. 7055, A. Juels and C. Paar, Eds. Amherst, Massachusetts, USA, 2012, pp. 19–31.
- [200] R. Beaulieu, D. Shors, J. Smith, S. Treatman-Clark, B. Weeks, and L. Wingers, "The SIMON and SPECK lightweight block ciphers," in *Proceedings of the 52nd Annual Design Automation Conference on - DAC '15*, 2015, pp. 1–6.
- [201] F. Karakoç, H. H. Demirci, A. E. Harmancı, F. Karakoc, H. H. Demirci, and A. E. Harmancı, "ITUbee: A Software Oriented Lightweight Block Cipher," in *2nd International workshop on lightweight cryptography for security & privacy (LightSEC 2013)*, vol. 8162, 2013, pp. 16–27.
- [202] ECRYPT, "eSTREAM project," 2008. [Online]. Available: <http://www.ecrypt.eu.org/stream/>.
- [203] A. Bogdanov, G. Leander, C. Paar, A. Poschmann, M. J. B. Robshaw, and Y. Seurin, "Hash Functions and RFID Tags: Mind the Gap," in *Cryptographic Hardware and Embedded Systems – CHES 2008*, vol. 5154, E. Oswald and P. Rohatgi, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 283–299.
- [204] K. Gaj, E. Homsirikamol, M. Rogawski, R. Shahid, and M. U. Sharif, "Comprehensive Evaluation of High-Speed and Medium-Speed Implementations of Five {SHA-3} Finalists Using {Xilinx} and {Altera} {FPGAs}." 2012.
- [205] J. Guo, T. Peyrin, and A. Poschmann, "The PHOTON Family of Lightweight Hash Functions," in *Advances in Cryptology – CRYPTO 2011, Springer, LNCS*, vol. 6841, P. Rogaway, Ed. Santa Barbara, CA, USA, 2011, pp. 222–239.
- [206] A. Bogdanov, M. Knežević, G. Leander, D. Toz, K. Varıcı, and I. Verbauwhede, "sponge: A Lightweight Hash Function," in *13th International Workshop on Cryptographic Hardware and Embedded Systems {{CHES '11}}*, vol. 6917, B. Preneel and T. Takagi, Eds. Nara, Japan, 2011, pp. 312–325.
- [207] A. A. Kamal and A. M. Youssef, "An FPGA implementation of the NTRUEncrypt cryptosystem," in *2009 International Conference on Microelectronics - ICM*, 2009, pp. 209–212.
- [208] C. Manifavas, G. Hatzivasilis, K. Fysarakis, and K. Rantos, "Lightweight Cryptography for Embedded Systems – A Comparative Analysis," in *6th Internations Workshop on*

- Autonomous and Spontaneous Security (SETOP 2013)*, vol. 8247, RHUL, Egham, U.K.: Springer-Verlag Berlin Heidelberg, 2014, pp. 333–349.
- [209] G. Hatzivasilis, A. Theodoridis, H. Gasparis, C. Manifavas, and I. Papaefstathiou, "ULCL - An Ultra-lightweight Cryptographic Library for Embedded Systems," in *Proceedings of the 4th International Conference on Pervasive and Embedded Computing and Communication Systems*, 2014, pp. 247–254.
  - [210] C. Kuo, M. Luk, R. Negi, and A. Perrig, "Message-in-a-bottle," in *Proceedings of the 5th international conference on Embedded networked sensor systems - SenSys '07*, 2007, p. 233.
  - [211] B. Doyle, S. Bell, A. F. Smeaton, K. McCusker, and N. O'Connor, "Security considerations and key negotiation techniques for power constrained sensor networks," *Comput. J.*, vol. 49, no. 4, pp. 443–453, May 2006.
  - [212] Jyh-How Huang, J. Buckingham, and R. Han, "A Level Key Infrastructure for Secure and Efficient Group Communication in Wireless Sensor Network," in *First International Conference on Security and Privacy for Emerging Areas in Communications Networks (SECURECOMM'05)*, 2005, pp. 249–260.
  - [213] K. M. Martin and M. B. Paterson, "An Application-Oriented Framework for Wireless Sensor Network Key Establishment," *Electron. Notes Theor. Comput. Sci.*, vol. 192, no. 2, pp. 31–41, May 2008.
  - [214] D. Boneh and M. Franklin, "Identity-Based Encryption from the Weil Pairing," in *SIAM Journal on Computing*, vol. 32, no. 3, 2001, pp. 213–229.
  - [215] L. B. Oliveira, M. Scott, J. Lopez, and R. Dahab, "TinyPBC: Pairings for authenticated identity-based non-interactive key distribution in sensor networks," in *2008 5th International Conference on Networked Sensing Systems*, 2008, vol. 34, no. 3, pp. 173–180.
  - [216] C.-Y. Chen and H.-C. Chao, "A survey of key distribution in wireless sensor networks," *Secur. Commun. Networks*, vol. 7, no. 12, pp. 2495–2508, Dec. 2014.
  - [217] M. A. Simplício, P. S. L. M. L. M. Barreto, C. B. Margi, T. C. M. B. M. B. Carvalho, M. A. Simplício Jr., P. S. L. M. L. M. Barreto, C. B. Margi, and T. C. M. B. M. B. Carvalho, "A survey on key management mechanisms for distributed Wireless Sensor Networks," *Comput. Networks*, vol. 54, no. 15, pp. 2591–2612, Oct. 2010.
  - [218] S. Yoshihama, T. Ebringer, M. Nakamura, S. Munetoh, T. Mishina, and H. Maruyama, "ws-Attestation: Enabling Trusted Computing on Web Services," in *Test and Analysis of Web Services*, L. Baresi and E. Di Nitto, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 441–469.
  - [219] P. Calhoun, J. Loughney, E. Guttman, G. Zorn, and J. Arkko, "Diameter base protocol," 2003.
  - [220] G. Hatzivasilis and C. Manifavas, "Building Trust in Ad Hoc Distributed Resource-Sharing Networks Using Reputation-Based Systems," in *2012 16th Panhellenic Conference on Informatics*, 2012, pp. 416–421.
  - [221] P. Resnick and R. Zeckhauser, "Trust among strangers in Internet transactions: Empirical analysis of eBay's reputation system," *Adv. Appl. ...*, no. 11, pp. 127–157,

- 2002.
- [222] S. Buchegger, C. Tissieres, and J.-Y. Le Boudec, "A Test-Bed for Misbehavior Detection in Mobile Ad-hoc Networks &#8212; How Much Can Watchdogs Really Do?," in *Sixth IEEE Workshop on Mobile Computing Systems and Applications*, 2004, pp. 102–111.
  - [223] S. Madhavi and T. H. Kim, "An intelligent distributed reputation based mobile intrusion detection system," *Int. J. Comput. Sci. Telecommun.*, vol. 2, no. 7, pp. 53–58, 2011.
  - [224] T. R. Abdalrazak and H. K. Sawant, "Collaborative trust-based secure routing based ad-hoc routing protocol," *Int. J. Mod. Eng. Res.*, vol. 2, no. 2, pp. 95–101, 2012.
  - [225] Y. Zhang, L. Xu, and X. Wang, "A Cooperative Secure Routing Protocol based on Reputation System for Ad Hoc Networks," *J. Commun.*, vol. 3, no. 6, pp. 43–50, Nov. 2008.
  - [226] K. Damandeep and S. Jyotsna, "Proposed P2P trust and reputation based model to secure grid," *IJCA Proc. Int. Conf. Recent Adv. Futur. Trends Inf. Technol. (iRAFIT 2012)*, vol. iRAFIT, no. 2, pp. 19–24, 2012.
  - [227] B. Gedik and L. Liu, "Protecting location privacy with personalized k-Anonymity: {Architecture} and algorithms," *IEEE Trans. Mob. Comput.*, vol. 7, no. 1, pp. 1–18, Jan. 2008.
  - [228] G. Zhong and U. Hengartner, "Toward a distributed k-anonymity protocol for location privacy," in *Proceedings of the 7th ACM workshop on Privacy in the electronic society - WPES '08*, 2008, p. 33.
  - [229] P. Golle and K. Partridge, "On the anonymity of home/work location pairs," in *7th International Conference on Pervasive Computing (Pervasive 2009)*, 2009, vol. 5538, pp. 390–397.
  - [230] L. Liu, "From data privacy to location privacy: models and algorithms," in *Proceedings of the 33rd international conference on Very large databases*, 2007, pp. 1429–1430.
  - [231] K. Fysarakis, C. Manifavas, I. Papaefstathiou, and A. Adamopoulos, "A lightweight anonymity & location privacy service," in *IEEE International Symposium on Signal Processing and Information Technology*, 2013, pp. 000124–000129.
  - [232] P. Kotzanikolaou, E. Magkos, N. Petrakos, C. Douligeris, and V. Chrissikopoulos, "Fair Anonymous Authentication for Location Based Services," 2013, pp. 1–14.
  - [233] L. Liu, "Privacy and location anonymization in location-based services," *SIGSPATIAL Spec.*, vol. 1, no. 2, pp. 15–22, Jul. 2009.
  - [234] A. Machanavajjhala, D. Kifer, J. Gehrke, and M. Venkitasubramaniam, "L -diversity," *ACM Trans. Knowl. Discov. Data*, vol. 1, no. 1, p. 3–es, Mar. 2007.
  - [235] K. Gupta, A. S. Yadav, and S. Yadav, "Location Privacy Using User Anonymity and Dummy Locations," *Int. J. Innov. Technol. Creat. Eng.*, vol. 1, no. 10, pp. 5–8, 2011.
  - [236] B. Lee, J. Oh, H. Yu, and J. Kim, "Protecting location privacy using location semantics," in *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '11*, 2011, p. 1289.

- [237] M. Gruteser and D. Grunwald, "Enhancing Location Privacy in Wireless LAN Through Disposable Interface Identifiers: A Quantitative Analysis," *Mob. Networks Appl.*, vol. 10, no. 3, pp. 315–325, Jun. 2005.
- [238] Leping Huang, K. Matsuura, H. Yamane, and K. Sezaki, "Enhancing wireless location privacy using silent period," in *IEEE Wireless Communications and Networking Conference, 2005*, 2005, vol. 2, pp. 1187–1192.
- [239] B. H. B. Hoh and M. Gruteser, "Protecting Location Privacy Through Path Confusion," in *1st International Conference on Security and Privacy for Emerging Areas in Communications Networks (SECURECOMM '05)*, 2005, pp. 194–205.
- [240] P. Hallam-baker and M. Hondo, "Web Services Policy Framework ( WS- Policy )," *Structure*, no. March. pp. 1–25, 2006.
- [241] K. Lawrence, C. Kaler, A. Nadalin, M. Goodner, and M. Gudgin, *WS-Trust 1.4*. 2009, pp. 1–85.
- [242] "Service-Oriented Architecture for Devices (SOA4D)." [Online]. Available: <http://cms.soa4d.org/>.
- [243] J. Hui and P. Thubert, "Compression Format for IPv6 Datagrams over IEEE 802.15.4-Based Networks."
- [244] Ieee, "IEEE Standard for Local and metropolitan area networks, Part 15.4: Low-Rate Wireless Personal Area Networks," *IEEE Std 802.15.4-2011*, no. June. 2011.
- [245] T. Kothmayr, C. Schmitt, W. Hu, M. Brunig, and G. Carle, "A DTLS based end-to-end security architecture for the Internet of Things with two-way authentication," in *37th Annual IEEE Conference on Local Computer Networks -- Workshops*, 2012, pp. 956–963.
- [246] "Gumstix Verdex Pro XL6P COM." [Online]. Available: <https://store.gumstix.com/verdex-pro-xl6p-com.html>.
- [247] Acme Systems, "FOX LX." [Online]. Available: <http://www.acmesystems.it/FOXLX>.
- [248] Freescale, "i.MX51 Processors." [Online]. Available: [http://www.freescale.com/webapp/sps/site/taxonomy.jsp?code=IMX51\\_FAMILY](http://www.freescale.com/webapp/sps/site/taxonomy.jsp?code=IMX51_FAMILY).
- [249] Xilinx, "Spartan-6 FPGA Family." [Online]. Available: <http://www.xilinx.com/products/silicon-devices/fpga/spartan-6/index.htm>.
- [250] K. Dietrich, J. Winter, G. Luzhnica, and S. Podesser, "Implementation Aspects of Anonymous Credential Systems for Mobile Trusted Platforms," in *12th IFIP TC 6/TC 11 International Conference on Communications and Multimedia Security (CMS '11)*, 2011, pp. 45–58.
- [251] K. Dietrich and J. Winter, "Towards customizable, application specific mobile trusted modules," in *Proceedings of the fifth ACM workshop on Scalable trusted computing - STC '10*, 2010, p. 31.
- [252] ARM Security Technology, "Building a Secure System using TrustZone Technology," 2008. [Online]. Available: [http://infocenter.arm.com/help/topic/com.arm.doc.prd29-genc-009492c/PRD29-GENC-009492C\\_trustzone\\_security\\_whitepaper.pdf](http://infocenter.arm.com/help/topic/com.arm.doc.prd29-genc-009492c/PRD29-GENC-009492C_trustzone_security_whitepaper.pdf).
- [253] A. Böttcher, B. Kauer, and H. Härtig, "Trusted Computing Serving an Anonymity

- Service,” in *1st International Conference on Trusted Computing and Trust in Information Technologies: Trusted Computing (Trust '08) -- Challenges and Applications*, 2008, pp. 143–154.
- [254] H. Hartig, M. Hohmuth, N. Feske, C. Helmuth, A. Lackorzynski, F. Mehnert, and M. Peter, “The Nizza Secure-System Architecture,” in *International Conference on Collaborative Computing: Networking, Applications and Worksharing*, 2005, p. 10.
  - [255] A. Muñoz, A. Maña, and P. Antón, “In the Track of the Agent Protection: A Solution Based on Cryptographic Hardware,” in *Computer Network Security*, vol. 6258, I. Kottenko and V. Skormin, Eds. Springer Berlin, Heidelberg, 2010, pp. 284–297.
  - [256] J. Winter and K. Dietrich, “A Hijacker’s Guide to the LPC Bus,” in *Public Key Infrastructures, Services and Applications*, vol. 7163, S. Petkova-Nikova, A. Pashalidis, and G. Pernul, Eds. Springer Berlin, Heidelberg, 2012, pp. 176–193.
  - [257] F. Vater, S. Peter, and P. Langendörfer, “Combinatorial Logic Circuitry as Means to Protect Low Cost Devices Against Side Channel Attacks,” in *Information Security Theory and Practices. Smart Cards, Mobile and Ubiquitous Computing Systems*, vol. 4462, D. Sauveron, K. Markantonakis, A. Bilas, and J.-J. Quisquater, Eds. Springer Berlin, Heidelberg, 2007, pp. 244–253.
  - [258] P. Christou, K. Kyriakoulakos, E. Sotiriadis, A. H. Authentication, K. Papadopoulos, G.-G. Mplemenos, and I. Papaefstathiou, “Low-Power Security Modules optimized for WSNs,” in *16th International Workshop on Systems, Signals and Image Processing (IWSSIP)*, 2009, pp. 1–4.
  - [259] G.-G. Mplemenos, P. Christou, and I. Papaefstathiou, “Using reconfigurable hardware devices in WSNs for accelerating and reducing the power consumption of header processing tasks,” in *2009 IEEE 3rd International Symposium on Advanced Networks and Telecommunication Systems (ANTS)*, 2009, pp. 1–3.
  - [260] A. Brokalakis, G.-G. Mplemenos, K. Papadopoulos, and I. Papaefstathiou, “RESENSE: An Innovative, Reconfigurable, Powerful and Energy Efficient WSN Node,” in *2011 IEEE International Conference on Communications (ICC)*, 2011, pp. 1–5.
  - [261] P. Simons, E. van der Sluis, and V. van der Leest, “Buskeeper PUFs, a promising alternative to D Flip-Flop PUFs,” in *IEEE International Symposium on Hardware-Oriented Security and Trust (HOST '12)*, 2012, pp. 7–12.
  - [262] S. Katzenbeisser, Ü. Koçabas, V. Leest, A.-R. Sadeghi, G.-J. Schrijen, H. Schröder, and C. Wachsmann, “Recyclable PUFs: Logically Reconfigurable PUFs,” in *Cryptographic Hardware and Embedded Systems (CHES '11)*, 2011, vol. 6917, pp. 374–389.
  - [263] J. Petit, C. Bosch, M. Feiri, and F. Kargl, “On the potential of PUF for pseudonym generation in vehicular networks,” in *IEEE Vehicular Networking Conference (VNC)*, 2012, pp. 94–100.
  - [264] M. Feiri, J. Petit, and F. Kargl, “Efficient and secure storage of private keys for pseudonymous vehicular communication,” in *First Workshop on Security, Privacy and Dependability for CyberVehicles (CyCar '13) at 20th ACM Conference on Computer and Communications Security (CCS '13)*, 2013, pp. 9–18.
  - [265] S. T. Ben Hamida, J.-B. Pierrot, and C. Castelluccia, “An Adaptive Quantization Algorithm for Secret Key Generation Using Radio Channel Measurements,” in *3rd*

- International Conference on New Technologies, Mobility and Security (NTMS '09)*, 2010, pp. 1–5.
- [266] J. Brakensiek, A. Dröge, M. Botteck, H. Härtig, and A. Lackorzynski, “Virtualization as an Enabler for Security in Mobile Devices,” in *1st Workshop on Isolation and Integration in Embedded Systems (IIES '08)*, 2008, no. April, pp. 17–22.
  - [267] M. Peter, H. Schild, A. Lackorzynski, and A. Warg, “Virtual Machines Jailed Virtualization in Systems with Small Trusted Computing Bases,” in *1st EuroSys Workshop on Virtualization Technology for Dependable Systems (VDTS '09)*, 2009, pp. 18–23.
  - [268] A. Lackorzynski, T. Universit, H. Schild, A. Lackorzynski, and A. Warg, “Faithful Virtualization on a Real-Time Operating System,” in *Eleventh Real-Time Linux Workshop*, 2009, pp. 237–243.
  - [269] U. Steinberg and B. Kauer, “NOVA : A Microhypervisor-Based Secure Virtualization Architecture,” pp. 209–222, 2010.
  - [270] S. Liebergeld, M. Peter, T. Universit, A. Lackorzynski, T. Universit, and A. Lackorzynski, “Towards Modular Security-conscious Virtual Machines,” in *Twelfth Real-Time Linux Workshop*, 2010.
  - [271] P. Trakadas, T. Zahariadis, H. C. Leligou, S. Voliotis, and K. Papadopoulos, “Analyzing energy and time overhead of security mechanisms in Wireless Sensor Networks,” in *2008 15th International Conference on Systems, Signals and Image Processing*, 2008, no. September, pp. 137–140.
  - [272] A. Kargl, S. Pyka, and H. Seuschek, “Fast Arithmetic on ATmega128 for Elliptic Curve Cryptography.” 2008.
  - [273] S. Li, T. Li, X. Wang, J. Zhou, and K. Chen, “Efficient Link Layer Security Scheme for Wireless Sensor Networks,” *J. Inf. Comput. Sci.*, vol. 4, no. 2, pp. 553–567, 2007.
  - [274] C. Karlof, N. Sastry, and D. Wagner, “TinySec: A Link Layer Security Architecture for Wireless Sensor Networks,” in *2nd International Conference on Embedded Networked Sensor Systems (SenSys '04)*, 2004, pp. 162–175.
  - [275] A. Poschmann, G. Leander, K. Schramm, and C. Paar, “New Lightweight Crypto Algorithms for RFID,” in *IEEE International Symposium on Circuits and Systems (ISCAS '07)*, 2007, pp. 1843–1846.
  - [276] L. Uhsadel, A. Poschmann, and C. Paar, “Enabling Full-Size Public-Key Algorithms on 8-Bit Sensor Nodes,” in *Security and Privacy in Ad-hoc and Sensor Networks*, vol. 4572, F. Stajano, C. Meadows, S. Capkun, and T. Moore, Eds. Springer Berlin, Heidelberg, 2007, pp. 73–86.
  - [277] F. Vater and P. Langendörfer, “An Area Efficient Realisation of AES for Wireless Devices,” *it -- Inf. Technol.*, vol. 49, no. 3, pp. 188–193, 2007.
  - [278] A. Poschmann, D. Westhoff, and A. Weimerskirch, “Dynamic Code-Update for the Efficient Usage of Security Components in {WSNs},” in *Communication in Distributed Systems (KiVS), ITG-GI Conference*, 2007, pp. 1–11.
  - [279] K. Potzmader, J. Winter, D. Hein, C. Hanser, P. Teu, and L. Chen, “Group Signatures on Mobile Devices: {Practical} Experiences,” in *Trust and Trustworthy Computing*, vol.

- 7904, M. Huth, N. Asokan, S. Čapkun, I. Flechais, and L. Coles-Kemp, Eds. Springer Berlin, Heidelberg, 2013, pp. 47–64.
- [280] G. Montenegro, N. Kushalnagar, J. Hui, and D. Culler, “Transmission of IPv6 Packets over IEEE 802.15.4 Networks,” *RFC*, vol. RFC4944, no. 4944. pp. 1–30, 2007.
  - [281] W.-B. Poettner and L. Wolf, “IEEE 802.15.4 Packet Analysis with Wireshark and Off-the-Shelf Hardware,” in *7th International Conference on Networked Sensing Systems (INSS ’10)*, 2010.
  - [282] J. Granjal, E. Monteiro, and J. Sá Silva, “Enabling Network-Layer security on IPv6 Wireless Sensor Networks,” in *IEEE Global Telecommunications Conference (GLOBECOM ’10)*, 2010, pp. 1–6.
  - [283] J. Granjal, E. Monteiro, and J. Sá Silva, “A Secure Interconnection Model for IPv6 Enabled Wireless Sensor Networks,” in *IFIP Wireless Days (WD ’10)*, 2010, pp. 1–6.
  - [284] P. Trakadas, T. Zahariadis, H. C. Leligou, S. Voliotis, and K. Papadopoulos, “AWISSENET: Setting Up a Secure Wireless Sensor Network,” in *50th International Symposium ELMAR 2008, focused on Mobile Multimedia*, 2008, pp. 519–523.
  - [285] K. Papadopoulos, T. Zahariadis, H. C. Leligou, and S. Voliotis, “Sensor Networks Security Issues in Augmented Home Environment,” in *12th IEEE International Symposium on Consumer Electronics (ISCE ’08)*, 2008, pp. 519–523.
  - [286] K. Dietrich, “Anonymous Client Authentication for Transport Layer Security,” in *Communications and Multimedia Security*, vol. 6109, Springer Berlin, Heidelberg, 2010, pp. 268–280.
  - [287] C. Wachsmann, L. Chen, K. Dietrich, H. Löhr, A.-R. Sadeghi, and J. Winterhor, “Lightweight Anonymous Authentication with TLS and DAA for Embedded Mobile Devices,” in *Information Security*, vol. 6531, M. Burmester, G. Tsudik, S. Magliveras, and I. Ilić, Eds. Springer Berlin, Heidelberg, 2010, pp. 84–98.
  - [288] K. Dietrich and J. Winter, “A Secure and Practical Approach for Providing Anonymity Protection for Trusted Platforms,” in *12th International Conference on Information and Communications Security (ICICS ’10)*, 2010, pp. 311–324.
  - [289] “Near Field Communication (NFC).” [Online]. Available: <http://www.nfc-forum.org/home>.
  - [290] K. Dietrich, “Anonymous RFID Authentication Using Trusted Computing Technologies,” in *Radio Frequency Identification: Security and Privacy Issues*, vol. 6370, S. B. Ors Yalcin, Ed. Springer Berlin, Heidelberg, 2010, pp. 91–102.
  - [291] K. Dietrich, “An Integrated Architecture for Trusted Computing for Java Enabled Embedded Devices,” in *ACM Workshop on Scalable Trusted Computing (STC ’07)*, 2007, pp. 2–6.
  - [292] F. Armknecht, L. Chen, A.-R. Sadeghi, and C. Wachsmann, “Anonymous Authentication for RFID Systems,” in *Radio Frequency Identification: {Security} and Privacy Issues*, vol. 6370, S. B. Ors Yalcin, Ed. Springer Berlin, Heidelberg, 2010, pp. 158–175.
  - [293] V. Manolopoulos, P. Papadimitratos, S. Tao, and A. Rusu, “Securing Smartphone based ITS,” in *11th IEEE International Conference on ITS Telecommunications (ITST)*,



- 2011, pp. 201–206.
- [294] N. Alexiou, M. Laganá, S. Gisdakis, M. Khodaei, and P. Papadimitratos, “VeSPA: Vehicular Security and Privacy-preserving Architecture,” in *2nd ACM Workshop on Hot Topics on Wireless Network Security and Privacy (HotWiSec '13)*, 2013, pp. 19–24.
  - [295] N. Alexiou, S. Gisdakis, M. Lagana, and P. Papadimitratos, “Towards a secure and privacy-preserving multi-service vehicular architecture,” in *IEEE 14th International Symposium and Workshops on a World of Wireless, Mobile and Multimedia Networks (WoWMoM '13)*, 2013, pp. 1–6.
  - [296] N. Bibmeyer, J. Petit, and K. M. Bayarou, “CoPRA: Conditional pseudonym resolution algorithm in VANETs,” in *10th Annual Conference on Wireless On-demand Network Systems and Services (WONS '13)*, 2013, pp. 9–16.
  - [297] *Road vehicles - Vehicle to grid communication interface - Part 1: General information and use-case definition*. 2013.
  - [298] C. Höfer, J. Petit, R. K. Schmidt, and F. Kargl, “POPCORN: Privacy-preserving charging for eMobility,” in *First Workshop on Security, Privacy and Dependability for CyberVehicles (CyCar '13) at 20th ACM Conference on Computer and Communications Security (CCS '13)*, 2013, pp. 37–48.
  - [299] T. Zahariadis, E. Ladis, H. C. Leligou, P. Trakadas, C. Tselikis, K. Papadopoulos, and S. Voliotis, “Trust Models for Sensor Networks,” in *50th International Symposium ELMAR-2008*, 2008, no. September, pp. 10–12.
  - [300] T. Zahariadis, H. C. Leligou, S. Voliotis, S. Maniatis, P. Trakadas, and P. Karkazis, “An Energy and Trust-aware Routing Protocol for Large Wireless Sensor Networks,” pp. 216–224.
  - [301] N. Kuntze, C. Rudolph, and J. Paatero, “Establishing Trust between Nodes in Mobile Ad-Hoc Networks,” in *Trusted Systems*, vol. 7711, C. J. Mitchell and A. Tomlinson, Eds. Springer Berlin, Heidelberg, 2012, pp. 48–62.
  - [302] A. Oberle, A. Rein, N. Kuntze, C. Rudolph, J. Paatero, A. Lunn, and P. Racz, “Integrating Trust Establishment into Routing Protocols of Today’s {MANETs},” in *Wireless Communications and Networking Conference (WCNC '13)*, 2013, pp. 2369–2374.
  - [303] A. Neumann, C. Aichele, M. Lindner, and S. Wunderlich, “Better approach to mobile ad-hoc networking (B.A.T.M.A.N.),” 2008.
  - [304] A. Karjalainen and J. Kangasharju, “On Interactions between Routing and Service Discovery in Wireless Sensor Networks.”
  - [305] A. Fagiolini, G. Valenti, L. Pallottino, G. Dini, and A. Bicchi, “Local Monitor Implementation for Decentralized Intrusion Detection in Secure Multi-Agent Systems,” in *2007 IEEE International Conference on Automation Science and Engineering*, 2007, pp. 454–459.
  - [306] A. Fagiolini, F. Babboni, and A. Bicchi, “Dynamic distributed intrusion detection for secure multi-robot systems,” in *2009 IEEE International Conference on Robotics and Automation*, 2009, pp. 2723–2728.
  - [307] M. Garcia-Otero, F. Alvarez-Garcia, and F. J. Casajus-Quiros, “Securing Wireless Sensor

- Networks by Using Location Information,” in *2009 16th International Conference on Systems, Signals and Image Processing*, 2009, pp. 1–4.
- [308] U. Zurutuza, E. Ezpeleta, Á. Herrero, and E. Corchado, “Visualization of Misuse-Based Intrusion Detection: Application to Honeynet Data,” in *Soft Computing Models in Industrial and Environmental Applications, 6th International Conference SOCO 2011*, 2011, vol. 87, pp. 561–570.
  - [309] M. Fiore, E. C. Casetti, C. Chiasserini, and P. Papadimitratos, “Discovery and Verification of Neighbor Positions in Mobile Ad Hoc Networks,” *IEEE Trans. Mob. Comput.*, vol. 12, no. 2, pp. 289–303, 2013.
  - [310] N. Bißmeyer, J. Njeukam, J. Petit, and K. M. Bayarou, “Central misbehavior evaluation for VANETs based on mobility data plausibility,” in *9th ACM international workshop on Vehicular inter-networking, systems, and applications (VANET ’12)*, 2012, pp. 73–82.
  - [311] S. Dietzel, J. Petit, G. Heijenk, and F. Kargl, “Graph-based Metrics for Insider Attack Detection in VANET Multi-hop Data Dissemination Protocols,” *IEEE Trans. Veh. Technol.*, vol. 62, no. 4, pp. 1505–1518, 2013.
  - [312] R. Wouter van der Heijden, S. Dietzel, and F. Kargl, “SeDyA: Secure dynamic aggregation in VANETs,” in *6th ACM conference on Security and privacy in wireless and mobile networks (WiSec ’13)*, 2013, pp. 131–142.
  - [313] C. Intanagonwiwat, D. Estrin, R. Govindan, and J. Heidemann, “Impact of Network Density on Data Aggregation in Wireless Sensor Networks,” in *22nd International Conference on Distributed Computing Systems (ICDS ’02)*, 2002, pp. 457–458.
  - [314] C. Castelluccia and C. Soriente, “ABBA: A Balls and Bins Approach to Secure Aggregation in WSNs,” in *6th International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks and Workshops (WiOPT ’08)*, 2008, pp. 185–191.
  - [315] P. Schaffer, I. Vajda, L. Buttyán, P. Schaffer, and I. Vajda, “CORA: Correlation-based resilient aggregation in sensor networks,” in *Ad Hoc Networks*, 2007, vol. 7, no. 6, pp. 373–376.
  - [316] M. Sivrianosh, D. Westhoff, F. Armknecht, and J. Girao, “Non-Manipulable Aggregator Node Election Protocols for Wireless Sensor Networks,” in *5th International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt ’07)*, 2007, pp. 1–10.
  - [317] T. Holczer and L. Buttyan, “Anonymous Aggregator Election and Data Aggregation in Wireless Sensor Networks,” *Int. J. Distrib. Sens. Networks*, vol. 2011, 2011.
  - [318] A. Francillon and C. Castelluccia, “Code Injection Attacks on Harvard-Architecture Devices,” in *15th ACM conference on Computer and Communications Security (CCS ’08)*, 2008, pp. 15–26.
  - [319] A. V Taddeo and A. Ferrante, “A Security Service Protocol for MANETs,” in *6th IEEE Consumer Communications and Networking Conference (CCNC ’09)*, 2009, pp. 1–2.
  - [320] J. Petit, M. Feiri, and F. Kargl, “Spoofed Data Detection in VANETs using Dynamic Thresholds,” in *3rd IEEE Vehicular Networking Conference (VNC 2011)*, 2011, pp. 25–32.

- [321] A. Reiter, G. Neubauer, M. Kapfenberger, J. Winter, and K. Dietrich, "Seamless Integration of Trusted Computing into Standard Cryptographic Frameworks," *2nd Int. Conf. Trust. Syst. (INTRUST '10)*, pp. 1–25, 2011.
- [322] M. Diaz, D. Garrido, and A. Reyna, "One Step Closer to the Internet of Things: SMEPP," in *Future Internet Symposium FIS:2009*, 2009.
- [323] "Device Profile for Web Services specification for Java (DPWS4J)." [Online]. Available: <https://forge.soa4d.org/projects/dpws4j/>.
- [324] C. Timm, J. Schmutzler, P. Marwedel, and C. Wietfeld, "Dynamic Web Service Orchestration applied to the Device Profile for Web Services in Hierarchical Networks," in *ICST/IEEE 4th International Conference on Communication System Software and Middleware*, 2009.
- [325] K. E. Kjær, "A Survey of Context-Aware Middleware," in *25th conference on IASTED International Multi-Conference: Software Engineering*, 2007, pp. 148–155.
- [326] W. Zhang and K. M. Hansen, "An OWL/SWRL based Diagnosis Approach in a Pervasive Middleware," in *20th International Conference on Software Engineering and Knowledge Engineering (SEKE '08)*, 2008, pp. 893–898.
- [327] A. Fiaschetti, F. Lavorato, V. Suraci, A. Palo, A. Tagliatela, A. Morgagni, R. Baldelli, and F. Flammini, "On the Use of Semantic Technologies to Model and Control Security, Privacy and Dependability in Complex Systems," in *Computer Safety, Reliability, and Security*, vol. 6894, Springer Berlin, Heidelberg, 2011.
- [328] M. M. R. Chowdhury and J. Noll, "Securing Critical Infrastructure: A Semantically Enhanced Sensor Based Approach," in *2nd International Conference on Wireless Communication, Vehicular Technology, Information Theory and Aerospace & Electronic Systems Technology (Wireless VITAE)*, 2011, pp. 1–5.
- [329] J. Noll, Z. Iqbal, and M. M. R. Chowdhury, "Integrating Context- and Content-Aware Mobile Services into the Cloud." 2011.
- [330] G. Dini and I. M. Savino, "A Security Architecture for Reconfigurable Networked Embedded Systems," *Int. J. Wirel. Inf. Networks*, vol. 17, no. 1–2, pp. 11–25, Jun. 2010.
- [331] P. Langendoerfer, S. Peter, K. Piotrowski, R. Nunes, and A. Casaca, "A Middleware Approach to Configure Security in WSN," in *1st ERCIM Workshop on eMobility, in conjunction with WWIC 2007*, 2007.
- [332] O. Derin, E. Diken, and L. Fiorin, "A Middleware Approach to Achieving Fault Tolerance of Kahn Process Networks on Networks on Chips," *Int. J. Reconfigurable Comput.*, vol. 2011, 2011.
- [333] A. Heinig, M. Engel, F. Schmoll, and P. Marwedel, "Using Application Knowledge to Improve Embedded Systems Dependability," in *Workshop on Hot Topics in System Dependability (HotDep '10)*, 2010.
- [334] G. Nunes, A. Cardoso, A. Santos, and P. Gil, "Multi-Agent Topologies over WSANs in the Context of Fault Tolerant Supervision," in *Technological Innovation for Sustainability*, vol. 349, L. M. Camarinha-Matos, Ed. Springer Berlin, Heidelberg, 2011, pp. 383–390.
- [335] G. Nunes, A. Cardoso, A. Santos, and P. Gil, "Multi-Agent Based Architecture for Robust

- Supervision over Wireless Sensor Networks,” in *9th Portuguese Conference on Automatic Control (Controlo 2010)*, 2010.
- [336] J. Barbarán, M. D’iaz, I. Esteve, D. Garrido, L. Llopis, and B. Rubio, “A Real-Time Component-Oriented Middleware for Wireless Sensor and Actor Networks,” in *1st International Conference on Complex, Intelligent and Software Intensive Systems (CISIS ’07)*, 2007, pp. 3–10.
  - [337] D. Slamanig and M. Pirker, “A Framework for Privacy-Preserving Mobile Payment on Security Enhanced {ARM TrustZone} Platforms,” in *IEEE 11th International Conference on Trust, Security and Privacy in Computing and Communications*, 2012, pp. 1155–1160.
  - [338] D. Derler, K. Potzmader, J. Winter, and K. Dietrich, “Anonymous Ticketing for NFC-enabled Mobile Phones,” in *Trusted Systems*, vol. 7222, L. Chen, M. Yung, and L. Zhu, Eds. Springer Berlin, Heidelberg, 2012, pp. 66–83.
  - [339] M. Pirker, D. Slamanig, and J. Winter, “Practical Privacy Preserving Cloud Resource-Payment for Constrained Clients,” in *Privacy Enhancing Technologies*, vol. 7384, S. Fischer-Hübner and M. Wright, Eds. Springer Berlin, Heidelberg, 2012, pp. 201–220.
  - [340] M. Pirker, J. Winter, and R. Toegl, “Lightweight Distributed Heterogeneous Attested Android Clouds,” in *Trust and Trustworthy Computing*, vol. 7344, S. Katzenbeisser, E. Weippl, L. J. Camp, M. Volkamer, M. Reiter, and X. Zhang, Eds. Springer Berlin, Heidelberg, 2012, pp. 122–141.
  - [341] M. Pirker, J. Winter, and R. Toegl, “Lightweight Distributed Attestation for the Cloud,” in *2nd International Conference on Cloud Computing and Services Science (CLOSER ’12)*, 2012, pp. 580–585.
  - [342] S. Nadjm-Tehrani and M. Vasilevska, “Towards a Security Domain Model for Embedded Systems,” in *13th IEEE International Symposium on High-Assurance Systems Engineering (HASE ’11)*, 2011, pp. 180–181.
  - [343] Y. Zhang, B. Hamid, and D. Gouteux, “A Metamodel for Representing Safety LifeCycle Development Process,” in *6th International Conference on Software Engineering Advances (ICSEA 2011)*, 2011, pp. 550–556.
  - [344] B. Hamid, J. Geisel, A. Ziani, and D. Gonzalez, “Safety Lifecycle Development Process Modeling for Embedded Systems -- Example of Railway Domain,” in *Software Engineering for Resilient Systems*, vol. 7527, P. Avgeriou, Ed. Springer Berlin, Heidelberg, 2012, pp. 63–75.
  - [345] A. Ziani, B. Hamid, and J.-M. Bruel, “A Model-Driven Engineering Framework for Fault Tolerance in Dependable Embedded Systems Design,” in *38th EUROMICRO Conference on Software Engineering and Advanced Applications (SEAA ’12)*, 2012, pp. 166–169.
  - [346] B. Hamid, S. Gürgens, C. Jouvray, and N. Desnos, “Enforcing S & D Pattern Design in RCES with Modeling and Formal Approaches,” *Model Driven Eng. Lang. Syst.*, vol. 6981, pp. 319–333, 2011.
  - [347] S. Trujillo, I. Alonso, B. Hamid, D. Gonzalez, M. Blanco, and H. (Yulin) Zhang, “Towards variability support for security and dependability patterns,” in *Proceedings of the 15th International Software Product Line Conference on - SPLC ’11*, 2011, vol. 2, p. 1.

- [348] C. Jouvray, M. Sall, and A. Kung, "Enforcing trust in embedded systems using models," in *Proceedings of the International Workshop on Security and Dependability for Resource Constrained Embedded Systems - S&D4RCES '10*, 2010, p. 1.
- [349] B. Hamid, N. Desnos, C. Grepet, and C. Jouvray, "Model-based security and dependability patterns in RCES," in *Proceedings of the International Workshop on Security and Dependability for Resource Constrained Embedded Systems - S&D4RCES '10*, 2010, p. 1.
- [350] F. Krichen, B. Hamid, B. Zalila, and M. Jmaiel, "Towards a Model-Based Approach for Reconfigurable DRE Systems," pp. 295–302, 2011.
- [351] A. Groll, J. Holle, C. Ruland, M. Wolf, T. Wollinger, and F. Zweers, "OVERSEE - A Secure and Open Communication and Runtime Platform for Innovative Automotive Applications," in *7th Embedded Security in Cars Workshop (ESCAR 2009)*, 2009.
- [352] A. Groll, J. Holle, M. Wolf, and T. Wollinger, "Next Generation of Automotive Security: Secure Hardware and Secure Open Platforms," in *17th ITS World Congress*, 2010.
- [353] N. McGuire, A. Platschek, and G. Schiesser, "OVERSEE - A Generic FLOSS Communication and Application Platform for Vehicles," in *12th Real-Time Linux Workshop*, 2010.
- [354] G. Griessnig, I. Kundner, E. Armengaud, S. Torchiaro, and D. Karlsson, "Improving Automotive Embedded Systems Engineering at European Level," *E I Elektrotechnik und Informationstechnik*, vol. 128, no. 6, pp. 209–214, 2011.
- [355] G. Pedroza, M. S. Idrees, L. Apvrille, and Y. Roudier, "A Formal Methodology Applied to Secure Over-the-Air Automotive Applications," in *IEEE Vehicular Technology Conference (VTC Fall)*, 2011, pp. 1–5.
- [356] D. Knorreck, L. Apvrille, and P. de Saqui-Sannes, "TEPE: A SysML Language for Time-Constrained Property Modeling and Formal Verification," *ACM SIGSOFT Softw. Eng. Notes*, vol. 36, no. 1, pp. 1–8, 2011.
- [357] A. Lackorzynski and A. Warg, "Taming Subsystems Capabilities as Universal Resource Access Control in L4," in *2nd Workshop on Isolation and Integration in Embedded Systems (IIES '09)*, 2009, pp. 25–30.
- [358] M. Kost, J.-C. Freytag, F. Kargl, and A. Kung, "Privacy Verification using Ontologies," in *6th International Conference on Availability, Reliability and Security (ARES '11)*, 2011, pp. 627–632.
- [359] A. Kung, J. Freytag, and F. Kargl, "Privacy-by-design in ITS applications," in *IEEE International Symposium on World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, 2011, pp. 1–6.

## ANNEX A – EMBEDDED SYSTEMS SECURITY

Embedded systems (ESs) permeate our lives in various forms, ranging from avionics to e-textiles, automobiles, home automation and wireless sensor nodes. In terms of their physical size, they range from miniature wearable or sensor nodes (i.e. motes) to large industrial deployments of programmable logic controllers (PLCs).

The various intrinsic and application-specific characteristics of ESs complicate the task of guaranteeing the security, namely handling the confidentiality, integrity and availability aspects of their applications and the data they handle. Their characteristics habitually include resource constraints (namely computational capabilities, storage capacity, memory and power), dynamically formulated, remotely-managed networking and even unattended operation in hostile environment and time-critical applications. Therefore, while securing networked computer systems is not a novel concern, the techniques developed for personal and enterprise systems are often unsatisfactory or even inapplicable to embedded devices.

In addition to the above, ES applications often feature direct interaction with the physical world, being responsible for vital, time-critical applications, where a delay or a speed-up of even a fraction of a second in system's response or reaction could have dire consequences. This further differentiates ES security, as a security incident in a critical application may lead to asset damage or even personal injury and death.

Moreover, next-generation ES services, like the ones pertaining to the Internet of Things (IoT), may require the integration of multiple administrative domains (e.g. one domain may host the devices and enable access to devices and information, whereas another domain may make use of the information for designing innovative services). Each domain will typically have its own security requirements and constraints, therefore ensuring their security [170], while maintaining interoperability [171], is a challenging task.

This section aims to provide an overview of the challenges in designing secure embedded systems, covering both node hardware and software issues, as well as relevant network protocols and cryptographic algorithms. In the next subsections we present physical security issues that are evident in embedded systems, the various access control mechanisms used for controlling access to resources. Indicative examples of cryptographic mechanisms specially-crafted for embedded systems and various protocol and management issues. Moreover, through a survey of pertinent EU-funded research efforts, recent advances in the field are identified, highlighting opportunities for future research.

### PHYSICAL SECURITY ISSUES

Regarding the physical layer and given the often unattended nature of deployed ESs, sometimes within hostile environments, the risk of device tampering should not be ignored. In the remainder of this section, some aspects of physical security in ESs are presented.

#### *SIDE CHANNEL ATTACKS*

A malicious entity's physical access to a non-tamper-resistant device, apart from providing physical access to the system components, would also enable the launch of various attacks. These include micro-probing and reverse engineering or sophisticated side-channel attacks

(SCA), timing attacks, simple power analysis (SPA), differential power analysis (DPA), as well as their electro-magnetic counterparts, SEMA and DEMA respectively, and also differential fault attacks (DFA). The aforementioned methods can potentially expose critical information concerning the operation of the device (algorithms used, length of keys etc.) which could prove critical to the security both of the device itself and the network as a whole.

In the case of ESs utilizing field-programmable gate arrays (FPGAs), the concept of run-time reconfiguration [172] can be explored to reduce component count and/or power consumption, increase fault tolerance etc. as needed. Self-reconfigurability can, for example, make a node more secure against side-channel attacks through the measurement of electromagnetic radiation and also implement self-healing properties. Self-recovery mechanisms could reallocate functional blocks to mark and replace faulty resources, through device reprogramming in the case of self-reconfigurable nodes or through controlled degradation of service techniques in less “intelligent” devices. Furthermore, the above-mentioned side channels could also have a positive aspect, as research has shown they can be exploited to verify the integrity of embedded software and to control the execution flow [173], [174].

#### *TRUSTED PLATFORM MODULE*

A Trusted Platform Module (TPM, [175]), is a microcontroller that can securely store a relatively small amount of information on it, which can then be used either for authenticating the platform (e.g. passwords, certificates, encryption keys) or for ensuring that the platform has not been breached (e.g. platform measurements, configuration data). Still, the microcontroller does not control the software that is running on the platform itself. Instead, through the tamper-resistant security functions it provides, the platform’s operating system or any running applications access the necessary information, to determine and implement their security policies accordingly. The software embedded on such a TPM component is directly related to the component’s physical size (the higher the memory requirements, the larger the module’s surface) and consequently cost. Any optimization of the module’s software will have a direct impact on the overall execution speed, as well as to the power consumption (faster execution allows the module to return to a more power sparing idle/sleep state).

#### *PROTECTION OF POWER SUPPLY*

Several types of embedded systems devices are mostly battery-powered, something that creates issues of energy constraints. Especially in cases where e.g. small sensor nodes are meant to be used in unattended environments, they are expected to operate for certain time intervals (sometimes spanning over a few months) until they have their power source replaced or recharged. An attacker could therefore launch a Denial-of-Service (DoS) attack, aiming at draining the battery power by forcing extensive use of the device’s wireless connection or CPU.

In order for the electronic parts of the embedded device to function properly, continuous power is required, with both voltage and current levels lying within specific limits. The power source should also be able to monitor its own state and react accordingly in cases where an issue is detected that could affect the normal operation of the system. Moreover, suitable fail-

safe mechanisms should exist, implemented in software and/or hardware that would protect the device and prevent any potential damage from spreading across the rest of its components.

Most of the requirements mentioned above are satisfied in modern uninterruptible power supply (UPS) systems; nevertheless, they cannot easily be applied to the operating environments of some embedded systems, due to strict size and cost constraints. Instead, alternative solutions are implemented, such as energy scavenging, super-capacitors, micro-solar cells and remote/wireless power transferring schemes [176], [177], to name a few. Still, such solutions must carefully be adopted to each specific scenario (e.g. a micro-solar cell is useless if the device cannot be reached by enough direct sunlight), also taking into consideration fail-safe options with respect to the criticality of the possible failures and the probability for them to occur, so as to protect the device effectively.

## ACCESS CONTROL

Access control mechanisms are essential to prevent unauthorized/malicious entities to access the resources, physical or otherwise, available to the ESs as well as the hosting devices. The way access control is implemented varies depending on the hardware capabilities of the nodes, the type of network and the application under consideration. Some common methods include:

1. Profile authentication: If a node has some specific characteristics (e.g. hardware specifications, operating system), it can join an existing network.
2. Access code: Demonstrating knowledge of the code grants access to the network and its resources. This code can either be programmable or configurable. This category includes typical password access, based on memory data, switch configuration or any other procedure.
3. Predefined topology: Only pre-established nodes can join the network (e.g. MAC filtering).

There is ongoing research on ES-specific access control protocols since the commonly-used authentication schemes, typically password-based, can be impractical or even insecure when considering the heterogeneous nature of ES networks can demonstrate and the scalable remote manageability often required [178]. Moreover, even in wired embedded networks and in industries such as automotive and aviation, most control networks utilized (e.g. Controller Area Network, Time-Triggered Protocol, FlexRay) are designed with safety and reliability in mind and do not feature any built-in security mechanisms like node authentication, data encryption or prevention of DoS attacks [179], [180], leading to critical vulnerabilities [8].

ESs are often deployed in applications bound by strict security requirements (e.g. e-Health applications are a prime example), including secure transmission of sensitive data to remote entities, instructions that need to reach actuators in an unaltered form, robust entity authentication and access control mechanisms [138].

Regarding the latter, among the proposed schemes that have gained popularity are those where decisions are made based on policy restrictions. Such a scheme is the standardized by OASIS eXtensible Access Control Markup Language (XACML), an XML-based general-purpose



policy decision language. Besides being used for representing authorization and entitlement policies for managing access to resources, it provides a processing model for evaluating requests and making decisions based on the defined set of policies [181]. Policy-based access control allows dynamic decision making on controlled nodes' resources based on policy restrictions set by the system owner.

Still, unprotected policy messages would expose the system's security, revealing private information to attackers who might also try to identify policy restrictions and do a mapping of the security measures taken for the specific environments, hence exploiting potential vulnerabilities. Moreover, in a more active approach, an attacker might masquerade as a legitimate entity or modify policy-related messages, such as authorization requests and/or decisions, obligations or advices in an attempt to downgrade adopted measures and bypass access controls. To avoid the aforementioned problems, appropriate security measures, like the ones detailed in later subsections, have to be deployed to safeguard message confidentiality, integrity and authentication.

#### *DENIAL OF SERVICE (DOS)*

The aim of a DoS or a Distributed DoS (DDoS) attack is to harm the availability of a specific node or a network of nodes, thus preventing or delaying legitimate entities from accessing the services or resources they wish to [182]. DoS attacks on ES nodes can take multiple forms, such as exploiting the vulnerabilities in their software/firmware, or attacking the network they belong to by jamming, misrouting, flooding and so on. DoS attacks can be mounted more or less on every layer, by exploiting the particular characteristics or mechanisms found in them. Their aim is to cause the nodes to constantly process or send dummy data, thus draining their power source via unnecessary use of their wireless connection and prolonged demand for memory and CPU cycles. The effects of such attacks are particularly critical in the case of nano and micro/personal nodes, where the power reserve is usually rather limited. In addition, flooding types of DoS attacks consume part of the network's bandwidth, which may also indirectly affect the normal operation of the network or a significant part of it, depending on the overall network's capacity.

Another type of DoS attack involves the case where an attacker gains physical access to the device and modifies or destroys it as a physical entity. Depending on the role of this particular node for the rest of the network or cluster of nodes it belongs to, its unavailability could have a significant impact on these entities. For instance, it could lead to partial or total loss of the data sink, selection of non-optimal routes, or even the loss of a control node that is vital for the normal operation of the system.

Given that large-scale networks are most probably heterogeneous in their nature, they contain different capabilities and vulnerabilities, which need to be addressed independently for achieving effective protection against DoS attacks. What is more, in cases where there is provision for dynamic network size variation, shielding against DoS attacks can become a very challenging task [183]. In addition, the problem becomes even more complex and difficult to solve for cases where the available resources and capabilities exhibit strict limitations.

In cases where the ES network is of an unattended nature, the use of a remote management system is vital. Nevertheless, such systems offer additional attack surface and their compromise can allow an adversary to upload malicious firmware. In this way, the attacker is able to corrupt memory and data sent/received or even lead to permanent damage of hardware sub-components by intended misconfiguration (Permanent DoS – PDoS, or *bricking*). The success of such attacks is based on the fact that the various firmware upgrade mechanisms are usually insecure and do not employ complex security mechanisms for verifying authenticity and integrity, such as the ones used in digital certificates. This process of rendering a device unbootable or non-reflashable is also known as *phlashing* [184].

One of the main reasons for the DoS attacks being relatively easy to successfully launch is the use of old protocols that suffer from lack of security requirements. For instance, the IP protocol takes for granted various assumptions regarding the trust of network nodes and consequently does not dispute the related information found in the packet headers and/or payload. Any integrity-checking mechanisms are rather primitive and simple in nature (e.g. checksums), as their aim is to detect accidental data corruption and not deliberate modification of information. Therefore, continuing to use such protocols as the base for building custom network communication protocols on it makes it particularly hard to design (D)DoS-resilient systems and services. An additional obstacle is the fact that basic software methodologies do not take into consideration security requirements, able to deal with such kinds of attacks [185].

The majority of the aforementioned attacks can be avoided by employing suitable authentication, access control and integrity-checking mechanisms. It is also equally important to provide secure mechanisms for node firmware deployment and software updates, able to verify both the authenticity and integrity of the firmware/software to be uploaded and reject it if it does not pass the required checks. Furthermore, more recent network protocols should be used, able to provide means and metrics for quantifying (D)DoS attack resilience [186]. The use of intrinsically secure ES firmware offering various fail-safe mechanisms or even hardware redundancy could be employed in cases where dependability is highly critical (e.g. avionics [187] and the military), as they are expected to increase the cost of the end-product.

## CRYPTOGRAPHIC MECHANISMS

As has already been mentioned, embedded devices often have inherent limitations in terms of processing power, memory, storage and energy. Efficient algorithm designs and implementations that adhere to these constraints, while satisfying application demands, can significantly impact battery lifetime and allow the implementation of many applications.

Key management is an equally important issue, both from a security and a management point of view. The rather simple pre-shared key (PSK) scheme, where every embedded device has the necessary cryptographic keys pre-installed, is difficult to manage in distributed and dynamic environments (physical access to the device is required) or in cases where there is a large number of such devices. Moreover, the disclosure of the master key leads to the instant compromise of all the system/network. Having an appropriate scheme that triggers periodic re-keying limits the amount of ciphertext that has been encrypted with the same key, thus increasing the system's security level.

This section presents several lightweight cryptographic and key management schemes, suitable for resource-constrained devices.

#### *LIGHTWEIGHT CRYPTOGRAPHY*

Embedded devices often have inherent limitations in terms of processing power, memory, storage and energy. The cryptographic functionality that ESs utilize to provide tamper resistant hardware and software security functions has direct impact on the systems:

- Size: Memory elements constitute a significant part of the module's surface.
- Cost: Directly linked to the surface of the component.
- Speed: Optimized code provides results faster.
- Power Consumption: The quicker a set of instructions is executed, the quicker the module can return to an idle state or be put in sleep mode where power consumption is minimal.

Traditional cryptography solutions focus in providing high levels of security, ignoring the requirements of constrained devices. Lightweight cryptography (LWC) is a research field that has developed in recent years and focuses in designing schemes for devices with constrained capabilities in power supply, connectivity, hardware and software (e.g. RFIDs, sensor nodes, contactless smartcards, and mobile devices). There has already been significant effort on the subject of crypto optimization, aiming to maintain the security level that "traditional" algorithms and implementations offer while narrowing what is often referred to as "battery gap" [188], i.e. the very high energy consumption overheads for supporting security on battery-constrained systems. A number of surveys [189]–[193] provide an overview of this subject.

Schemes proposed include hardware designs, which are typically considered more suitable for ultra-constrained devices, as well as software and hybrid implementations for lightweight devices.

- Hardware designs implement the exact functionality without redundant components. The main design goal is the reduction of the logic gates that are required to materialize the cipher. This metric is called Gate Equivalent (GE [194]). A small GE predisposes that the circuit is cheap and consumes little power. For constrained devices an implementation including up to 3000 GE can be considered acceptable while for even smaller devices, like 4-bit microcontrollers, implementations of 1000 GE are being studied. Energy consumption and power constraints are other significant factors. Energy consumption is important when a device is running on batteries while power constraints affect passive devices, like passive RFID tags, that must be connected to a host device to operate. Security attacks and relevant countermeasures that are correlated to power analysis are also considered in hardware designs.
- Software implementations typically only require a microprocessor to operate. The main design goals are the reduction of memory and processing requirements of the cipher. Implementations are optimized for throughput and power savings. Portability is their main advantage over hardware implementations.

Hybrid schemes combine the two approaches exploiting the best features from both. Hardware implements the basic cipher functionality and software performs the data and communication manipulation. A common practice is the design of cryptographic co-processors. The throughput is mostly affected by the communication bandwidth between hardware and software components. Hybrid implementations target on specific communication applications, like RFID tags, portable devices and Internet servers.

ISO/IEC 29192 includes ciphers and cryptographic mechanisms for LWC. The standards are PRESENT [194] and CLEFIA [195] for block ciphers, and TRIVIUM [196] and Enocoro [197] for stream ciphers.

Compact implementations of “traditional” ciphers, like AES [198], are also applied to embedded devices. Newer lightweight block ciphers are Humminbird-2 [199], Piccolo [195], SIMON and SPECK [200], ITUbee [201]. For stream ciphers, the finalists of the eSTREAM project [202], Grain, Salsa20, Rabbit and HC128 are suitable for LWC.

Hash functions design is another area where further research is required. For LWC, compact implementations of the standardised functions SHA-2 [203], SHA-3 [204] are examined. Newer functions that are suitable in this domain are Blake [204], Photon [205], SPONGENT [206] and other hash functions based on lightweight block ciphers, like DM-PRESENT [203].

Asymmetric algorithms and protocols must also be adapted to operate on devices with the afore-mentioned resource limitations. This is an elaborate task, since asymmetric ciphers are computationally far more demanding than their symmetric counterparts and are usually executed on powerful hardware. The performance gap is exacerbated on constrained devices, such as 8-bit microcontrollers. Even an optimized asymmetric algorithm (e.g. elliptic-curve cryptography – ECC), performs 100 to 1000 times more slowly than a standard symmetric algorithm (e.g. AES), which correlates to a proportionally higher power consumption.

In terms of practical relevance, two families of established public-key algorithms stand out: ECC and NTRU [207]. ECC in particular is considered the most attractive option in ESs, due to its small operand length and its relatively low processing requirements. NTRU is the most popular lattice-based cryptosystem. Its security is based on the shortest vector problem and it can efficiently be deployed on embedded systems. Compared to ECC in hardware, NTRU is 1.5 times faster with only the 1/7 of the memory footprint. Compared to RSA in software, it is 200 times faster in key generation, almost 3 times faster in encryption and 30 times faster in decryption.

The need for lightweight cryptography introduces major multi-dimensional challenges in cryptographic algorithms design, from the ES operating system (OS) to the hardware and software cryptographic provisions embedded on the device itself. Hardware and software co-design seems to offer the best results in terms of speed/size ratio for many ubiquitous computing applications [190]. Regarding primitives that cannot yet be effectively implemented (e.g. hashes in the case of crypto and public key crypto in the case of asymmetric), alternatives could be investigated so that the protocols which are based upon them can be researched further and perhaps put into practice. Special care should be taken

during the development of optimized implementations so that they do not introduce new leakage channels which could be exploited by Side-Channel Attacks (SCA).

A comparative analysis of lightweight ciphers on embedded systems was performed recently, where the authors evaluate the proposed schemes based on performance metrics and classify them for various types of embedded devices [208]. Various cryptographic libraries exist that offer key establishment mechanisms and communication protocols, like for example ULCL [209], wolfSSL<sup>11</sup> and OpenSSL<sup>12</sup>.

#### *KEY DISTRIBUTION MECHANISMS*

Key distribution, either for initialization or re-keying [210] has been a challenging topic especially for dynamic, heterogeneous and resource-limited environments. The majority of these schemes is based on symmetric mechanisms, thus requiring pre-distribution of the shared secret with all the disadvantages already discussed. Other schemes [211] are being proposed as well, some of which feature location-aware and identity-based mechanisms. Although some of the proposed schemes are indeed energy efficient [212], key management based on shared secrets has proven ineffective, especially in dynamically-formulated infrastructures. There have been attempts to correlate key establishment techniques to applications but these were based solely on the use of symmetric keys and on a framework level [213].

This has led part of the research community to focus its attention on public-key schemes (e.g. ECC). This enables the distribution of authentic public keys via insecure channels, as the verifying party does not need to have a copy of the secret key. Therefore, a mobile node's key database may, for example, be updated with all valid public keys once, according to a pre-defined schedule or ad hoc, and from that point onwards the device will be able to authenticate other entities in off-line mode.

Identity-based cryptography – IBC [214] provides an alternative solution to the very expensive, for many nodes, public-key cryptography. IBC allows publicly-available information that can uniquely identify participating nodes to be utilized for the secure exchange of keys. Thus, nodes do not have to depend on a public key infrastructure and digital certificates for the exchange of authenticated public keys. The advent of identity-based schemes and pairing-based cryptography has shown that such schemes offer a promising solution for managing keys in resource-constrained nodes [215].

Even with a public-key scheme, we still need to implement symmetric cryptography. Thus, alternative schemes that are based only in symmetric cryptography are also proposed [216], [217]. Some of them are location-aware and identity-based mechanisms and are energy-efficient. Their disadvantage is the pre-distribution phase that is required prior to first usage. The schemes are inefficient in dynamic environments but are proposed in applications where public-key schemes cannot be used.

---

<sup>11</sup> <https://www.wolfssl.com/wolfSSL/Home.html>

<sup>12</sup> <https://www.openssl.org>

## NETWORK PROTOCOL AND MANAGEMENT ISSUES

Certain applications of embedded systems, like Wireless Sensor Networks, rely on the integrity of the platform for providing trustworthy services (e.g. measurements taken by a sensor). It is therefore essential to have a method for validating this integrity and assuring that system components have not been compromised. The integrity of the service-requester platform, i.e. control node, must also be validated before allowing it to allocate resources to the nodes it controls or receive the data these nodes have collected. In addition, it should be established that these secure resource management mechanisms will not act as a bottleneck in service performance. Examples of current research on the subject are the WS-Attestation mechanism [218], which enables Trusted Platform Module (TPM) remote platform attestation using web services.

### *SECURE RESOURCE MANAGEMENT*

Inspecting the problem from a higher level, middleware resources should be managed by monitoring their availability, enforcing a policy based on which of these resources are assigned, implementing a secure model for the identification and authorization of requests, as well as an accounting system to track resource usage. Most of the above can be found in protocol Diameter [219], successor to RADIUS, which offers strong authentication, authorization, accounting and resource management mechanisms. Diameter is already adopted by many IP systems like in the 3rd Generation Partnership Project (3GPP).

### *REPUTATION-BASED SCHEMES*

Reputation-based schemes are a novel paradigm for enhancing security in various applications, including secure routing and intrusion detection systems for Mobile Ad Hoc Networks – MANETS [220]. These systems are easy to implement, lightweight and can protect a MANET from a wide variety of attacks.

The basic concept is inspired from social behavior and relies on the cooperation of the nodes. Much like human interaction, each entity decides to trust or ignore a new, unknown entity based on the opinion of its peers about the individual in question. Consequently, much like in social networks, trustworthy behavior is encouraged. The three main goals identified [221] for reputation systems are:

- To provide the required information in order to distinguish between a trustworthy principal and an untrustworthy one.
- To encourage principals to act in a trustworthy manner.
- To discourage untrustworthy principals from participating in the service.

Reputation-based mechanisms are used in Intrusion Detection Systems (IDSs) and provide two main functionalities: Secure routing and resource management. Watchdog and Path-rater [222] are the building blocks of such systems. Watchdog is a monitoring component and based on its observations Path-rater ranks the available routing paths. The main steps in the reasoning process of a reputation-based scheme are as follows:

1. Gather information.
2. Score and rank.

3. Select entity.
4. Transaction.
5. Reward and punish.

Misbehavior detection and intrusion detection can either be distributed, where information about entities' reputation changes are immediately broadcast to the whole network (or local), in which case each entity decides, based solely on its own data about the reputation of other nodes. It should be noted, however, that the latter is not as effective in terms of speed in detecting and isolating of malicious nodes. Known reputation-based systems for secure routing are IRmIDS [223], Reputated-ARAN [224] and CSRAN [225].

While secure routing provides P2P protection, reputation-based systems for secure resource management provide end-to-end protection. Such systems rank resources, providers and consumers. Based on these values, network entities are able to recognize legitimate providers and resources of good quality while keeping out selfish and malicious users. Several reputation-based schemes have been proposed for resource management in P2P and Grid networks [226].

#### *ANONYMITY AND LOCATION PRIVACY*

Location-based applications constitute a rapidly expanding market, owing to the widespread use and advances in mobile devices and positioning systems alike. Enhanced reality applications and other similar services are starting to emerge and are expected to spread in the coming years. Other examples of such smart services include location-aware emergency response, enhanced entertainment and/or advertisement services, or even location monitoring of personnel and fleets of vehicles.

The location of individual users is necessary in order to enable the abovementioned services but, even though its disclosure may not pose a security risk for the embedded device itself, said information constitutes sensitive personal data of the user or users associated with each device and should be handled accordingly. Disclosure of such information can enable a malicious user to harass, blackmail or even enter the individual's residence (e.g. when he/she is away). There is on-going research on the subject, including mechanisms for safeguarding location privacy [227], [228] as well as reports on the weaknesses of current "sanitization" mechanisms [229].

Literature on this topic includes variations and enhancements of a few recurring methods. Anonymization methods aiming to remove identifying information using generalizations and suppressions are the most popular in the literature, with  $k$ -anonymity being the basic mechanism used, as described in [228]. The principle of  $k$ -anonymity involves the use of a cloaking area, where there are at least  $k$  users in it, and blurs their identities in order to make each user's identity indistinguishable from the rest  $k-1$  users. In general, there are two important and mostly unavoidable trade-offs when choosing this  $k$ -value: A trade-off between privacy and quality of service and a trade-off between privacy and personalization [230].  $K$ -anonymity-based schemes have typically being deployed for privacy-preserving location monitoring and to allow mobile users to take advantage of location-based personalized services without compromising their privacy [231], [232].

Elementary anonymity schemes (e.g. a pure  $k$ -anonymity scheme) are inadequate if used independently and hence they are often combined with auxiliary mechanisms, in order to achieve better privacy safeguards [229], [233]. Attempting to further enhance these basic schemes, the use of personalized  $k$  values is proposed, for systems with context-sensitive privacy requirements [227]. Moreover, if combined with  $l$ -diversity [234], another dimension can be added to  $k$ -anonymity, where  $l$  is a set of distinct locations. Other than the above suggested improvements, the use of dummy locations [235] and semantic information [236] are also proposed in the literature, in order to address issues that are not solved by plain  $k$ -anonymity mechanisms.

Pseudonym-based methods are another common anonymisation theme, involving disposable pseudonyms for each node in the location service [237]. As the pseudonyms change over time, they are being used as temporal identifiers, making it hard for potential attackers to track the users. Silence periods can be introduced to further enhance this concept, as detailed in [238]. Other literature schemes rely on path perturbation, i.e. trying to cross paths in areas where at least two users meet [239].

#### *SECURE SERVICE DISCOVERY, COMPOSITION AND DELIVERY PROTOCOLS*

Services in distributed networks must be discovered, composed and delivered in a secure way. The Organisation for the Advancement of Structured Information Standards (OASIS) has released related standards including WS-Security [124], WS-Policy [240], WS-Trust [241] and WS-Secure Conversation [131] which have already been approved and the current trend is to bring web services into ESs and it is thus imperative to adopt the aforementioned specifications.

For this purpose, OASIS developed two standards: Devices Profile for Web Services (DPWS [23]) and Web Services Dynamic Discovery (WS-Discovery [94]), which specify the use of web-services-based communications in resource-constrained and ad hoc environments. The profile's architecture includes hosting and hosted services. A single hosting service is associated with each device while the same device may accommodate various hosted services. The latter represent the device's various functional elements and rely on the hosting service for discovery. Discovery services are included as well, enabling devices to "advertise" their presence on the network and search for other devices. Metadata exchange services provide dynamic access to services hosted on a device and their meta-data. Furthermore, publish/subscribe eventing services allow other devices to subscribe to messages provided by a certain service.

Additional research on this area has been conducted by the Service-Oriented Architecture for Devices (SOA4D[242]) open-source initiative which facilitates the development of service-oriented software components adapted to the requirements of embedded devices. Web Services for Devices (WS4D) is another open source initiative, providing a number of toolkits aimed at developing DPWS-compliant applications for resource-constrained devices in ad-hoc networks, maintaining interoperability with regular W3C-specified Web Services. A detailed overview of the WS4D initiative can be found in [146].

#### *COMMUNICATIONS SECURITY*



Embedded nodes have quite a few choices regarding the protocols they adopt in their communication stack, depending on their computation capabilities and needs. One of the predominant solutions is the 6LoWPAN [243] stack, i.e. IPv6 over 802.15.4 [244]. Such an approach benefits from the adoption of well-known and standardized solutions for providing security at the network layer. This is IPsec which, however, has to utilize compressed header format [111], [112] to fit into the limited message space provided by IEEE802.15.4, i.e. for low-power, low data rate wireless communication standard for small devices.

The network layer, however, is not the only layer in the TCP/IP stack where messages can be protected. The others are application and data link layer. The corresponding security mechanisms for these two layers are the Datagram Transport Layer Security – DTLS protocol [245], which is based on the well-known TLS (Transport Layer Security [115]) but uses datagram protocols and the inherent security mechanism of IEEE802.15.4 which is defined in the same standard. All these mechanisms were designed to provide at least confidentiality, integrity and message authentication, yet they have slightly different properties.

Security mechanisms found at the bottom layers of the communication stack relieve applications from deploying their own distinct security mechanisms. However, this comes at a cost. With regards to the IEEE 802.15.4 security protocol which protects messages at the data link layer, protection takes place on a node-by-node basis. This introduces significant computational overhead to the nodes which have to decrypt incoming messages, verify their integrity, and re-encrypt them, typically using a different set of keys, prior to forwarding them to the next node. This process consumes valuable node resources on routing nodes.

As opposed to 802.15.4, IPsec offers end to end protection of messages. Therefore, intermediate routing nodes' resources are only used for routing packets and not for message protection. In this sense, security provided at the network layer can be considered as a valuable mechanism for low power and lossy networks.

IPsec can be either used within such a network to secure communications among participating nodes in cases where these are deployed in a hostile environment to secure communications among participating nodes, or between a node and remote party. This second choice requires utilizing a gateway, e.g. sink node, which can simply forward messages or set up a tunnel to further protect messages and also provide communicating node details and traffic flow confidentiality. As an example, consider the secure remote access to an aircraft's device to control it in case of an emergency. The aircraft's gateway can be used to further protect the message and hide the addresses of communicating entities, while padding can conceal communication patterns and characteristics.

Compared to IPsec, DTLS demonstrates similar characteristics, in terms of end-to-end message protection, but it suffers from an expensive handshake mechanism and the inability to cope with applications that utilize the TCP protocol.

One of the problems one should consider when deploying one of these three solutions, is the use of robust key management mechanisms designed for resource-constrained devices. Traditional public key cryptography solutions are considered inappropriate in some

environments and alternatives, including those that utilize lightweight cryptography, elliptic curves and identity based cryptography should be considered.

## ANNEX B – PERTINENT EU-FUNDED RESEARCH

This section gives an overview of the research efforts in some recent EU-funded projects related to embedded systems security, following a layered approach. In particular, the first subsection presents the various technologies related to embedded systems' nodes. The second subsection deals with the network-related technologies, while the third subsection presents the approaches followed for the middleware and overlay layers. The final subsection focuses on architectures and frameworks, as well as on the formal validation of the security of embedded systems.

Embedded systems security is a recurring theme in current research efforts, brought in the limelight by the wide adoption of ubiquitous devices. Significant funding has been allocated to various European projects on the subject of embedded systems security, in order to investigate and overcome the various security challenges. A survey was conducted to provide an overview of recent EU research efforts pertaining to embedded systems security, where several prominent security issues and the respective proposed approaches are presented. Twenty such projects that focus on embedded systems security aspects were identified and the investigated technologies were categorized using a layered approach, to facilitate the presentation of the results; the categories comprise the node, network and middleware & overlay layers, as well as architectures, frameworks and formal validation of the security of embedded systems.

The survey presented here aims at providing an overview of said past and running projects in order to identify emerging trends, state-of-the-art technologies being used or developed, opportunities for composing or expanding past work and, generally, highlight open issues that need to be addressed in the future. In addition, this survey partly demonstrates the broader areas of embedded systems security that researchers chose to focus on, given the latest technological advancements (nationally-funded projects have not been taken into account).

Out of a large number of projects initially gathered, a smaller subset of the most recent ones was selected to be included in this work, based on their relevance to security and dependability aspects of embedded systems design, as well as their availability of public deliverables and publications lists. Research in these projects has been conducted using EU resources, hence they have a budget within the funding limits imposed by the EU itself and have undergone a similar review process in terms of novelty, application and quality requirements. In addition, they all try to achieve the common goal of attaining a uniform technological level among all EU state-members. Details on the EU-funded projects related to embedded systems security that were selected for the purposes of this survey are presented in Table 10.

**TABLE 10. SELECTED EU-FUNDED PROJECTS RELATED TO EMBEDDED SYSTEMS SECURITY.**

#	Acronym	Project Title	Start Date End Date	Cost (EUR)	Call
1	AETHER	Self-adaptive embedded technologies for pervasive computing architectures	01/01/2006	5.92M	FP6

			31/12/2008		
2	AWISSENET	Ad-hoc PAN and wireless sensor secure network	01/01/2008 28/02/2010	3.10M	FP7
3	CESAR	Cost-efficient methods and processes for safety relevant embedded systems	01/03/2009 01/03/2012	33.5M	ARTEMIS JU
4	CHAT	Control of heterogeneous automation systems: Technologies for scalability, reconfigurability and security	01/09/2008 31/08/2011	3.58M	FP7
5	EVITA	E-safety vehicle intrusion protected applications	01/07/2008 31/12/2011	5.89M	FP7
6	GINSENG	Performance control in wireless sensor networks	01/09/2008 29/02/2012	4.66M	FP7
7	HYDRA	Networked Embedded System middleware for heterogeneous physical devices in a distributed architecture	01/07/2006 30/06/2010	12.75 M	FP6
8	MADNESS	Methods for predictAble Design of heterogeNeous Embedded System with adaptivity and reliability Support	01/01/2010 31/12/2012	2.92M	FP7
9	MORE	Network-centric Middleware for group communications and resource sharing across heterogeneous embedded systems	01/06/2006 31/05/2009	2.75M	FP6
10	OVERSEE	Open VEhicular SEcurE platform	01/01/2010 30/06/2012	3.91M	FP7
11	PRESERVE	Preparing Secure Vehicle-to-X Communication Systems	01/01/2011 31/12/2014	5.44M	FP7
12	pSHIELD	Pilot Embedded Systems Architecture for Multi-layer Dependable Solutions	01/06/2010 31/12/2011	5.40M	ARTEMIS JU
13	SecFutur	Design of Secure and energy-efficient embedded systems for Future internet applications	01/05/2010 30/04/2013	4.20M	FP7

14	SEPIA	Secure, Embedded Platform with advanced Process Isolation and Anonymity Capabilities	01/06/2010 31/05/2013	3.26M	FP7
15	SMEPP	Secure Middleware for embedded peer to Peer Systems	15/09/2006 14/09/2009	4.46M	FP6
16	TECOM	Trusted Embedded Computing	01/01/2008 31/03/2011	9.02M	FP7
17	TERESA	Trusted computing Engineering for Resource constrained Embedded Systems Applications	01/11/2009 31/10/2012	3.79M	FP7
18	UbiSec&Sens	Ubiquitous sensing and security in the European homeland	01/01/2006 31/12/2008	2.91M	FP6
19	UNIQUE	Foundations for Forgery-Resistant Security Hardware	01/09/2009 29/02/2012	4.22M	FP7
20	WSAN4CIP	Wireless sensor networks for the protection of critical infrastructures	01/01/2009 31/12/2011	4.02M	FP7

The ubiquitous nature of embedded systems is evident in Table 9, which features the various application areas pertaining to each project. It should be noted that the application table was produced based on how the researchers themselves identify the application areas of the technologies they present, as this emerges from the project deliverables and publications. It goes without saying that many of the identified technologies could belong to other application areas as well, either with or without additional modifications.

**TABLE 11. APPLICATION AREAS OVERVIEW.**

Acronym	Aerospace	Automotive	Railway	Smart home and smart buildings	Smart metering	e-Health	Industry 4.0 and agriculture	Mobile devices	Critical infrastructure and environmental monitoring
AETHER		X		X		X	X		X

AWISSENET		X							X
CESAR	X	X	X				X		
CHAT	X								
EVITA		X							
GINSENG							X		
HYDRA				X	X	X	X		
MADNESS							X		
MORE						X			X
OVERSEE		X	X						
PRESERVE		X						X	
pSHIELD			X						
SecFutur				X	X				X
SEPIA								X	
SMEPP				X				X	X
TECOM		X				X	X	X	
TERESA		X	X	X	X		X		
UbiSec&Sens		X					X		X
UNIQUE					X				X
WSAN4CIP							X		X

In terms of the layers that were used to classify identified literature work, the lowest one is the node which involves the hardware and firmware technologies. The network layer includes various protocols, authentication schemes and other security-related mechanisms. Middleware layer mainly refers to low-level software that operates on top of the device's operating system but, in most cases, below any other applications (namely, overlay). Finally, the architectures and formalization classification comprises various frameworks and other holistic approaches to the security of embedded systems, including solutions that consider their formal validation.

## NODE TECHNOLOGIES

The heterogeneous nature of the field is evident from the literature review. In terms of hardware used, it was confirmed that there is a variety of platforms being utilised, with equally varied capabilities, such as the low-power TelosB, IRIS and MICAz platforms from Crossbow Technology,<sup>13</sup> the more capable Verdex Pro XL6P COM from Gumstix [246] and the FOX LX board from Acme Systems [247]. In some cases, even more powerful devices are being used, such as the Freescale i.MX51 [248] and the Xilinx Spartan-6 FPGA family [249]. The latter, along with low power x86-based platforms are also typically used in the development of future vehicular applications. Equally varied are the software security solutions being utilised and developed, featuring different operating environments, protocols and cryptographic primitives.

Given the often unattended nature of deployed embedded systems, sometimes within hostile environments, the aspect of physical security cannot be ignored. Gaining physical access to a device enables the launch of various side-channel attacks, such as simple/differential power analysis and differential fault attacks, which could potentially expose security-related information (cryptographic algorithms used, length of keys, etc.), thus jeopardising the security of both the device itself and the network it belongs to as a whole. What is more, the inherent limitations of embedded systems devices in terms of processing power, memory, storage and energy, require suitable cryptographic techniques that take those constraints into consideration. Such lightweight cryptographic mechanisms can facilitate secure communication without becoming a burden, resource-wise, on the device itself. Alternatively, virtualisation techniques can be used to fortify ESs security and specialised hardware modules can be employed to speed up various cryptographic functions.

This section is dedicated to presenting technologies aiming at protecting the embedded system's physical security, a variety of lightweight cryptography schemes and other techniques for enhancing a node's physical security that take into consideration the various resource constraints. An overview of the node-related technologies identified, can be found in Table 12.

**TABLE 12. NODE TECHNOLOGIES OVERVIEW (PROJECTS THAT DID NOT FOCUS ON THESE ASPECTS HAVE BEEN LEFT UNCHECKED).**

Acronym	Purpose-built hardware and features	Virtualisation	Lightweight crypto	Side-channel security issues	Trusted Platform Modules
AETHER	X	X			
AWISSENET	X		X		
CESAR					

<sup>13</sup> <http://www.moog-crossbow.com>

CHAT					
EVITA					
GINSENG					
HYDRA					
MADNESS					
MORE					
OVERSEE					
PRESERVE					
pSHIELD					
SecFutur					X
SEPIA	X	X		X	X
SMEPP			X		
TECOM		X			X
TERESA					
UbiSec&Sens			X	X	
UNIQUE	X				
WSAN4CIP				X	

## HARDWARE-RELATED SECURITY MODULES

### *TAMPER-RESISTANT MODULES*

A significant area of security research related to Wireless Sensor Networks (WSN) aims at utilising Trusted Platform Module (TPM) hardware and adapting it to the specific needs of resource-constrained applications. Such a TPM-related subject is that of the implementation of the Direct Anonymous Attestation (DAA) scheme specified by the Trusted Computing Group (TCG). In [250] a detailed report on the implementation of the aforementioned functionality is provided, as well as suggestions for improvements. The presented experimental results indicate that especially the rogue detection part of the DAA protocol can be very time consuming and the overhead is very evident on resource-constrained devices, increasing linearly with the size of the black lists of rogue TPMs. Moreover, problems with the mechanisms and protocols used to report compromised TPMs are identified. On the subject of TPMs, research has also focused on the security extensions of mobile platforms for hosting Mobile Trusted Module (MTM) functionality. Two different reconfigurable MTM architectures



are presented in [251]; the first one is based on a software implementation of the MTM running on the same physical processor as the applications using that MTM and the second is based on JavaCards providing the MTM functionality via the Java runtime environment, each with its own set of isolation mechanisms between the MTM and its users. The techniques utilise security features commonly found on mobile devices, i.e. Secure Elements and ARM TrustZone [252], proposing respective techniques for dynamic loading of TPM commands, aiming to alleviate the performance and memory issues arising from the security facilities of mobile platforms. In [253] the server side of Trusted Computing functionality is examined, presenting a design based on the Nizza Architecture [254] but minimising the trusted computing base and aiming to provide anonymous and trustworthy service for users, even counteracting certain insider attacks which, with the proposed scheme cannot go undetected.

An approach for protecting agents by utilising tamper-resistant cryptographic hardware is presented in [255]. The proposed agent migration protocol (Secure Migration Library – SecMiLiA) is based on the use of Trusted Computing technology that attempts to protect the agent from malicious hosts. A weakness of this system is the key management system that requires further improvement. In particular, due to the fact that the available key storage in the TPM is very limited, the key to be used is loaded into memory when required and is offloaded as soon as it is no longer useful, thus triggering many key transactions. Issues such as the use of key caching and the best possible management of cached keys remain topics that future research could deal with.

It should be evident from the above that TPMs are an important tool for building secure embedded system platforms; still, it must be noted that they should not be considered fail-proof. In [256] an active hardware attack on TPMs is detailed, which may not allow access to protected data (e.g. cryptographic keys), but circumvents the chain of trust assumed to be provided by the trusted platform. So, the module itself might be tamper resistant but the communication channels are often vulnerable and this is something that must be taken into consideration at the design phase.

Regarding defence against more invasive attacks, a clock frequency watch dog, implemented using a digital standard CMOS library, is presented in [257]. The proposed scheme is able to prevent clock speed manipulations, thus preventing side channel attacks on cryptographic hardware devices. The cost in terms of both additional area and energy requirements is low and is therefore suitable for being applied to low-cost devices, such as wireless sensor nodes.

#### *HARDWARE ACCELERATION*

Another approach to WSN node security is based on the use of low cost, low energy consumption Complex Programmable Logic Devices (CPLDs), which are programmable logic devices having a complexity between that of Programmable Logic Arrays (PLAs) and that of Field Programmable Gate Arrays (FPGAs), sharing architectural features with both. A WSN platform which embeds a CPLD in a standard WSN node is presented in [258]. As real-world experiments show, this CPLD-equipped platform can increase the performance of a standard WSN node by a factor of 1220 to 3000 when executing certain algorithms and also reduce power consumption, with a reported reduction of up to 98%. This concept is further expanded in [259], where various networking and security protocols are implemented on the

mentioned platform and real-world performance is compared to existing schemes. In [260] RESENSE is presented, a complete node platform integrating this technique on popular WSN nodes (MICAz and IRIS from Crossbow Technology) running the TinyOS operating system.

#### *PHYSICALLY UNCLONABLE FUNCTIONS (PUFs)*

The use of Physically Unclonable Functions (PUFs) is a method for protecting devices against attacks on their keys [261]. These functions extract secrets from physical characteristics of integrated circuits (ICs), which can be used, amongst others, for storing keys securely. The keys are therefore “hidden” into the various hardware parts, instead of being stored into the device’s memory. In this way, even by using very advanced tools for attacking hardware, any such attempts for side-channel attacks will be unsuccessful in retrieving any useful information. For an additional layer of security, Logically Reconfigurable PUFs (LR-PUFs) can be used that have the ability of changing their challenge/response behaviour in a random manner [262]. Hence, a potential attacker will also have to deal with a continually-changing behaviour. PUFs can be used in any embedded system comprising ICs, even in the very resource-constrained RFID tags, especially when the latter are used for high-security applications, such as passports.

Combining PUFs and Public PUFs (PPUFs) with Fuzzy Extractors, it is possible to substitute dedicated hardware security modules, as demonstrated in [263], [264], where these technologies are investigated in the context of pseudonymous communication in vehicular networks. There are various open issues in said field, as these anonymisation techniques are not really effective when every vehicle knows the PPUF characteristics and alternatives should be investigated (e.g. using lists of the PPUF characteristics themselves as pseudonyms). Moreover, using PPUFs in challenge-response and authentication mechanisms in general could be further investigated.

#### *CHANNEL CHARACTERISTICS EXPLOITATION*

A similar concept of exploiting physical characteristics in order to derive cryptographic keys, is the method presented in [265]. No dedicated hardware is being used in this case; instead, an adaptive quantisation algorithm is proposed, able to generate sufficiently long keys by exploiting the radio channel randomness between two communicating parties. In multipath radio environments, due to the scatters effects, the waveforms travel differently from one location to another. Hence, a potential eavesdropper is incapable of obtaining similar channel measurements and therefore cannot extract the secret key from the communicated data.

#### *VIRTUALISATION*

Virtualisation is a feature that, as research has shown, adds to the overall security of the system, in various ways. Firstly, it seems to be a remedy for facing the severe security challenges that mobile devices have, given that they are usually targeting a completely open setup [266]. In addition, efficient virtual machines have successfully been implemented in micro-kernel based systems, thus enabling the reuse of arbitrary operating systems [267]. The overhead imposed on the kernel growth was rather marginal and the overall performance was found to be similar to other virtual machine implementations. An analysis on how and to which degree recent x86 virtualisation extensions can influence the response times of a real-time operating system that hosts virtual machines was performed in [268]. In [269] it was

shown that a thin and rather simple virtualisation layer can add to the overall system's security, as it provides fewer options for attack to a potential adversary. What is more, this approach was found to exhibit significantly better performance, compared to contemporary full virtualisation environments. Finally, regarding the way virtual machines should be implemented, it is claimed in [270] that their construction should follow the principle of incremental complexity growth. Namely, additional functionality should not be included in the trusted computing base of a component if the benefits it offers are less than the drawbacks (e.g. due to larger risk for introduced bugs and errors). Such an approach can be efficiently implemented and it was possible to achieve high throughput and good real-time performance.

The utilisation of Trusted Platform Modules and virtualisation techniques is an emerging pattern in relevant EU projects. A combination of said technologies is presented in [251], intending to provide a reference design for a Trusted Computing-based, lightweight, virtualisation framework specifically aimed at cloud computing scenarios, an increasingly important area of applications. An overview of the proposed architecture, which exploits both the ARM TrustZone and TPM DAA technologies. Lightweight containers are supervised by a relevant supervisor application. In the proposed scheme lightweight containers ( $\mu$ compartments) are built on top of the Linux kernel, with each isolation container enclosing the code and data required for the compartment to operate. Each compartment is monitored and managed by a per-compartment supervisor application, responsible for constructing the security policies, enforcing them and finally destructing its compartment.

## LIGHTWEIGHT CRYPTOGRAPHY

An overview of the literature pertaining to time and energy overhead various cryptographic primitives impose on popular types of wireless sensor nodes is presented in [271]. A number of symmetric and public-key algorithms, hash functions and cryptographic primitives in general are mentioned as well as their lightweight counterparts, where available. It is worth pointing out that the node lifetime data presented in the literature usually refers to the overhead imposed by the security-related functionality alone and, in a real-life scenario, values would be significantly lower due to additional functions running on the same node.

In the literature, whenever strong encryption is required on rather resource-constrained devices, elliptic-curve cryptography (ECC) is always a strong candidate. In [272] the finite fields  $F_p$ ,  $F_{2^d}$  and  $F_{p^d}$  are being investigated for suitability for performing ECC on the ATmega128 microcontroller and it turns out that binary fields are most preferable when efficient implementations are required.

An interesting security scheme for WSN that provides transparent security is proposed in [273]. This scheme is effectively a lightweight CBC-X mode cipher that is able to provide encryption/decryption and authentication, combined as a one-pass operation. Consequently, it exhibits significant energy gains of about 50-60%, compared to TinySec [274]. Furthermore, the proposed scheme has no ciphertext expansion for the transmitted data payload, thus significantly reducing the communication overhead. Although a block cipher is used, ciphertext expansion is avoided by having padding rules making use of a Data Stealing technique and a MAC Stealing technique, thus allowing for zero redundant padding bytes.

A strong, compact and efficient block cipher, DESL (DES Lightweight extension), based on the DES (Data Encryption Standard) cipher design is proposed in [275]. Instead of using 8 S-boxes as in DES, it uses a single S-box repeated eight times, thus considerably reducing chip size requirements. Furthermore, a lightweight implementation of DESL is also proposed, that requires almost half the chip size and 86% less clock cycles compared to the best AES implementations targeted for RFID applications, therefore rendering DESL a strong candidate for ultra low-cost encryption applications.

An optimised implementation of a modular multiplication is presented in [276]. The proposed algorithm was tested on an 8-bit microcontroller (AVR), using an 160-bit standard compliant elliptic curve (namely, secp160r1). Given that the majority of the processing time for elliptic-curve cryptography (ECC) is spent on modular multiplication, related schemes such as EC ElGamal or ECDSA would greatly benefit from it, as well as their applications in the field of resource-constrained devices (such as WSNs).

Hardware-specific optimisations have also played an important role in lightweight cryptography research efforts. The authors in [277] present an area-efficient implementation of AES (requires  $0.33mm^2$  in a  $0.25\mu m$  technology), featuring good performance and low power consumption. These goals were achieved by both optimising individual functional blocks of AES, as well as the overall architecture.

## MISCELLANEOUS NODE TOPICS

The authors in [278] propose a scheme for implementing security on extremely low-cost sensors that run with minimal resources regarding computational power, energy consumption and memory size. The sensors are initially loaded with firmware suitable for providing asymmetric cryptography during the one-time bootstrapping phase. Then, through a dynamic code update, it is replaced by other security protocols that are required for the operation of the WSN, effectively offering hybrid security functionality. Their proof-of-concept implementation makes use of the FlexCup plug-in for TinyOS.

The practicality of group signature schemes on mobile devices is examined in [279], where the authors constructed a Java framework that allows for an in-depth evaluation of three such schemes (out of a total of seven defined in the upcoming ISO20008-2 standard). Performance evaluation took place on a laptop bearing an Intel i7 CPU, as well as on three recent Android-based smartphones, so as to gather up-to-date results. The conducted tests were aiming at determining the required signing time, as this is considered very important in the investigated scenarios. Initial results ranged from 304.2 to 4752.7 ms, among the three smartphones, for various algorithms and key lengths. However, when pre-computation was employed, the times dropped significantly and fell within the range of 0.71 to 631.11 ms, respectively. Verification times were significantly longer for the mobile devices (245.6 to 9735.5 ms), nevertheless still within acceptable limits for real-world implementations.

## NETWORK TECHNOLOGIES

The resource-constrained and often heterogeneous and distributed nature of embedded systems, imposes restrictions and introduces issues at the network layer as well. It is quite common that certain applications of embedded systems require the integrity of the provided

service. If web services are being used, it is important to be able to ensure the validity of each participating node, thus ensuring that the system has not been compromised and their communicated data (e.g. measurements) is trustworthy. This is the objective that the various attestation techniques try to achieve. One other issue that needs to be taken care of is the secure transmission of the obtained data to their destination. Examples for meeting this requirement are the implementation of secure routing or secure data aggregation techniques. Detection of potentially malicious nodes in a network is another important issue that may be achieved with the deployment of a suitable intrusion detection system (IDS). Such systems usually look for abnormalities in the overall system behaviour and raise alerts accordingly. Once again, all these additional security mechanisms should not burden the overall system's performance to an extent where the system is effectively rendered useless, therefore suitable lightweight techniques must be employed.

This section is dedicated to presenting technologies related to node attestation and authentication techniques, secure routing and secure data aggregation and various intrusion detection schemes. Table 13 features an overview of the network technologies identified and their related projects.

**TABLE 13. NETWORK TECHNOLOGIES OVERVIEW (PROJECTS THAT DID NOT FOCUS ON THESE ASPECTS HAVE BEEN LEFT UNCHECKED).**

Acronym	6LoWPAN and 802.15.4	Privacy and anonymity	NFC and RFID	Secure routing and secure services protocols	Intrusion and malicious node detection	Secure aggregation
AETHER				X		
AWISSENET	X		X	X	X	
CESAR						
CHAT					X	
EVITA						
GINSENG	X					
HYDRA						
MADNESS						
MORE						
OVERSEE						
PRESERVE		X		X	X	X

pSHIELD					X	
SecFutur				X	X	
SEPIA		X	X			
SMEPP						
TECOM						
TERESA						
UbiSec&Sens			X			X
UNIQUE		X	X			
WSAN4CIP						X

## NODE ATTESTATION AND AUTHENTICATION

The interoperability with existing infrastructures and the Internet is a major challenge which must be tackled in a definitive way if we are to realise what is often referred to as the Internet of Things (IoT). A very valuable tool in this area is the combination of the IEEE 802.15.4 standard with 6LoWPAN (IPv6 over Low power Wireless Personal Area Networks, [280]) which, expectedly, introduces new security challenges and opportunities. An example of the security challenges introduced by using these new technologies can be found in [281], where an off-the-shelf T-Mote Sky wireless sensor is transformed into an 802.15.4 packet sniffer. Analysis is then trivial using open source software like Wireshark. Of course, this technique is a valuable tool in the hands of researchers developing protocols but can also be exploited by malicious users to eavesdrop on a network or even launch active attacks (e.g. packet injection). The authors in [282] propose new compression mechanisms for 6LoWPAN security headers, along with cryptographic mechanisms typically used with the IP security architecture, allowing the establishment of end-to-end secure channels between internet hosts and sensor nodes. The proposed mechanisms also allow for fine-grained control over the energy consumed on security-related tasks on the nodes, while the proposed model was evaluated in [283], with AES/CCM and SHA1 as the cryptographic primitives of choice.

The security and constraints stemming from the limited resources of sensor nodes have been investigated in EU projects extensively. Such an EU-funded attempt at trying to tackle these issues is presented in [284], giving an overview of the topic, including security and operational requirements, sensor and network constraints as well as the objectives of this specific project. Another overview, more focused on smart-home applications, can be found in [285], where, among others, key privacy and security issues are identified.

## PRIVACY AND ANONYMITY

Anonymous Authentication and Anonymity schemes in general are another key area of current research, since privacy is essential in many applications (e.g. social, medical) and

anonymising access to resources and services is a common technique to safeguard users' privacy. An analysis of how trusted computing technologies can be used for anonymous authentication and how they can be integrated into common security frameworks (e.g. Java Crypto Architecture) can be found in [286]. This work is based on the DAA scheme for providing anonymity over secure communications channels (i.e. anonymous TLS client authentication), but using alternative, more lightweight, schemes than those defined in the TPM v1.2 specification. Another interesting aspect of this work are the discrepancies reported between various TPM manufacturers (e.g. Infineon, Atmel, Winbond, Intel, ST Micro), TPM emulators and the original specification.

Another anonymous authentication scheme based on an optimised version of DAA and aimed at resource-constrained mobile devices is presented in [287]. Functionality includes secure devices authentication, credential revocation as well as anonymity and untraceability of said devices against service providers. The proof-of-concept implementation was deployed on an ARM11-equipped development platform (exploiting the ARM TrustZone feature, using an elliptic curves and pairings scheme, while integration with the OpenSSL security framework was also demonstrated).

Further work on Trusted Computing Group anonymity schemes (i.e. PrivacyCA and DAA) is attempted in [288]. The goal is to overcome the need for a trusted third party which is evident in the aforementioned standard schemes, while maintaining compatibility with the TPM v1.2 specification. The proposed anonymisation scheme for trusted platforms overcomes the need for a trusted third party while, relying on the TPM's DAA functionality so that no TPM modifications are required.

Regarding anonymous authentication, a Direct Anonymous Attestation protocol utilising Near Field Communication-equipped (NFC, [289]) mobile devices and RFID is proposed in [290], expanding on the now relatively popular Secure Element (SE) scheme presented in [291]. Experimental results are also presented, using off-the-shelf mobile devices.

The scheme proposed in [292] offers anonymous authentication for RFID, with the use of additional devices, the *anonymisers*. The latter interact with the RFID tags and any communication with external devices (e.g. RFID tag readers) is performed through the anonymisers that mask certain information, thus ensuring the tags' anonymity and unlinkability. What is more, the anonymisers operate as some sort of proxies to the RFID tags, by undertaking the task of performing the required public-key cryptographic operations that the tags are unable to do so, due to their very resource-constrained nature.

As smartphones are already ubiquitous, some researchers focus on taking advantage of the features of modern smartphones in smart vehicle applications, as this alleviates some of the requirements from the vehicle platform itself (in terms of processing power, presence of GPS functionality etc.). In [293] the privacy issues of such an application, namely the use of smartphones for data acquisition of Intelligent Transportation Systems (ITS). The authors propose the extension of an existing architecture with anonymous authentication, allowing for privacy-aware traffic and location sample collection and protecting users' privacy even in cases of compromised ITS servers.

Wireless communication in Vehicular Ad hoc Networks (VANETs) is typically protected by digital certificates. As such certificates and related identifiers must not be usable to track vehicles, short-term pseudonymous certificates are applied and regularly changed in order to protect the driver's privacy. The authors in [294], [295] introduce and implement a distributed PKI architecture for vehicular networks, utilising pseudonymous certificates for privacy-preserving vehicular applications complying with related standards. Using tickets as cryptographic tokens, the proposed scheme offers authentication, authorisation and accountability, while maintaining the vehicle's privacy (e.g. guaranteeing that consecutive pseudonym requests cannot be correlated).

Nevertheless, there are certain incidents (e.g. traffic accidents) where it should be possible to identify the actual user via the certificate issuer. Hence, resolution of pseudonym identifiers is needed. The authors in [296] propose a generic pseudonym resolution protocol to be used by network infrastructure entities under such critical pre-defined conditions. The proposed protocol, CoPRA, does not increase pseudonym certificate size and imposes no additional overhead nor delay in the certificate acquisition phase. Moreover, it allows for validation of the situation that warrants the pseudonym resolution, prior to providing any information regarding the users' identity.

Privacy concerns pertaining to future smart vehicles are not restricted to VANET-related issues. The expected wide deployment of electric vehicles and charging infrastructures will require the use of protocols to control authentication, authorization and billing of vehicle owners. The ISO/IEC 15118 [297] standard defines the vehicle to charging station communication interface, also including the necessary security mechanisms. Still, it does not cater for the privacy protection of service users, making it trivial to, for example, let charging station operators track the location of a specific user. Therefore, authors in [298] propose POPCORN, a modular extension to the protocol defined in the abovementioned standard, which includes various privacy enhancing technologies like anonymous credentials. A proof-of-concept implementation is also presented to demonstrate the feasibility and investigate the performance of the proposed scheme.

## SECURE ROUTING

Secure routing protocols constitute another critical research area of networking technologies. In [299] an overview of security issues and current trends in trusted routing for ad-hoc networks is provided, evaluating their applicability in WSNs. Various trust-management enhanced routing protocols and trusted routing frameworks are investigated, focusing on their applicability on resource constrained environments. A secure routing protocol better suited to such environments is proposed in [300], namely Ambient Trust Sensor Routing (ATSR) and its performance and effectiveness is evaluated. In ATSR the geographical location of nodes along with other parameters (e.g. their remaining energy; for better load balancing and lifetime extension) are considered. Moreover, the protocol features a distributed trust model, based both on direct and indirect trust data, to detect malicious nodes.

The authors in [301], [302] also proposed a mobile ad-hoc network routing protocol based on the B.A.T.M.A.N. (Better Approach To Mobile Ad-hoc Networking) protocol [303] that utilises concepts from the domain of trusted computing. Device attestation is integrated on the



protocol itself, thus routing and data transmissions can be restricted to trustworthy devices only. A Trusted Platform Module (TPM) serves as root-of-trust on each device and hence, any devices that have been identified as being malicious can automatically be recognised by all network nodes, thus leading to their exclusion from the trusted network. Additional issues need to be looked into, such as interoperability with other network standards, interaction with other networks (homogeneous or not) and the maintenance of a trustworthy connection over another layer-2 protocol.

The interactions between secure routing protocols and the Service Discovery functionality on WSN networks where the nodes are used as service providers are investigated in [304]. Simulation results presented in the aforementioned work indicate that in some situations there is an efficiency gain if routing protocols allow the higher layers to override the routing decisions which might, for example, try to avoid using an untrusted node that the service discovery layer wants to use.

## INTRUSION AND MALICIOUS NODE DETECTION

Intrusion Detection Systems (IDS) are a key tool in safeguarding distributed ES networks. A dynamic and distributed IDS scheme is presented in [305] and further expanded in [306], where nodes act as local monitors of their neighbours and, in combination with data received from other monitors, are able to detect malicious entities. Simulations are used to prove the effectiveness of the proposed methods, with applications focusing mostly on smart vehicles. Defensive techniques for sensor networks based on the nodes' locations are surveyed in [307]; assuming every node is capable of detecting its own location. Furthermore, concepts of robust statistics (i.e. robust regression) are proposed, aiming to localise a node in the presence of malicious beacons. To facilitate the analysis and understanding of IDS data, various advanced methods have been investigated in EU funded products, including neural network-based techniques for the visualisation of said data, as presented in [308].

Awareness of nearby vehicles and their location is a basic foundation of electronic safety application in VANETs. However, the ad hoc nature of the vehicular networks makes them vulnerable to malicious nodes. Moreover, it is realistic to assume that in some cases there will be no pre-established trust relationships between vehicles. Researchers in [309] try to address these challenges by proposing a fully distributed cooperative solution, namely a lightweight protocol which relies only on information exchange among neighbouring entities, enabling the effective identification of adversarial nodes.

A central evaluation scheme is proposed in [310], where malicious peers are detected and excluded from the VANET using misbehaviour detection systems. These systems use trust and reputation information provided in misbehaviour reports submitted by vehicles as well as roadside units. As simulations indicate, the presented system is significantly effective against ghost/malicious vehicles broadcasting faked position and other information, which is one of the most critical attacks on VANETs. It should be noted that the proposed scheme is not fully distributed and vehicles rely on a central authority, namely the Misbehaviour Evaluation Authority, to detect attackers.

Even in cases where a PKI scheme is established, insider attackers, i.e. malicious entities possessing legitimate key material, must be considered as well. Authors in [311] propose the exploitation of redundant information dissemination for consistency checks and evaluate dissemination protocols using three graph-based metrics they introduce.

## SECURE AGGREGATION

Information aggregation techniques are a useful tool, especially in mobile ad-hoc networks (MANETs), to facilitate information dissemination and to reduce bandwidth requirements. In the context of VANETs, which are considered a sub-category of MANETs, the vehicles must communicate to exchange information for various enhanced services (safety, efficiency, traffic, entertainment etc.). Aggregation techniques can be used so that vehicles exchange high quality summaries of said information, instead of exchanging each individual message. Still, it is essential to utilise secure aggregation schemes, as the information exchanged between vehicles can be used of important decisions (e.g. traffic management, fleet control or road safety). Authors in [312] present such a dynamic and secure aggregation scheme for VANETs, which prevents insider attacks from influencing the aggregation results. Furthermore, its security mechanisms can be applied to existing aggregation schemes to produce dependable aggregates.

When it comes to Wireless Sensor Networks (WSNs), maximising the sensors' battery life is, naturally, of great importance. For this reason, *in-network aggregation* protocols have been proposed, where the required function(s) on the measurements is/are computed as data traverses the network [313]. One problem in such a scheme is that a corrupted sensor providing incorrect measurements cannot be distinguished from a sensor under attack, where the attacker has either modified the environmental conditions or has obtained the sensor's cryptographic secrets, in order to inject false measurements into the data sink. A novel secure data aggregation protocol is presented in [314] that is able to provide security, privacy and integrity for sensor networks, using inexpensive cryptographic tools. The main idea of the proposed ABBA (A Balls and Bins Approach) protocol is to define several *bins* for different *sensing intervals* and to demand each sensor to provide its sensed value adding one *ball* in the appropriate *bin*.

The aforementioned problem of deliberately-introduced corrupt data in an in-network aggregation protocol may be countered by exploiting the statistical properties found in the communicated data [315]. In particular, the naturally existing correlation between the readings produced by different sensors are taken into consideration to increase the resilience of data aggregation, without any special assumption on the distribution of the sensor readings, or the attacker's strategy.

Should the scheme involve the election of aggregator nodes, the authors in [316] discuss the requirements that need to be fulfilled, in order to have a non-manipulable aggregator node election protocol. Moreover, they provide a comparative review of three Secure Aggregator Node Election (SANE) protocols, based on a particular threat model.

A similar private aggregator node election protocol, where the election is performed in an anonymous manner, was proposed in [317]. The objective of this protocol was to make it

difficult for an adversary to identify aggregator nodes in the network and then physically compromise them. The protocol was deployed alongside with a private data aggregation protocol and a private query protocol, that masked the data flows to and from the aggregator nodes, thus maintaining their privacy. An enhanced version of the protocol was also proposed that is able to detect any misbehaviour within the network, such as the one introduced by an attacker who injects false reports. Nevertheless, further research is required for the system to be able to identify the misbehaving node.

## MISCELLANEOUS NETWORK TOPICS

Due to the very limited memory of certain types of devices (e.g. Harvard-based architecture devices, such as Mica motes), it was believed that these devices were immune to buffer overflow attacks that inject code into the stack and then execute it. Nevertheless, the authors in [318] demonstrated the feasibility of a remote code injection attack for Mica sensors, where the injected code is permanent, thus enabling the attacker to gain full control of the target sensor, persistently across reboots. What is more, they show how this attack can be transformed into a worm, namely how to make the injected code self-replicating and therefore able to propagate through the WSN, with the potential of eventually forming a sensor botnet. The employed techniques for this attack involve return-oriented programming and fake stack injection. It only suffices for the attacker to corrupt one network node and use its keys to propagate the malware to its neighbours. Packet authentication and cryptographic techniques in general can make such code injection attacks more difficult, nevertheless they cannot completely prevent them.

A security service protocol for MANETs, able to negotiate the security settings for the communications is presented in [319], a feature which is particularly useful in heterogeneous networks, both in terms of hardware and of services provided. This negotiation protocol aims at selecting the cheapest services that consume the least possible amount of energy, while offering the highest possible security level among nodes with different security requirements. In addition, run-time negotiation of services is supported, thus making it suitable for cases where self-adaptivity is involved. Nevertheless, the protocol is not yet complete and additional work is required on the message exchange for key management and errors.

The trustworthiness of messages received by peers is especially important in the context of e-vehicle safety-related applications (e.g. local danger warnings), as critical decisions often need to be made on those messages; decisions directly affecting passenger safety. To increase the trustworthiness in said messages, a consensus mechanism can be used, i.e. the same warning needs to be received at least  $x$  times from peers before it is considered legitimate. Researchers in [320] investigate this threshold and its effect on the decision delay, including the possibility of malicious peers launching information forgery attacks.

## MIDDLEWARE AND OVERLAY TECHNOLOGIES

Moving to higher layers, namely middleware and overlay, researchers have to tackle additional challenges as system complexity increases. On the other hand, operating from a higher level allows the utilisation of more advanced features, like the secure and efficient resource management (by aggregating information from the lower layers) and mechanisms to facilitate the interoperability and management of heterogeneous ES networks. It is, therefore,

of no surprise that middleware and overlay technologies are a common area of research and development efforts for many of the projects investigated, aiming to exploit the aforementioned tools in order to design secure embedded systems and related services.

This section is dedicated to presenting various types of middleware, such as trusted, service-oriented, context-aware, reconfigurable and fault-tolerant. Table 14 presents the middleware and overlay technologies as well as the attempts related to architecture and formalisation, which were identified in this work.

**TABLE 14. MIDDLEWARE AND OVERLAY TECHNOLOGIES OVERVIEW (PROJECTS THAT DID NOT FOCUS ON THESE ASPECTS HAVE BEEN LEFT UNCHECKED).**

<b>Acronym</b>	<b>Secure middleware</b>	<b>Services and overlay applications</b>	<b>Reconfigurability and fault tolerance</b>
AETHER			X
AWISSENET			
CESAR			
CHAT			X
EVITA			
GINSENG	X		X
HYDRA	X		
MADNESS			X
MORE	X	X	
OVERSEE			
PRESERVE			
pSHIELD	X		
SecFutur			
SEPIA	X	X	
SMEPP	X		X
TECOM	X		
TERESA			X

UbiSec&Sens	X		X
UNIQUE			
WSAN4CIP			

## TRUSTED MIDDLEWARE

Trusted Software is another important area of middleware layer research and [321] proposes a Trusted Software Stack (TSS – which acts as an interface between applications and a TPM) to be integrated into existing security framework, facilitating the adaptation to Trusted Computing technology. The prototype developed and proposed uses the .NET programming environment, taking advantage of the environment's fault-detection functionality (e.g. regarding buffer overflows), portability and developer base.

## SERVICE-ORIENTED MIDDLEWARE

The main features of a secure, service-oriented middleware for embedded peer-to-peer systems, in order to face the various security challenges of the Internet of Things (IoT) are presented in [322]. The notion of groups is used, as peers offer services inside groups and the discovery of these services is also performed within the group. Services can be state-less or state-full, and the latter ones may be session-less or session-full. The offered API allows for abstract peer and group management, as well as for events and message handling, features that facilitate application development within this environment. The presented service model and component-based middleware satisfies necessary principles such as security, heterogeneity, interoperability and scalability. The model was validated with two very different applications, including applications of WSN for monitoring radiation in nuclear power plants and for health-care in a mobile environment.

The deployment and orchestration of web services on heterogeneous embedded systems is another emerging research area and a task often assigned to the middleware layer, following the standardisation of the Devices Profile for Web Services (DPWS, [23]) open framework and research already conducted in the SIRENA project<sup>14</sup> and its follow-ups, SODA<sup>15</sup> and SOCRADES<sup>16</sup>. Some pervasive applications often require remote management and monitoring while maintaining interoperability, and the Web Services standard offers a solid basis for that. It is therefore justifiable that the runtime of the middleware developed for the MORE project<sup>17</sup> was based on the aforementioned DPWS specifications, as detailed in [41]. The DPWS4J [323] Java-based stack was extended and, to further facilitate development, the middleware is managed via the OSGi [78] modular service platform environment running on a Java Virtual Machine. Further enhancements were also introduced, enabling small footprint service orchestration in a DPWS-compliant environment [324]. The whole concept was validated on Gumstix Verdex XL6P embedded platforms.

<sup>14</sup> <http://www.sirena-itea.org>

<sup>15</sup> <http://www.soda-itea.org/>

<sup>16</sup> <http://www.socrades.eu/>

<sup>17</sup> <http://www.ist-more.org/>

## CONTEXT-AWARE MIDDLEWARE

An extensive overview of context-aware middleware, is presented in [325], categorising their properties and use. An ontology-based approach has been followed in [326], using the Web Ontology Language (OWL) and Semantic Web Rule Language (SWRL) in order to develop monitoring and diagnosis rules. In this way, any malfunctions can be detected and self-healing procedures can be invoked, in an effective, extensible and scalable way, as it was proved by the experimental results.

Enriching the relations between the different systems' parts with semantic information, as well as exploiting contextual process data, can yield useful information which can be fed into the various control and decision-making algorithms [327], [328]. Utilising the aforementioned concepts to enhance user profiling and trust sharing and to offer content and context-awareness for cloud-based services is also examined in [329]. The proposed model is based on a representation of the cloud service through semantic integration and ontologies for user profiles, trust, context and content. OWL restrictions are specified to guarantee access to trust-based context data.

## RECONFIGURABLE AND FAULT-TOLERANT MIDDLEWARE

The aspect of reconfigurability and its repercussions on security are considered from a higher-level perspective in [330]. A security architecture is proposed which, based on a middleware layer, offers secure reconfiguration and communication (i.e. SecComm component framework) with fine-grained application-specific policy enforcement, authenticated downloading from a remote source (i.e. ALoader component framework) as well as a re-keying service for key distribution and revocation (i.e. Rekeying component framework).

The scheme presented in [331] is a configurable and adaptive middleware, that aims at reducing the complexity of the realisation of an appropriate security level for a given WSN application. It consists of a modular middleware architecture which separates core functionality needed for adaptability support from pure security functionalities and also introduces the concept of a middleware compiler. A suitable configuration tool compiles a security architecture at development time and the architecture allows for dynamic exchange of security modules at run time. An initial set of security modules get configured before the deployment of the application; the application programmer then has to specify the security functionality that is required by the application, such as secrecy and authentication, as well as some additional information regarding the hardware platform of the sensors (processor type, memory size, etc.). Based on this information, the appropriate security modules are selected. In cases where either the application needs have changed or an update is required for facing a newly-detected vulnerability, security modules can be exchanged after deployment. Such functionality is particularly useful for long-living applications.

Middleware can also be used in Kahn Process Networks (KPN) implemented on a Network on Chip (NoC). In [332], a methodology for identifying requirements and implementing fault tolerance and adaptivity is presented. The overhead in terms of computational time and total data traffic can be lower than 10%, depending on the chosen bound of the connectors and the tokens' size being transferred at the application level. Since embedded systems exhibit a significant number of soft errors, their correction imposes equally significant hardware and

real-time overhead. For improving embedded systems' dependability, the authors of [333] proposed an approach that exploits application knowledge to classify errors according to their relevance and the impact of their correction to the system. Avoiding to correct every single error (effectively delaying the error-correcting process) caused a reduction in the imposed correction overhead, thus making it easier to meet mandatory deadlines in cases where real-time behaviour is an absolute requirement.

Fault monitoring and fault tolerant control for constrained sensor nodes is also examined in the GINSENG project [334], [335], wherein a multi-layered, middleware-based architecture is proposed. The scheme involves multiple agents implementing distributed artificial intelligence techniques for robust control over the wireless sensor nodes and also details the communication and coordination mechanisms involved.

In [336] a middleware called MWSAN is proposed that provides high-level services for Wireless Sensor and Actor Networks (WSANs), where the nodes are not only able to sense environmental data, but can also react by affecting the environment. It follows the component-oriented paradigm and it leaves it up to the developers to configure it according to the actor and sensor resources, by taking into consideration issues such as the network configuration, the quality of service and coordination among actors. Since actor nodes are usually more powerful than sensor nodes, the middleware features high configurability to match the diversity of requirements between these two types of nodes. For instance, the middleware for sensors does not include the various actor-related components, thus leading to a much smaller memory footprint. What is more, provision has been made for enabling the definition of real-time characteristics, in order to offer improved temporal behaviour, such as cases of priority schemas where the highest priority events are executed first.

## OVERLAY APPLICATIONS

Facilitating seamless online payments is another key issue researchers try to address. Such services often raise privacy concerns, and location-based services even more so. Privacy-preserving payment schemes are one of the main themes examined in the SEPIA project. Application scenarios involve end-users being equipped with mobile devices featuring ARM processors and TrustZone support [337], like NFC-equipped smartphones. An application of the aforementioned privacy-preserving mechanism on NFC-enabled smartphones is presented in [338]. The proposed method is based on selective disclosure protocols and experimental results on a standard JavaCard indicate a key of up to 1024 bits may be feasible. Utilisation of the ARM TrustZone features would be beneficial to the security and overall performance of the model, as would further support for lightweight cryptography (e.g. ECC) on the JavaCard.

Cloud-related scenarios are an associated theme where, for instance, privacy issues arise from the application of the split processing mode on mobile transactions. In such schemes, lightweight tasks are executed on end-user devices (e.g. smartphones, tablets), whereas more demanding tasks are offloaded to the Cloud. The proposed payment scheme utilises ARM's TrustZone and Intel's Trusted Execution Technology (TXT), assuming said support is present on both the client and cloud provider platforms and allows the end-user to take advantage of the cloud resources while the cloud provider is unable to track users' activity patterns [339].

Moreover, the authors in [340], [341] propose a node join protocol which, via remote-attestation, doesn't allow nodes with unknown configurations to join the cloud network, thus alleviating concerns for control over data and code execution on such networks. Proof-of-concept implementations are presented for the Android operating system, both on Intel and ARM-based platforms. Presented work assumes every node hosts a TPM which, in the case of the ARM platform, requires an add-on module to be installed. With the add-on module in place, the ARM prototype's security qualities were similar to that of the x86-based platform.

## ARCHITECTURES AND FORMALISATION

Embedded systems are usually the building blocks of a greater and more complex systems, created for a given purpose. A careful design of such architectures, as well as of their provided services would certainly have a positive effect on any security-related issues, by minimising unforeseen flaws and deficiencies. What is more, formalising the process of designing and building embedded systems would, in most cases, lead to an easier integration of the final system, while maintaining high levels of security and dependability.

In this section, such proposed frameworks and architectures are presented for different application areas: safety-critical applications, security and dependability applications, and smart vehicle applications. The pertinent efforts identified are presented in Table 15.

**TABLE 15. ARCHITECTURE AND FORMALISATION EFFORTS OVERVIEW (PROJECTS THAT DID NOT FOCUS ON THESE ASPECTS HAVE BEEN LEFT UNCHECKED).**

Acronym	Architectures	Formalisation
AETHER		
AWISSENET		
CESAR	X	X
CHAT		
EVITA	X	X
GINSENG		
HYDRA		
MADNESS		
MORE		
OVERSEE		X
PRESERVE	X	
pSHIELD		



SecFutur	X	
SEPIA		
SMEPP		
TECOM		
TERESA	X	
UbiSec&Sens		
UNIQUE		
WSAN4CIP		

Some approaches in current research focus on providing fully-featured frameworks and/or formalising the process of designing and developing secure and dependable embedded systems, especially in applications where safety is critical. In [342], the two distinct domains of embedded systems and security are considered, and an appropriate view of a final system model is provided, aiming to support cooperation between the two domains, while leaving them independent from each other. The proposed scheme is intended for on-demand provision of communication services in crisis-related situations, where different actors could be involved, also bearing heterogeneous client devices. The model consists of two components: The System Security Interface (SSI) that abstracts the system design model for communicating security needs and resource availability and the Security Building Block (SBB) that abstracts the implementation for a security mechanism.

In [343] a process metamodel is introduced which takes safety lifecycle requirements into consideration for secure software engineering (e.g. validation). This concept is explored further in [344], where a process metamodel, the Repository-Centric Process Metamodel (RCPM) is described. RCPM includes safety lifecycle concepts at its core and includes software tools for creating the required models, as well as a case study based on a railway application. Moreover, the authors in [345] present a model-based framework which focuses on formalising and managing fault-tolerance and redundancy concepts and which uses composable UML components to construct fault-tolerant infrastructures. A test case of a fault-tolerant GPS is evaluated using the aforementioned system. A similar model-based technique is used in [346] aiming to encode security and dependability patterns (S&D), while introducing artefacts for the formal validation of these patterns. Therefore, the fulfilment of S&D requirements identified at higher abstraction levels can be validated via the proposed process. The concept of S&D formalisation is further explored in [347], where the authors focus on the systematic reuse of S&D patterns in embedded systems where security and dependability are major concerns. To facilitate, automate and enforce fulfilment of S&D requirements, [348] defines a trust-aware platform-independent architecture, the TECOM architecture, as it was the outcome of the research project bearing the same name. An attempt to encode S&D patterns utilising meta-modelling techniques can also be found in

[349], while said work also includes an implementation of those patterns using a profiled UML and adapted to resource-constrained embedded systems. The goal is to help application developers integrate the application building blocks they typically use with security and dependability building blocks. Furthermore, the authors in [350] apply modelling techniques on reconfigurable systems; namely distributed real time embedded systems. The approach presented is called RCA4RTES and published work includes the case study of a Global Positioning System (GPS), where state machines describe the dynamic reconfigurations of the system.

With the widespread use of embedded systems leading to the Internet of Things, smart vehicles are another emerging and significant application. The potential new features and services available to vehicle occupants are, of course, numerous. Still, security and dependability is essential in this scenario and any compromise to the safety of vehicle occupants and other road users would not be acceptable. For example, updates could be exploited by an attacker to install malicious firmware during an over-the-air diagnosis and firmware update. The authors in [351] introduce the Open VEHiculaR SEcurE platform (OVERSEE), which aims to provide a standardised vehicular infrastructure with a protected runtime environment and on-board access and communication points. The proposed platform allows the integration of multiple Engine Control Units (ECUs) into one hardware node. It offers temporal and spatial isolation, a secure interface for connecting to external networks (e.g. the Internet) and also the required interfaces and open APIs to allow the secure download and execution of OEM and third party applications, much like the functionality offered by smartphones and their application “markets” [352], [353]. Part of the research in the field is more engineering-oriented in nature. The authors in [354] presents such an approach, as developed on project CESAR. Functional safety and tool-chain integration are the main challenges which researchers try to address by developing a reference technology platform. The work presented in [355] extends the safety-oriented environment AVATAR (a SysML modelling language framework) [356] with security constructs and verification techniques, to formally secure safety-critical automotive applications.

A capability-based, object-oriented software architecture is presented in [357]. Featuring a micro-kernel interface and enforceable security policies along with virtualisation provisions, it aims to improve security and provide isolation between multiple un-trusted software components.

The authors in [358] propose Privacy-by-Design, i.e. a systematic approach of integrating privacy requirements onto the design and implementation of a system. Using ontologies, a formal method is introduced which allows the evaluation of the system in terms of the realization of those pre-defined privacy requirements. The application of this method on the development of Intelligent Transportation Systems (ITS) applications is demonstrated in [359].

## IDENTIFIED OPEN ISSUES

The increased complexity and interconnection of the current systems’ components, as well as the varying and often undefined security levels of the networks they consist of, demand different approaches in the way the requirements are stated, in addition to the way these systems are designed. An integrated approach is required, where the components’ security

level is properly and systematically assessed, thus enabling the correct evaluation of the architecture's overall security level. In order for this to occur, reliable and useful metrics need to be defined, also applicable to legacy and therefore potentially insecure systems.

Furthermore, lightweight alternatives or improvements to existing cryptographic primitives and key distribution mechanisms could be looked into. Even though plenty of mechanisms and techniques already exist that would typically be deployed to secure other types of computing devices (e.g. for access control, cryptography, network routing etc.), they are not always applicable or have limited efficacy in the context of embedded systems.

The development of comprehensive cryptographic tools focused on embedded systems, featuring lightweight primitives, could be a very important development, including utilization of TPM functionality, and virtualization features, where available. Similarly, extending and improving the interoperability of existing, standardized, cryptographic mechanisms (e.g. IPsec) with new types of networks (e.g. 6LoWPAN deployments), would be desirable.

Wearable systems introduce more challenges, like developing the means to securely and seamlessly collect, store and transmit various data, some of which might be private sensitive in nature (thus having to consider regulatory compliance issues that arise when dealing with such data). Access to location-based services is commonly required in such applications, as well as various vehicular and smartphone-related smart services, which again raises various privacy concerns. This mandates the development of efficient anonymizing schemes, which must allow the user to access said services, while prohibiting the service provider from uniquely identify the specific user and her location among the rest of the users.

Future research is also expected to focus on revising the traditional role of middleware (namely, facilitating interaction and compositions via discovery and orchestration). By upgrading middleware technologies and transforming them into recommendation engines, able to dynamically and adaptively detect patterns and predict potential service interactions, embedded systems will better reflect the new crowdsourcing, social and generally human-related applications. These changes though are bound to introduce novel security and privacy issues that will have to be addressed.

The concepts of self-reconfiguration (e.g. in order to adapt to changes in the security levels, network, application/user requirements or location) and self-recovery (e.g. in fault conditions) could be investigated further. This can be achieved via on-the-fly hardware and/or software changes and can even be used to enhance the robustness of embedded systems against side-channel attacks (by controlling electromagnetic emissions etc.).

Moreover, there is room for improvement on the formalisation, definition and application of security and dependability (S&D) concepts. It is important to be able to formalise S&D requirements and product lifecycle in general, accurately modelling the processes from research and development until the end product. In this way, it will enable the validation of the end-product whether it meets all S&D and the other requirements defined at earlier stages.

As future work in the context of the survey presented in this section, it would be interesting to examine each of the sections (e.g. node) and their identified technologies separately and with respect to the current state of the art for each of the identified technologies of the section. Such a comparison would facilitate the evaluation of the results of EU-funded research efforts in a global context and help draw useful conclusions about the quality of the projects' output and the return on European investments in said research topics.