UNIVERSITÀ DI PISA

DIPARTIMENTO DI INFORMATICA

PHD THESIS

Usage Control in the Internet of Everything

Author

Athanasios Rizos

Supervisors

Fabio Martinelli
IIT-CNR

Andrea Saracino
IIT-CNR

February, 2020

To my family and all my friends for all their support
throughout the period of the preparation of this work.

“With your mind power, your determination, your instinct,
and the experience as well, you can fly very high.”
AYRTON SENNA

Abstract

The Internet of Things (IoT) is a world-wide network of heterogeneous interconnected objects, uniquely addressable, which are based on standard communication protocols and are able to interact with each other in order to share information based on communication protocol. Over the last years, IoT devices have become more and more pervasive to our daily life. Moreover, IoT is also one of the technological factors that enable the Internet of Everything (IoE) which includes not only *Things*, but also *People*, *Data* and *Processes*.

Security of IoT and IoE is a major aspect nowadays. The communication protocols that are used on this area lack of mechanisms that enforce policies continuously and control the access to the resources. In addition to the previous, the difference of the communication protocols in terms of architecture and characteristics has the effect that information sharing is difficult to be controlled via a single tool or mechanism.

Thus, there is the challenge of enhancing the security features of IoT protocols with a single mechanism so as to be able to cope with this aspect. For this reason, we present in this thesis a hierarchical Usage Control (UCON) model that is based on a distributed verion of the standard UCON model. UCON enhances Attribute Based Access Control (ABAC) models in two novel aspects: continuity of control, and mutability of attributes. Hence, we can control with this access requests inside complex IoE architectures. In order to demonstrate the viability of our approach, we present how UCON can be added on the most common communication protocols, how we can enhance the interoperability of UCON for different IoE services, and finally we present the hierarchical UCON accompanied by policy simplification methods.

Acknowledgement

I would like firstly to thank my supervisors for their guidance throughout the period of the PhD and the University of Pisa for the opportunity that they gave to me.

Furthermore, I would like to thank Prof. Pierpaolo Degano and Prof. Paolo Ferragina for their crucial help and guidance during this period. Also, I would like to thank the internal committee and the external reviewers of my thesis and all the people of the Department of Computer Science of the University of Pisa for their assistance.

I would like also to thank all my colleagues in the Institute of Informatics and Telematics (IIT) of the National Research Council of Italy (CNR).

At this point, I would like to thank the European Union and the NeCS European Project (H2020 - GA #675320) for the opportunity given to me, contributing on the research over this very important topic and my colleagues in NeCS for the collaboration we had.

Finally, I would like to thank my family, my friends and all the people I collaborated with for their continuous support and help.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The Internet of Things (IoT) is a world-wide network of heterogeneous interconnected objects, uniquely addressable, which are based on standard communication protocols [115], and are able to interact with each other in order to share information [44]. Over the last years, IoT devices have become more and more pervasive to our daily life. IoT has already, and will have in the future, a significant impact on several aspects of everyday life and behavior of users [43].

The number of connected devices has already passed the 20 billion milestone and we are heading to 50 billion devices by 2025 [32] [128]. This number becomes even more dramatic if we consider the Internet of Everything (IoE) paradigm, which also includes user devices such as smartphones, smartwatches, tablets, etc. [91]. Moreover, many new smart devices are going to enter our houses and replace the traditional ones, such as smart TVs, refrigerators, thermostats, ovens, lights, showers, and many others [104]. This is a paradigm that things are rapidly evolving in the area of modern wireless telecommunications [6]. Thus, the most obvious effects of the IoT shall be visible in both working and domestic fields.

The security predictions for the following years claim that IoT security is a top concern anticipating more vulnerabilities and attacks from cyber-criminals [12] [118]. They also claim that the devices that are sold are not secure by design and that patching of IoT devices has been proven to be difficult to manage [147]. Moreover, attacks on IoT devices has been found as both simple and destructive [157]. The main motivation of hackers is to hijack devices so as to create powerful botnets so that all the hijacked devices will be ready to act in unity.

In addition to the previous, IoT communication protocols, in their current form, lack of crucial security features and mechanisms [9]. Furthermore, the difference of the communication protocols in terms of architecture and characteristics has the effect that information sharing is difficult to be controlled via a single tool. Thus, there is the challenge of enhancing the security features of IoT protocols with a single mechanism so as to be able to cope with this aspect.

Furthermore, IoT is also one of the technological factors that enable the "IoE" which, according to Cisco[1], is the networked connection of not only *Things*, but also of *People*, *Data* and *Processes* [63]. This definition means the goal of connecting people in more relevant and valuable ways, converting *Data* into intelligence to make better decisions on information sharing, and delivering the right information to the right recipient [34].

Differently from IoT, where objects store the information produced in order to provide sharing or querying to them, in IoE, there is no duplicating of the information, but the values are extracted by filtering, aggregating, and merging the data originally created [160]. Undoubtedly, the IoE is already impacting the everyday life of many people and its importance is expected to rise in the next few years [115].

Moreover, apart from the IoT and the IoE, the term Cyber-Physical System (CPS) has been also introduced. This term does not refer only to the connectivity and coordination between computational and physical resources that are able to communicate with sensors like IoT. It also refers to the indication of monitoring and mostly to the usage of actuators that can modify both the cyber and the physical environment [142]. CPS is targeting to adaptability, autonomy, efficiency, functionality, reliability, safety, and usability [38] and combines different scientific fields such as cybernetics, design and process science, mechatronics [25]. Examples of CPS include smart grid, autonomous automobile systems, medical monitoring, process control systems, robotics systems, and automatic pilot avionics [69]. Summarizing, the previous lead to the fact that CPS is a part of IoE [110].

The IoE is an inherently complex ecosystem, due to the many different architectures and protocols used, and the different types of hardware and software present in IoT devices. Thus, securing the communications and operations of the IoE area has become of capital importance [112]. Besides, there are a lot of immature products and cheap devices being released in the market with insufficient security mechanisms.

As a result, several security issues have been identified, such as interception of communications, data compromise by unauthorized parties to collect Personal

[1] http://ioeassessment.cisco.com/

Identifiable Information (PII), authentication can be brute-forced, credentials can be extracted from device firmware, mobile apps or intercepted at login, and new firmware can be uploaded with malware [10]. IoT devices could be very different, because they typically have different types of hardware, depending on the provided functionalities, and software applications to manage them. Thus, it is of paramount importance to provide techniques that allow secure information exchange between IoT devices although it is very difficult to control the information exchange in IoT, due to the different architectures and characteristics of the communication protocols that are used in this area.

Considering the previous facts, it is necessary to have a unique mechanism that should make easier the information exchange of the different devices that are used in the same environment. But, in order to have a unique mechanism which eases the control of all the smart devices owned by the same user, a necessity has arisen to be able to easily communicate with a set of distinct IoT devices. To this aim, several protocols have been proposed in the scientific literature, and among them, Message Queue Telemetry Transport (MQTT)[2], which is recently standardized by OASIS[3] and Constrained Application Protocol (CoAP) are two of the most widely used [3] [10]. Both protocols were designed for IoT and Machine-to-Machine (M2M) applications.

On the one hand, CoAP was introduced as a standard by the Internet Engineering Task Force (IETF) in 2014 [131] and its key features are simplicity for constrained environments, very low overhead, and easy interoperability with Hypertext Transfer Protocol (HTTP). Similarly to HTTP, CoAP uses the REpresentational State Transfer (REST) model. Hence, CoAP uses a Client/Server communication pattern, in which *Servers* make resources available and *Clients* interact with resources using REST methods (Get, Post, Put, Delete). When a *Client* requests access to a certain piece of information, the access granting decisions are based on the optional use of the Datagram Transport Layer Security (DTLS) library, which provides the security-oriented features of CoAP. CoAP security mechanisms only rely on the optional use of DTLS for controlling the access and enforcing authorization policies.

On the other hand, MQTT works according to the Publish/Subscribe protocol pattern, where a central *Broker* handles the communications and data sharing, collecting data from a set of *Publishers* and redistributing them to a set of *Subscribers*,

[2]http://mqtt.org

[3]https://www.oasis-open.org/

according to their specified interests. The Broker is working as a middleman by receiving all the subscription requests and distributing the information published to every Subscriber interested. According to [141], the MQTT standard and the existing implementations, provide support only for basic authentication and simple authorization policies, applied to Subscribers at subscription time and enforcing such policies only in case of subscription of clients to message topics. While the spreading of IoT devices is widely and rapidly increasing, the problem of securing all those data from IoE is growing and getting more relevant [146]. Since MQTT is based on HTTP functionalities, most of the MQTT security solutions seem to be either application specific, or just leveraging Transport Layer Security (TLS)/Secure Socket Layer (SSL) protocols [3]. Currently, OASIS MQTT security subcommittee is working on a standard to secure MQTT messaging using MQTT Cybersecurity Framework [96].

Although the effort concerning security of IoT communication protocols is rising, two main obstacles occur. The first one is that, although the protocols have the ability to deal with various components that become Publishers/Subscribers or Clients/Servers, the fact that they use different platforms makes it difficult to create and enforce a generic security policy [68]. The second problem is that the current efforts are mainly directed to message communication security, to avoid eavesdropping, integrity violation and Man In The Middle (MITM) attacks [21]. Still no efforts have been done in the directions of supporting policy enforcement at Broker level, nor it has been considered the possibility of dynamically revoking subscriptions. MQTT security subcommittee has not provided any result yet on how authorized part of the MQTT ecosystem can remain authorized for all the duration of usage of a service [96].

Summarizing the previous, in addition to mobility, security is a high requirement for IoT and IoE. But, the connection of the cybernetic and the physical world means that vulnerabilities in terms of security and privacy are not limited to the hardware of our computer but they can be found to our energy grid systems, physical access control systems, cloud services, or even when we cross the street in a smart city [63]. Also, due to the heterogeneity of IoT devices, regarding hardware components, software implementation, and communication protocols, their management becomes very difficult. The most important challenge is to create scalable and feasible security mechanisms for controlling access requests in a future with billions of heterogeneous devices connected to the Internet. In the IoE, the evaluation of requests for access to certain piece of information, must provide policies about attributes related to each component which are called security policies. The latter means the definition of a set of rules according to which access request is evaluated [125].

The process of mediating access requests to resources of a system is called access control and is determining whether the request should be granted or denied. The decision is enforced by a mechanism implementing regulations established by a security policy. Different access control policies can be applied, corresponding to different criteria for defining what should be allowed [125].

But, modern interconnected System of Systems (SoS), such as IoE systems, require scalable and efficient security mechanisms, for controlling a very large number of access requests in a future with billions of heterogeneous devices connected to the Internet. The evaluation of requests for access to certain pieces of information and services commonly relies on dedicated policies [93], which incorporate object, subject, and environmental attributes. Such policies are based on predefined rules, while access control is *a process by which use of system resources is regulated according to a security policy and is permitted only by authorized entities (users, programs, processes, or other systems) according to that policy* [132]. A multitude of access control policies can be defined, corresponding to distinct criteria for what should be allowed and what not [125].

There are various access control models in the literature [105]. Whereas most of them provide specific types of access control, the Attribute-Based Access Control (ABAC) models appear as the most generic because they consider various types of different attributes. A withdraw of the most access control models is that the access request is checked only once initially, which indicates the challenge that there are no mechanisms that check the values of attributes during a session so at to reevaluate the policies and eventually the right of continuing access granting.

This type of continuous control is a feature that Usage Control (UCON) model [76] can provide. UCON enhances ABAC models [152] in two novel aspects [106]: continuity of control, and mutability of attributes. Continuity of control is the evaluation of access decision happens not only at request time, but also when the requester executes access rights on the resource [75]. Mutability of attributes means that if there occur changes in attribute values while a session is in progress and the security policy is not anymore satisfied, UCON can revoke immediately the access and terminate the usage of the resources [166].

Furthermore, ABAC models evaluate security policies in eXtensible Access Control Markup Language (XACML) [152]. In addition, UCON uses a XACML extension called XACML with Continuous Usage Control Features (U-XACML) [26]. XACML is an Extensible Markup Language (XML)-based language standardized by OASIS[4] with a very high expressiveness potential, hence fitting the flexibility requirements of complex environments, such as IoE ones.

To further increase its expressiveness, XACML pairs standard ABAC constructs such as authorizations and prohibitions with *Obligations*, i.e. mandatory actions that have to be performed in conjunction with the policy enforcement [26]. Obligations can be of two types according to their target. These are obligations focusing on attribute updates and obligations that force subjects to enforce them. Currently, XACML does not provide a standard for the description of the obligation semantics targeting subjects. There is an existing effort of OASIS to standardize obligations, but only in the level of introducing a tag responsible for forcing attribute updates when necessary. While the intention is the one of non imposing constraints on format of the represented information, the direct consequence is that every obligation management engine has to be developed ad-hoc.

Having a standard semantic for representation which is meaningful at least in an application macro-environment, such as IoE, can push developers and policy editors to use common expressions to represent and enforce obligations. In particular, it is possible to express obligations as commands for an inter-operable service or platform used by a multitude of IoT/IoE devices, such as If This Then That (IFTTT)[5]. In real world implementations, obligations are agreed before obtaining the rights and at that time obligation-related authorization rules are checked.

This arises the necessity of a unique way that could achieve more secure data distribution amongst the various IoT devices together with seamless interaction between them when an obligation must be enforced. IFTTT is a free web-based platform used to create applet chains of conditional statements in IoT settings. Each applet chain, also called *Recipe* or *Applet*, is triggered by changes that happen within various web-services and as a result does specific actions on other web-services. Expressing triggers through UCON obligations will allow the device receiving the policy evaluation decision to easily execute the obligations, without hard-coding the actual obligation interpretation, demanded to the specific UCON applet.

Yet, IoE environments carry inherent limitations in terms of both computational and communications capacity. Accordingly, corresponding optimizations must be

[4] https://www.oasis-open.org/committees/xacml/

[5] https://www.ifttt.com/

implemented to the original UCON design, seeking to maintain operational efficiency at run-time, but also further security objectives related to resilience. Such optimizations must be initially integrated architecturally, and further enhanced within the components of the deployed policy based management systems.

Furthermore, managing complex IoT systems is further complicated by the presence of different communication protocols, application standards, architectures and interaction models, which make the management of security in this environment an extremely challenging task [85]. Still, such a task is mandatory, considering both issues for captured and processed data, and the effect of IoT devices actions in the physical world. To manage security and privacy in IoT environments, UCON has been successful used as a control tool able to handle seamless access management and data protection [72].

1.1 Goal and Overview of the Thesis

This section gives and overview of the thesis and its goal. Taking into consideration all the previous, we identified the gap of providing more secure solutions in the level of IoT protocols, whereas in the meanwhile to also consider the enforcement of more complex security policies that the IoE creates. Thus, we do not only need a solution that can work to each protocol separately, but we need a mechanism that can be used simultaneously in more than one protocols and enforce policies accordingly. Moreover, this mechanism must be very light so as to perform efficiently in IoT constraint environments but powerful enough to provide continuous control in the access to the resources.

So, in order to cope with the previous aspects, the goal of this thesis is to define and evaluate a generic policy enforcement architectural model that is based to a extended and adapted version of UCON framework, targeting to provide continuous access control and policy enforcement on the IoE. This model shall target to provide a hierarchical variation of UCON that will continuously enforce policies and control the access to the resources of complex IoE systems. This model shall be able to provide a holistic solution independently from the type, application protocol, policy attribute number or language between all components of an IoE environment. i.e. when attributes refer to the behavior of people, which increases policy complexity, UCON can cope with it seamlessly.

1.1.1 Overview Regarding IoT Protocols

In order to achieve the goal of this thesis we firstly started on understanding the status of the current bibliography and finding the gaps described above. The next step was to identify how UCON is able to work alongside IoT protocols in order to enhance their security capabilities. Considering the architecture of IoT application protocols we tried to figure out the most viable way for UCON to be added on their architecture. This way is by intercepting access requests in the communication level besides the fact that the communication pattern is not the same in every protocol. Our survey, though, revealed that most of the IoT protocols follow two main communication patterns which are either the Publish/Subscribe or the Client/Server one. For this reason we selected to work with two protocols, where each of them uses one of the previous mentioned communication patterns. After surveying wich protocol could be the most appropriate we made the choice to use in the case of the Publish/Subscribe pattern the MQTT protocol, and in the case of the Client/Server pattern the CoAP protocol. Our goal in this step was to add UCON in the architecture and workflow of the protocols so as to prove that UCON can acomplish this task.

Firstly, we added UCON in MQTT where we aim at enforcing dynamically fine grained policies, which do not only consider the identity of the Subscriber as a parameter for granting access to data, but also dynamic attributes such as Subscriber reputation, data reliability, or environmental conditions of a specific application. The proposed framework is designed to be general, easy to integrate in the Broker component, remaining oblivious to both Publishers and Subscribers. The addition of UCON in fact, does not modify the MQTT protocol, enforcing the policies independently from the implementation of Publisher and Subscriber, which allows the proposed solution to be compatible with any Off-the-Shelf MQTT Publisher/-Subscriber application. Furthermore, we demonstrate the viability of the approach by presenting a real implementation of the framework on both general purpose and performance constrained devices, discussing also the performance measured on two Rasperry Pi 3 model b[6], used respectively as Broker and Subscriber.

After understanding and exploring the results of this work, our next step was to try and see how UCON can perform the same task in CoAP protocol. But, the result showed us that considering IoT it is better to use distributed UCON [109]. The proposed solution is based on a fully distributed Peer-to-Peer (P2P) UCON system

[6]https://www.raspberrypi.org/products/raspberry-pi-3-model-b/

[85]. It has been specifically designed to be implemented in IoT architectures and it is based on a group of smart-nodes which have their own logical architecture that can match UCON.

By working on a distributed environment, each node has its own access directly to *local* attributes that are accessed only by this node. Conversely, there are also the *remote* attributes the values of which can be remotely acquired by other nodes. The policies in the distributed UCON framework consider both attribute types. By using this version of UCON framework, we have the ability to create subsystems that have a standalone Usage Control System (UCS) with its own Policy Enforcement Point (PEP)s and *local* attributes. At the same time, all of them are connected together in a distributed system which allows access to *remote* attributes [27].

This work gave the possibility of creating for the first time, a SoS, consisted of different IoT subsystems, so that the latter can exchange information about attributes, and achieve the security enhancement of both CoAP and MQTT protocols by adding strong authorization capabilities provided by the distributed UCON. With this solution, we aim to provide dynamic policy enforcement towards more secure data distribution on both protocols at the same time.

1.1.2 Overview Regarding Interoperability of IoT Devices and Services

Moreover, we enhance the interoperability between the protocols, by sharing attribute values via distributed UCON. This work presents the implementation of UCON on top of CoAP and MQTT. We highlight that the presence of UCON does not impact how the protocols work, which means any CoAP and/or MQTT application can support it. To further our approach, we measure the performance of a real implementation.

In addition to the previous, we found that the interoperability in IoE is not only a protocol matter, but also between services. This is a matter of how UCON can not only stop the access when necessary, but also to start other services as a result to the previous revoke. For this part, we achieved this goal by presenting the application of IFTTT triggers to express and enforce UCON obligations. This work is describes a novel architecture where the standard UCON framework is combined with the IFTTT platform services. Our architecture, is providing a standardized format of obligations so that they can trigger valid IFTTT *Applets*. The proposed architecture is designed to be independent from the specific device implementation, specific application and communication protocols.

Thus, our work does not alter the XACML model and workflow, whereas we enhance its capabilities by proposing a standardized way of expressing obligations, reusing pre-existing components, being thus non intrusive. In this work we discuss the full operative workflow, detailing both UCON and IFTTT operation parts, proposing two relevant use cases and a set of performance experiments to demonstrate the viability. Moreover, our work does not interfere with the execution of the IFTTT *Applets* and does not require any modification in the model of UCON. Our contribution on this part is to define a standardized way of expressing obligations in XACML where the recipient of the obligation can create a valid IFTTT expression out of it in order to trigger an existing *Applet*. We present the viability of the proposed approach by presenting the details of the architecture and implementation of our framework, relevant use-cases and also a set of experiments to measure the timings that an obligation needs to be enforced over an example that triggers an existing IFTTT *Applet* created for this purpose.

1.1.3 Overview Regarding Enhancements in Usage Control

Based on the previous findings we have created a hierarchical architectural model based on the distributed version of UCON on which we can control access requests inside a system of systems that is consisted of subsystems in the area of IoE. On this system, we have the ability to exchange attribute information between the subsystems and mediate action requests considering policies that either affect the local subsystem or the global system. This hierarchical model, will can control complex IoE systems, even though they work with different communication protocols and different types of attributes, that although inside the subsystems a specific access request may be permitted, but a total violation of the resource usage may lead to a revoke of an upper hierarchically entity.

On the top of the previous, we identified and tried to mitigate the limitations of the original and distributed UCON [45]. The current UCON architecture, requires the complete re-evaluation of access permissions per user-asset-session triplet, both at the initiation and at runtime. This, has been experimentally proven to require many computational resources, especially as the users, assets, sessions and policy attributes increase [46]. Accordingly, we describe the developed architectural optimizations to the original UCON, seeking to positively affect run time efficiency, scalability, and resilience against active attacks. In order to achieve that, a service group functionality is introduced to the existing model alongside with a dynamic role allocation subsystem, both based on risk aggregation. Thus, the right of access will be granted to a user, based on his allocated role for each group of services and not for one service at a time. The integrated optimizations improve the performance of the model, while increasing its resilience by allowing the mitigation of specific types of active attacks that are based on request flooding. Architectures of this nature can be described in three abstraction levels, maintaining consistency and completeness. These levels are the (i) architectural model and components, (ii) protocol and interface, and (iii) implementation. In this thesis, we present and discuss the suggested architecture in all three levels, highlighting the integrated optimizations to the original UCON and the corresponding affects.

Moreover, security policy-based management is founded on the establishment of explicit goals and courses of action, in order to govern decisions in respect to the use of assets. These are instantiated as a set of rules in distinct levels of abstraction, which within their formulation bind the actions required to meet the policy goals to a set of preconditions. Finally, these preconditions are evaluated, by the use of a policy administration and enforcement subsystem, in order to determine if the action is permissible. Monitoring the values and evaluating such preconditions in order to enforce fine-grained security policy based management, requires the consumption of resources, such as network bandwidth, computational capacity and storage, which in constrained distributed dynamic systems commonly exceeds the capacity of both the edge nodes and the proximity network.

The last part of this thesis presents a mechanism for the specification and enforcement of security policies within such environments, where the computational burden is transferred to high-tier nodes, while low-tier nodes apply risk-aware policy enforcement based on a compromise solution. The proposed mechanism for calculating such a solution is founded on a compensatory multi-criteria decision making function, based on the calculation of the Euclidean distance between the run-time values of the preconditions and their ideal values, as those are specified within the established policy rules.

Summarizing, the overall contribution of this thesis shall be to provide for the first time continuous access control in IoE ecosystems in the level of communication protocols. This continuous access control is based on UCON ABAC model and also several enhancements to it are proposed so as to deal with the complexity of IoE. The biggest advantage is that apart from the merging of UCON and protocols to communicate with each other there is no other modification to protocol structure and components. Finally, another contribution is to provide overall policy enforcement in a IoE SoS independently from the protocol the various IoT devices use.

1.1.4 Structure of the Thesis

The structure of this thesis is the following: In Chapter 2 we provide background information about IoT protocols, detailing also UCON and its distributed version and we discuss the limitations that motivated this thesis. In Chapter 3 we analyze the way how we can use UCON together with IoT protocols, and how we can build on the interoperability proving that we do not need to change either the protocol or the UCON architectures. In Chapter 4 we present the hierarchical model that we created and 5 our proposed optimizations considering UCON limitations and optimized policy management in IoT. Finally, Chapter 6 provides the conclusions that stem out of this work.

1.2 Research Questions of the Thesis

After surveying the literature and during the conduction of this thesis material the following research questions arised.

1. Since IoE is a continuous changing and dynamic ecosystem, are the communication protocols that distribute information inside an IoE ecosystem capable of providing monitoring and control on the access continuously?

2. Is UCON capable of contributing towards the addressing of the previous question?

3. Can UCON be used as it is in the workflow of the communication protocols, or are there any modifications / alterations of it necessary to achieve the goals of this thesis?

4. Do the communication protocols need any modifications to be used together with UCON?

5. Can we provide a solution that is protocol independent and can work inside an heterogeneous ecosystem?

6. How can we deal with the increased complexity in IoE ecosystems?

1.2.1 Answers to the Research Questions

Regarding the first research question, the methodology that was followed was to survey the models and the architecture of the most used communication protocols that are used in this area. Following that, the next step was to survey and find their security limitations over access control techniques and continuous access monitoring so as to justify the need of UCON. These survey is described in Chapter 2 and also in the first and second publications of the publication list in Section 1.3.

Furthermore, trying to understand how UCON can be architecturally combined with these protocols we checked firstly to find any similarities in the architecture of any of the protocols surveyed that lead to the first result of implementing UCON inside MQTT protocol. The next step was to understand whether we can use UCON in other protocols so as to prove that it is in fact independent of the protocol architecture. Following the previous methodology we achieved to do the same in the CoAP protocol that does not belong in the Publish/Subscribe group like MQTT, but in the Client/Server one. These two are the main groups of the communication protocols and, by achieving to inesrt UCON in one protocol of each group, we proved that UCON can work seamlessly with the major communication protocols used in IoT. This work is presented in the penultimate publication of the list in Section 1.3. By doing that we answer to the second question and this was proved by conducting experiments in both a simulated and a real IoT environment. In addition to the previous, and regarding the third research question, we realized that UCON can work simultaneously in a variety of protocols without alterations and have the ability to provide continuous access monitoring and policy enforcing.

One of the major advantages of our approach is the fact there was not any modification needed in UCON architecture and model which answers to the third question above. Moreover, another advantage is that apart from creating the connection in the central entity of MQTT protocol (Broker) and CoAP protocol (Server) to be able to call UCON and vice versa there was not any modification in the protocol architecture either and the components used were not explicitly created but off-the-self ones. Especially the satellite components that either publish or subscribe for information have no modifications at all. The architecture and the result of this effort are presented in 3 and the outcome has been published in the publications one, two and six of the list presented in Section 1.3.

For question number five, we show in Chapter 3 that by taking advantage of a more recent architecture of UCON that is distributed and provides attribute sharing amongst the several instances of the UCON frameworks. This version gives the ability to have a global system of several UCON instances, that attribute changes in one instance can affect the policy decisions to the other. Also we can build a complex, hierarchical structure, as shown in 4, where each instance can operate individually, but with the correct changes in shared attributes, there can be policy enforcement from higher in hierarchy components to lower ones. The following appear to Chapters 3 and 4 and also in the sixth publication of the list in Section 1.3.

But, the complexity of IoE ecosystems leads to numerous attributes and policies that are not only complex, but difficult to deal with while maintaining efficiency, scalability and time restrictions which is the last research question. In order to answer to this question, we tried to address this issue by surveying and grouping the several systems of a certain bigger SoS where the same level of access is needed and the same attributes and values must appear to provide access to resources. Thus, we can reduce time evaluation by providing access to the whole group of services which acts as a reply to the last research question. The research methodology that we followed towards this aspect is shown in the first part of Chapter 5 and its outcome is presented in publications three and four of the list in the following section. Moreover, in order to reduce the attribute number of the policies and, thus, the evaluation time of them, we have created a mechanism to pre-compute the attribute values and extract a single value that acts as the role of the subject that asks for access to the resources. The evaluation is done for this role, which provides simpler policies and less time evaluation. Finally, the methodology for this appears in the last part of Chapter 5 and its outcome in the last publication of Section 1.3.

1.3 Publications

This section presents the outcome of this thesis.

- La Marra A., Martinelli F., Mori P., Rizos A., Saracino A. (2018) Introducing Usage Control in MQTT. In: Katsikas S. et al. (eds) Computer Security. SECPRE 2017, CyberICPS 2017. Lecture Notes in Computer Science, vol 10683. Springer, Cham

- La Marra A., Martinelli F., Mori P., Rizos A., Saracino A. (2017) Improving MQTT by Inclusion of Usage Control. In: Wang G., Atiquzzaman M., Yan Z., Choo KK. (eds) Security, Privacy, and Anonymity in Computation, Communication, and Storage. SpaCCS 2017. Lecture Notes in Computer Science, vol 10656. Springer, Cham

- V. Gkioulos, A. Rizos, C. Michailidou, F. Martinelli and P. Mori, "Enhancing Usage Control for Performance: A Proposal for Systems of Systems" 2018 International Conference on High Performance Computing & Simulation (HPCS), Orleans, 2018, pp. 1061-1062

- Gkioulos V., Rizos A., Michailidou C., Mori P., Saracino A. (2019) Enhancing Usage Control for Performance: An Architecture for Systems of Systems. In: Katsikas S. et al. (eds) Computer Security. SECPRE 2018, CyberICPS 2018. Lecture Notes in Computer Science, vol 11387. Springer, Cham

- La Marra A., Martinelli F., Mori P., Rizos A., Saracino A. (2019) Using IFTTT to Express and Enforce UCON Obligations[7]. In: Heng SH., Lopez J. (eds) Information Security Practice and Experience. ISPEC 2019. Lecture Notes in Computer Science, vol 11879. Springer, Cham

- Rizos A., Bastos D., Saracino A., Martinelli F. (2019) Distributed UCON in CoAP and MQTT Protocols. In: Katsikas S. et al. Computer Security. SECPRE 2019, CyberICPS 2019. Lecture Notes in Computer Science, Springer (To Appear)

- Michailidou C., Gkioulos V., Shalaginov A., Rizos A., Saracino A.(2019) RESPOnSE - Risk Aware Security Policy Based Management In Constrained Distributed Dynamic Systems. Under review in Pervasive and Mobile Computing, an Elsevier Journal

[7] Awarded with Best Paper Award of the ISPEC 2019 Conference

Chapter 2

Background

This chapter provides with the background information about IoE, IoT, access control, UCON and the related work considering all the areas that this thesis deals with.

2.1 The Internet of Everything

As discussed above, IoE is a concept that extends the IoT emphasis on M2M communications to describe a more complex system that also encompasses data, people and processes [92].

2.1.1 The Internet of Things

IoT consists of a network of heterogeneous interconnected objects that interact with each other in order to collect and share information. Following the vast evolution of electronics, sensors and actuators, researchers and engineers aim to exploit the potentials of systems in order to introduce new types of services targeting to interconnect every single device that people interact with [40]. This process leads us to the era of urban computing, because, in the IoE, many heterogeneous devices, such as the power grid, transportation means, traffic lights etc., can be interconnected. This inter-networking of Cyber-physical devices and other items embedded with electronics, software, and sensors allows them to collect and exchange data [40]. The connectivity of computing devices, systems and services is Internet Protocol (IP) based. There are many devices already deployed, featuring sensors and actuators, that are applied in a variety of domains such as residential automation

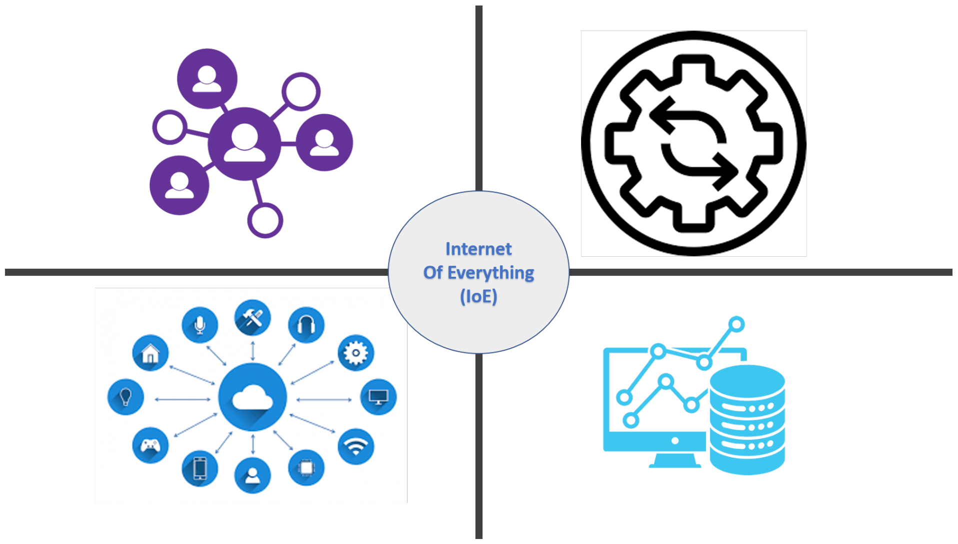


Figure 2.1: The Internet of Everything (IoE) Example.

industrial systems, military and healthcare [20]. From the perspective of implementers, there is a need for rapid development whilst issues about the limitations of resources have to be tackled [3]. One example of urban IoT architecture is the Smart City services, which are based on a centralized architecture where all the various peripheral devices installed all over the urban areas generate different types of data that are delivered to a control center via various communication protocols where the data is stored and processed [163].

2.1.2 From the Internet of Things to the Internet of Everything

The recent advancement in connection technologies, smart devices, sensor technology and networks combined with a strong interaction with people and social environments have a huge impact on everything, from city planning to military, health, and transportation [160]. However, IoE provides links among not only things, but also data, people, and (business) processes. That is why the IoT environment is transforming into the IoE one. IoE has become a very common phrase in order to give a description about adding Internet connectivity and intelligence to almost every device so as to be able to provide to them special functionalities [160].

Figure 2.1 shows that the IoE is the interconnection of *Things*, *People*, *Data* and *Processes* among them. There is a need to provide a common ground for integrating information coming from heterogeneous sources. Such a shared ecosystem would allow for the interaction among data, sensor inputs, and heterogeneous systems. We can have countless IoE applications in our everyday life around the globe [61]. Presently, it becomes a lot easier to network smart objects with each other, due to advancement in sensors and wireless technology. Some of the most commonly deployed applications include home and industrial automation, e-health system, environmental and infrastructure monitoring and management and smart transportation system:

- Home automation and security that may include smart grid, appliance control, light management etc.
- Industrial automation and management of manufacturing equipment and assets.
- E-Health system spanning from heart rate monitoring to remote surgeries.
- Environmental monitoring (e.g. air, water quality, soil, wildlife).
- Infrastructure management and monitoring of urban and rural assets.
- Smart transportation system (e.g. smart parking, traffic control, vehicle to vehicle communication etc.).
- Industrial projects in food industry, agriculture, surveillance etc.

A nice example of IoE that includes various type of IoT services, their usage of people and data processing of them is "IFTTT"[1]. IFTTT is a free web-based services platform where it is possible to create chains of simple conditional statements, called applets. This is an example of an area of combining several different services together expressed in a high level language. In IFTTT, services are combined via a simple if statement that on the one hand we have the cause which triggers a specific action to happen without the need of creating specific applications for this scope. The applet is triggered by changes that occur within other web services such as Gmail, Facebook, Instagram, or Pinterest. For example, an applet may send an e-mail message if the user tweets using a hashtag, or copy a photo on Facebook to a user's archive if someone tags a user in a photo. IFTTT employs the following concepts for every applet:

[1] https://ifttt.com/

- `Services`: The basic building blocks. They mainly describe a series of data from a web service such as YouTube.
- `Triggers`: The "this" part of an applet and trigger the action.
- `Actions`: The "that" part meaning the results from the trigger.
- `Applets`: The predicates made from Triggers and Actions. For example, if you like a picture on Instagram (trigger), an IFTTT app can send the photo to your Dropbox account (action).
- `Ingredients`: Basic data available from a Trigger. For example, in an email Trigger they are the subject, body, attachment etc.

All triggered applets by the various changes cause policy re-evaluation before granting any actions. Exchanged data might be very sensitive and they have to be continuously protected so there will be no violation when they are moving among different and heterogeneous services of IFTTT. In this area, UCON can be very useful because it can control information sharing by monitoring continuously attributes over different IFTTT services and enforce appropriate policies.

2.2 Application Layer Communication Protocols

This paragraph surveys the most known communication protocols for IoT. The most known are CoAP [20], MQTT [82], Extensible Messaging and Presence Protocol (XMPP) [3], HTTP [22], Advanced Message Queuing Protocol (AMQP) [83]. Their main characteristics and differences are shown in Table 2.1.

In the area of IoT, the devices are generally constrained in regards to computational power and network availability, so application protocols are usually designed to be very light. Popular protocols for IoT include CoAP, MQTT, XMPP, AMQP, Web Application Messaging Protocol (WAMP) and also HTTP for web compatibility [73]. Considering most of these protocols have similar purposes and goals, the main differences between them are the communication patterns and the security features they support. The CoAP and HTTP protocols only use the Request/Response communication pattern, while MQTT only uses the Publish/Subscribe. AMQP and XMPP are Message-Oriented Middleware (MOM) protocols, which are more complex and support different communication patterns. WAMP is based on WebSockets, which allows a great deal of flexibility regarding communication patterns.

The Request/Response communication pattern is one of the most basic communication patterns. In a Client/Server architecture, it allows *Clients* to perform real-time requests on resources held on a *Server*. For each request from a *Client* to the *Server*, a response follows from the *Server* to the Client. The Publish/Subscribe pattern is comprised of three entities: a Broker, one or Publishers and one or more Subscribers. A Publisher composes messages which are pushed to Subscribers. Subscribers have the option to subscribe to specific messages which are organized in Topics. The Broker manages the messaging between publishers and subscribers, therefore neither Subscribers or Publishers need knowledge of each other. Asynchronous Messaging is usually associated with MOM protocols, which comprise a category of inter-application communication software. In asynchronous systems, message queues provide temporary storage when the destination program is busy or not connected. In addition, most asynchronous MOM systems provide persistent storage to back up the message queue. Thus, the main difference is that in MOM protocols the messages are stored and delivered, while in for example in MQTT they are simply routed.

When deciding which communication pattern fits better for a particular IoT system, it is important to consider the following questions:

- Does information need to be updated in real-time for continuous values (non-discrete values)?
- Does information need to be updated on demand?
- Is published information always used?
- Does the system require strong authentication?
- Does the system require data confidentiality?

The answers to these questions help deciding which communication pattern and protocol to choose for use in that particular IoT system.

As for the protocol characteristics, the protocols that provide a form of Quality of Service (QoS) are CoAP, MQTT and AMQP. All protocols use the Transmission Control Protocol (TCP) transport layer and the TLS/SSL security layer apart from CoAP that uses the User Datagram Protocol (UDP) and DTLS respectively. Data Distribution Service (DDS) has the ability to to use both TCP and UDP for

Table 2.1: Comparison of IoT Protocols

Protocol → / Features ↓	**MQTT**	**CoAP**	**XMPP**	**AMQP**	**WAMP**	**HTTP**
Open Standard	Yes (OASIS, ISO)	Yes (IETF)	Yes (IETF)	Yes (OASIS, ISO)	Yes	Yes (IETF)
Architecture	Centralized	Centralized	Decentralized	Centralized	Decentralized	Centralized
Transport Layer	TCP	UDP	TCP	TCP	TCP	TCP
Publish / Subscribe	Yes	No	Yes	Yes	Yes	No
Request / Response	No	Yes	Yes	Yes	Yes	Yes
Asynchronous Messaging	Yes	No	Yes	Yes	Yes	No
RPC	No	Yes	Yes	Yes	Yes	Yes
WebSocket	Yes	No	No	Yes	Yes	Yes
TLS	Yes	No	Yes	Yes	Yes	Yes
DTLS	No	Yes	No	No	No	No
SASL	No	No	Yes	Yes	No	No

transport and both TLS and DTLS for security layer respectively [65]. In terms of header size MQTT provides the smaller with 2 bytes whereas CoAP has 4 bytes and AMQP 8 bytes [3]. HTTP performs better in non-constrained environments due to high overload. According to [40], among the two most widely used protocols (MQTT and CoAP), CoAP provides less CPU load and has lower delays for high packet loss in comparison to MQTT. Also, CoAP can be applied in very constrained environments, which is useful in the area of the IoT.

As a synopsis to the basis, this comparison gives the details about the existence of QoS. During literature review, we identified that MQTT is more general purpose and CoAP is the one that is targeting the most constrained environments. Finally, it refers to the communication pattern which in the case of all the main protocols is either Publish/Subscribe (e.g. MQTT) or Client/Server (e.g. CoAP). The following paragraphs survey briefly the protocols, motivating the choice to focus on MQTT and CoAP. From now and on and for the shake of simplicity, they will described briefly as IoT protocols.

2.2.1 Constrained Application Protocol (CoAP)

CoAP is an application layer protocol designed for the IoT and for M2M applications. The protocol was introduced as a standard by the IETF in 2014 [131] and its key features are simplicity for constrained environments, very low overhead, and easy interoperability with HTTP.

Communication in CoAP happens over a Client/Server architecture and follows

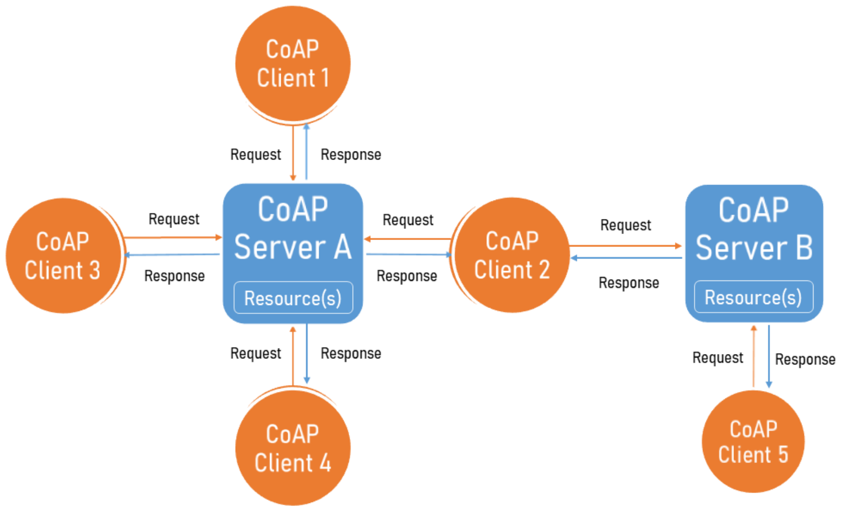


Figure 2.2: Constrained Application Protocol (CoAP) Architecture.

the Request/Response pattern. A *Client* sends a request to the *Server* to perform an action on a resource. Each request is composed by a method code (*Get, Post, Put, Delete*) and a Uniform Resource Identifier (URI) which identifies the targeted resource on a *Server*. After receiving the request, the *Server* processes the request and sends a response to the *Client* containing the information acquired by other *Clients* and in accordance with the requested action. A token is used in order to match responses to requests. The response contains a code similar to HTTP ones and, if requested, the respective resource representation.

CoAP offers the ability for a *Client* to follow the updates of a certain resource on a scheduled basis, without sending multiple requests, via a feature called *Observe* [58]. When the *Observe* flag is set on inside a CoAP *Get* request, the *Server* continues to reply after the initial response, streaming resource state changes to the *Client* as long as they occur. After being set, an observation can be canceled at any point by the *Client* if it wants to stop receiving resource updates. This operation is triggered by sending a request to the *Server* with the *Observe* flag off or by replying to a resource update with a reset message to the *Server*. Figure 2.2 presents the CoAP architecture, showing a *Server* interacting with multiple *Clients* and, on the right side, a *Client* interacting with other *Server*.

Conversely to HTTP, CoAP uses the UDP instead of the TCP for message exchanging. In essence, this means that reliability for message exchange is not guaranteed as UDP doesn't support mechanisms for reliable communications, focusing instead on fast and simple message exchanging. As a result, CoAP implements two request types: Confirmable (CON) and Non-Confirmable (NON). On the one hand, CON requests achieve reliable communications by expecting an acknowledgement (Ack) message from the *Server* in response to each request. On the other hand, NON requests are "fire and forget" messages that do not expect any confirmation that the request was indeed received by the *Server*. As a result to the previous, CoAP provides two QoS levels: *at least once* (using CON requests), and *At most once* (using NON requests). Because CoAP uses UDP it supports multicast requests, where one *Client* can send the same request to multiple *Servers* at the same time. However, a CoAP *Server* always replies in unicast to a multicast request. The CoAP protocol by itself does not provide any security features (e.g. authentication), so security is not designed in the context of the application layer but instead, it is optionally supported at the transport layer [52]. CoAP supports the DTLS protocol [113]. DTLS provides three modes for secure message exchanging: *PreSharedKey, RawPublicKey and Certificate* [42]. These modes allow for strong authentication and data encryption and integrity in transit. However, they do not provide refined authorization capabilities. In the *PreSharedKey* mode there is a list of pre-shared keys and each key includes a list of which endpoints it can be used to communicate with. In the *RawPublicKey* mode each endpoint holds an asymmetric key pair without a certificate (a raw public key) that defines its identity and includes a list of other endpoints it can communicate with. Finally, in *Certificate* mode each endpoint has an asymmetric key pair with an X.509 certificate that binds it to its subject and is signed by some common trust root. A CoAP endpoint is either the source or the destination of a CoAP message, and is identified by an Internet Protocol (IP) address and an UDP port number. However, with DTLS enabled, the endpoint is identified as defined by the security mode as explained above. Default ports for CoAP are 5683 (with no security) and 5684 (with DTLS enabled). This work focuses on the CoAP protocol since it is the most constrained one and widely used in the area of IoT. Existing security features of CoAP protocol rely only on the use of DTLS, which does not deal with what happens after the information is shared to the *Client* and does not provide a continuous checking mechanism on the access to the information shared between *Client(s)* and *Server(s)*. Our work addresses this problem by providing continuous verification of access authorization to resources on *Server(s)*.

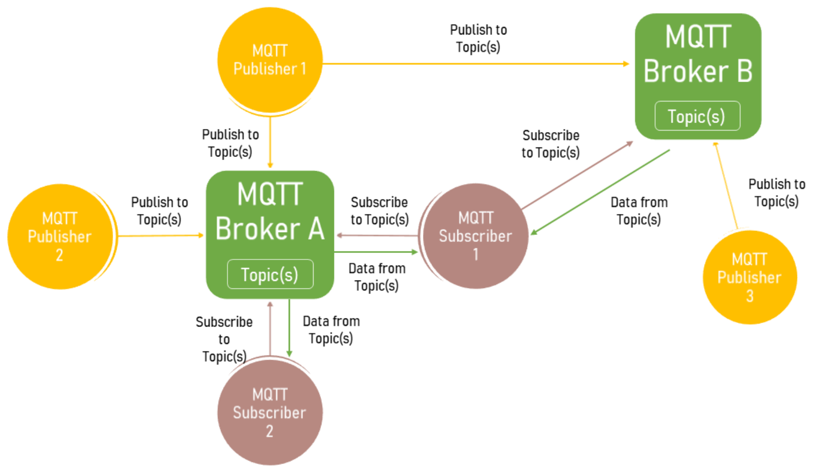


Figure 2.3: Message Queuing Telemetry Transport Protocol (MQTT) Architecture.

2.2.2 Message Queuing Telemetry Transport Protocol (MQTT)

MQTT is a Publish/Subscribe protocol designed for constrained devices used in telemetry applications. MQTT is designed to be very simple on the *Client* side either this is the *Subscriber* or the *Publisher*. Hence, all of the system complexities reside on the *Broker* which performs all the necessary actions for the MQTT functionality. MQTT is independent from the routing or networking specific algorithms and techniques. However, it assumes that the underlying network provides a point-to-point, session-oriented, auto-segmenting data transport service with in-order delivery (such as TCP/IP) and employs this service for the exchange of messages.

MQTT is also a topic-based Publish/Subscribe protocol that uses character strings to create and control support of hierarchical topics. It is possible for a *Publisher* to publish to multiple topics and for a *Subscriber* to subscribe to multiple topics at the same time. In Figure 2.3, we can see the topology of the protocol. It shows *Publishers* that publish data to topic(s) in the *Broker*. *Subscribers* authenticate with the *Broker* in order to subscribe to topics. The *Broker* is responsible to send value updates for each topic that every *Subscriber* is subscribed to. The *Publishers* and the *Subscribers* can be very constrained devices, especially in the case of *Publishers* which can be simple sensors. Conversely, the *Broker* must provide enough computational power to be able to handle the amount of data being distributed. Depending on how critical is the application, MQTT provides three QoS levels for message delivery [19]. QoS level 0 only offers a best-effort delivery service, in which messages are delivered either once or not at all to their destination. No re-transmission or acknowledgment is defined. QoS level 1 re-transmits messages until they are acknowledged by the receivers. Hence, QoS level 1 messages may arrive multiple times at the destination because of the re-transmissions, still multiple copies are not natively handled. QoS level 2 ensures not only the reception of the messages, but also that they are delivered only once.

We primarily focused on the MQTT protocol since it is the most generic among the IoT protocols described, and libraries are available for all major IoT development platforms, like Arduino, for several programming languages (C, Java, PHP, Python, Ruby, Javascript) and for the two major mobile platforms (iOS and Android) [24]. The authentication to the Broker can be done by providing the following credentials [82]: Topic to be Subscribed on, Username and Password. The most known effort to add more security features in MQTT is SMQTT [24], but no solution is given to the policies that are followed by the information after it is delivered to the Subscribers.

2.2.3 Advanced Message Queuing Protocol (AMQP)

AMQP is an open messaging protocol that can be used to build cross-platform, hybrid applications. AMQP version 1.0 is an OASIS standard [97] and an International Organization for Standardization (ISO) standard [139]. AMQP is a MOM protocol and its characteristics include being a wire-protocol, interoperable, reliable, and efficiently supporting multiple messaging applications and communication patterns. This protocol operates over TCP, supports TLS and has integration with the Simple Authentication and Security Layer (SASL) protocol for authentication. Similarly to MQTT, AMQP uses a centralized Client/Server architecture and, while supporting multiple communication patterns, it prefers the Publish/-Subscribe pattern. Following the pattern nomenclature the central entity is called Broker, however in AMQP the Publishers and Subscribers are referred to as Producers and Consumers respectively. The AMQP architecture is shown in Figure 2.4.

In AMQP there are two important concepts: exchanges and queues. When a Producer sends a message to the Broker, the message is published to an Exchange, which is comparable to a mailbox. The Exchange is responsible for routing the messages to different Queues with the help of bindings and routing keys. Queues are entities that store and forward messages. Subsequently, the Broker either deliv-

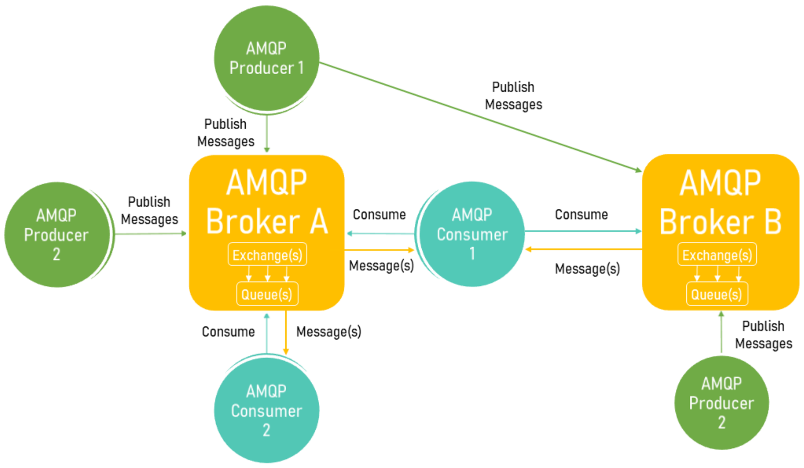


Figure 2.4: Advanced Message Queuing Protocol (AMQP) Architecture.

ers the message to Consumers subscribed to the Queues, or consumers fetch/pull messages from the Queues on demand.

Furthermore, the complexity of AMQP is greater in comparison to other protocols like MQTT, since it is comprised of several layers. The lowest layer defines a symmetric, asynchronous protocol for the transfer of messages between two processes over a network. Above this, the messaging layer defines an abstract message format, with standard encoding. Then a type system, and finally a set of standardized but extensible "messaging capabilities." AMQP provides QoS message-delivery guarantees such as at-most-once, at-least-once and exactly-once.

2.2.4 Hypertext Transfer Protocol (HTTP)

HTTP is the foundation of data communication for the World Wide Web [28]. Hypertext is structured text that uses logical links (hyperlinks) between nodes containing text. HTTP is the protocol to exchange or transfer hypertext. The standards development of HTTP was coordinated by the IETF and the World Wide Web Consortium (W3C)W, culminating in the publication of a series of Requests for Comments (RFCs). The first definition of HTTP/1.1, the version of HTTP in common use, occurred in RFC 2068 in 1997, although this was obsoleted by RFC 2616 in 1999. HTTP functions as a Request-Response protocol in the Client/Server computing model. A web browser, for example, may be the *Client* and an application running on a computer hosting a web site may be the *Server*. The *Client* submits an

HTTP request message to the *Server*. The *Server*, which provides resources such as Hypertext Markup Language (HTML) files and other content, or performs other functions on behalf of the client, returns a response message to the Client. The response contains completion status information about the request and may also contain requested content in its message body. A web browser is an example of a user agent. HTTP is an application layer protocol designed within the framework of the Internet Protocol Suite. Its definition presumes an underlying and reliable transport layer protocol, and TCP is commonly used. However, HTTP can use unreliable protocols such as the UDP. HTTP resources are identified and located on the network by Uniform Resource Locators (URL)s, using the URI schemes "http" and "https". URIs and hyperlinks in HTML documents form inter-linked hypertext documents.

2.2.5 Extensible Messaging and Presence Protocol (XMPP)

XMPP is an open standard introduced in 2011 and developed by the IETF [122]. It is based on the XML and enables the near-real-time exchange of structured data between network entities. XMPP is a MOM protocol and the successor of Jabber, a protocol that was used for on-line instant messaging (IM). This protocol uses a decentralized client-server model which supports asynchronous *Client* to *Server* and *Server* to *Server* messaging. XMPP entities are referred by globally unique addresses, often called Jabber IDs or JIDs. The XMPP architecture is shown Figure 2.5.

The two most important concepts of XMPP are XML stanzas and XML streams. XML stanzas are small pieces of structured data and there are three types of stanzas: message, presence, and iq (short for "info/query"). These three types are part of a first-level element whose qualifying namespace is "jabber:client" or "jabber:server", and stanzas usually have one or more child elements (with accompanying attributes, elements, and XML character data). XML streams are open TCP connections between *Clients* and *Servers* or between *Servers* which allow exchange of XML stanzas over a network. A stream only supports unidirectional communication from the initiating entity to the receiving entity (e.g. *Client* to *Server*). Thus, in order to achieve two way exchange of stanzas two streams must be negotiated.

Stream negotiation is a multi-stage process and stream features can be either mandatory-to negotiate or voluntary-to-negotiate. Authenticating a stream could be achieved using the SASL protocol, and for *Client* to *Server* connections

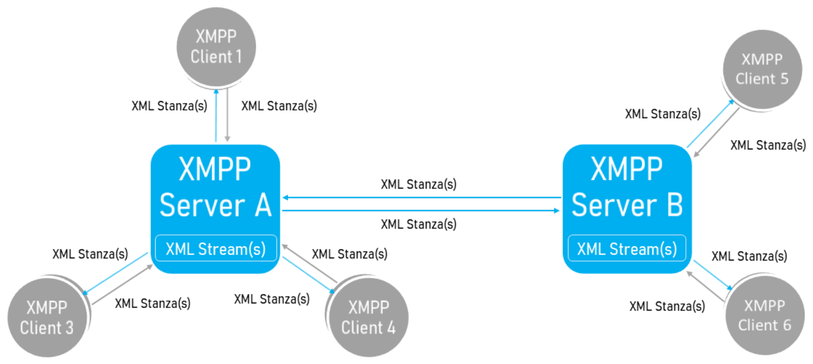


Figure 2.5: Extensible Messaging and Presence Protocol (XMPP) Architecture.

its use is mandatory. XMPP uses a specific XML namespace profile of SASL ("urn:ietf:params:xml:ns:xmpp-sasl"). Transport security using TLS can also be defined by the receiving entity as mandatory-to-negotiate which means that negotiating a stream can ultimately involve four layers: first TCP, then TLS, then SASL and finally XMPP.

The XMPP natively uses the Asynchronous Messaging communication pattern, where entities send messages whenever they decide to and in real-time. However, it also supports Request/Response pattern (using the iq stanza) and work has been developed to allow for support of the Publish/Subscribe [39] pattern.

2.2.6 Other Protocols

In this section, we present a couple of protocols that, although they belong to the application layer communication protocol group, they either not intend to IoT or they are more than simple protocols for communication.

2.2.6.1 Websocket Protocol

WAMP is a protocol created by Crossbar.io developers in 2012 [29]. It is a subprotocol of WebSocket, offers routed Remote Procedure Call (RPC) and supports the Publish/Subscribe pattern. The WebSocket protocol is a browser-based protocol standardized by the IETF in 2011 [90]. The Websocket protocol [37] was developed as part of the HTML5 initiative to facilitate communications channels over TCP [65]. Websocket is neither a Request/Response nor a Publish/Subscribe protocol. In Websocket, a *Client* initializes a handshake with a *Server* to establish a session. The handshake process is intended to be compatible with HTTP-based server-side software so that a single port can be used by both HTTP and Websocket *Clients* [37]. However, what comes after the handshake does not comply with the HTTP rules. In fact, during a session, the HTTP headers are removed and *Clients* and *Servers* can exchange messages in an asynchronous full-duplex connection. The session can be terminated when it is no longer needed from either the *Server* or the *Client* side. Websocket was created to reduce the Internet communication overhead while providing real-time full-duplex communications.

2.2.6.2 Web Application Messaging Protocol

There is also a Websocket sub-protocol called Websocket Application Messaging Protocol (WAMP) that provides Publish/Subscribe messaging systems [65]. Websocket runs over the reliable TCP and implements no reliability mechanisms on its own. If needed, the sessions can be secured using the Websocket over TLS/SSL. During the session, Websocket messages have only 2 bytes of overhead. As reported by relevant studies [108], the HTTP polling (in REST) repeats header information when the data transmission rate increases, thus increasing latency.

The goal of WAMP is to provide an open standard for soft real-time message exchange between application components and ease the creation of loosely coupled architectures based on microservices. It provides bi-directional, full-duplex communication channels over a single, persistent TCP connection. This protocol has four main features which provide different benefits: First, a single and persistent TCP connection that allows for higher performance since it does not require the need to set a new TCP connection for every request. Second, bi-directional communications that allow *Client* and *Server* both to send data independently of each other. Third, full-duplex communications that allow both *Client* and *Server* to send data simultaneously. Forth, the WebSocket protocol does not put any condition for the exchange of messages, which means any rules can be defined within the protocol and all messaging patterns are supported. With this in mind, WebSockets can be used to encapsulate other protocols based on TCP, such as: HTTP, MQTT or AMQP. In HTTP, the HTTP upgrade header is used in order to make the change from the HTTP protocol to the WebSocket protocol. In the case of MQTT and AMQP, these protocols need to be set up after the WebSocket connection is established.

Websocket is estimated to provide a three-to-one reduction in latency against the half-duplex HTTP polling, but it is not designed for resource constrained devices as the previous protocols and its Client/Server based architecture does not suit IoT applications [65]. However it is designed for real-time communication, it is secure, it minimizes overhead and with the use of WAMP it can provide efficient messaging systems and it can compete any other protocol running over TCP.

2.2.6.3 Alljoyn Protocol

Qualcomm recently introduced AllJoyn, an open-source, general networking framework which supports multiple direct networking technologies (Wi-Fi Direct, Bluetooth, etc.) [78], and enables ad hoc, proximity-based communication without the use of an intermediary *Server* [154]. Now, this open project is hosted by AllSeen Alliance[2], and many consumer brands has signed on [95]. AllJoyn's vision is to enable the IoE near users, which provides a software framework and a set of services that enable interoperability among connected products and software applications, across manufacturers, to create dynamic proximal networks [154]. The range of consumer products enabled by AllJoyn is very wide: From the mobile devices consumers always have with them, to the appliances and media equipment in their homes, to the electronics in their cars and the office equipment in their workplaces [151]. Technically, AllJoyn was developed as a mesh networking service which offers automatic discovery and communication for a number of different devices, agnostic of operating system [80]. And furthermore, the goal of AllJoyn is to be crossplatform with support for Android, OS/X, Windows variants, gaming engines and other thin clients.

However, the AllJoyn technology is not powerful enough to manage complex smart IoT environments [151]. In fact, AllJoyn does not scale well because it does not support communications among devices belonging to different broadcast domains and it does not provide any feature for the storage and real-time analytics of huge amount of data (Big Data problem). Thus, when the number of devices and information acquisition frequency increase, data management becomes quite hard [167].

2.2.7 Summary Over the Protocols

Summarizing the previous, in [40] it is claimed that CoAP is more Resource-friendly than MQTT. But, in terms of Message Oriented Approach (MOA), MQTT

[2]https://allseenalliance.org/

stands out and MQTT also needs less Random Access Memory (RAM) but more Central Processing Unit (CPU) load than CoAP. All the protocols mentioned above use TCP as transport layer and TLS/SSL as security layer. Only CoAP uses UDP and DTLS respectively [3].

Furthermore, according to [143], MQTT provides the smallest header size of two bytes, although it is based on TCP. Moreover, it provides three levels of QoS which puts this protocol in the first place in terms of QoS, even though it needs extra load in the network for message retransmission [40]. On the other hand, XMPP requires processing and storing XML data, which necessitates memory space too large for most IoT devices. In addition, HTTP performs better in non constrained environments when PC, Laptop and Servers are used. It is generally not applicable in IoT constrained devices due to its high overhead. AMQP [65], is more suitable for server-to-server communication than device-to-device communication. Websocket is neither a Request/Response nor a Publish/Subscribe protocol. In Websocket, a *Client* initializes a handshake with a *Server* to establish a Websocket session. The handshake process is intended to be compatible with HTTP-based server-side software so that a single port can be used by both HTTP and Websocket *Clients* [23]. According to [65], MQTT messages experience lower delays than CoAP for lower packet loss and vice versa. When the message size is small the loss rate is equal. AllJoyn [151], is a full stack of protocols intended for IoT. Though quite popular, the main disadvantage of AllJoyn is that the application protocol cannot be separated from the rest of the protocol stack. Due to this fact, Alljoyn is a complete framework and not only an application layer protocol. Thus, it is not taken into consideration in this thesis.

2.3 Access Control

Access control describes the process of providing security mechanisms to mediate every request to sources of a system in order to get access to information. Also, it can be describes as *a process by which use of system resources is regulated according to a security policy and is permitted only by authorized entities (users, programs, processes, or other systems) according to that policy* [132]. In order to control access requests the 40-year-old framework of the access matrix [56] has been extended as researchers have found it to be inadequate for their needs. This control over requests occurs by checking the relevant attributes only once when the access is initially requested. The evaluation is done by providing specific policies about the relevant attributes to each component. These policies are called security policies. The languages with which security policies are described are called security policy languages. Some examples in the literature are ABAC [60], Context Aware Access Control (CAAC) [164], Role-based Access Control (RBAC) [127], Task-Based Access Control (TBAC) [144] models, Risk-Adaptable Access Control (RAdAC) [36]. This paragraph briefly surveys RBAC and ABAC models and XACML that used in access control.

2.3.1 Role Based Access Control (RBAC)

RBAC is a widespread approach for regulating the access to information and resources [127]. The principal idea of this model is that a set of roles is created based on the application environment, where as an example, these roles can arise from the hierarchy of an organization or a company. Each subject is assigned to a role, depending on which, it is also entitled to a set of privileges. Hence, considering this example, subjects that are higher in the hierarchy have the possibility to perform more actions over the resources, whilst subjects belonging in the base of the hierarchy have limited access. The RBAC model can be characterized as flexible, since subjects can be reassigned to roles if needed and also privileges can be given to roles or taken from them considering the current state of the application environment. Another positive aspect of this model is that subjects can be also organized in groups based on their role or some common characteristics, while each group has its own permissions. As an example, a group can be the IT department of a company with permissions to modify user-names/passwords, but no permissions on changing data related to the salary of the employees.

Notwithstanding the benefits in efficiency [100], RBAC also comes with a certain amount of limitations. The inability to take into account time and location constraints, and the fact that in order to change the privileges of a user the role must be also changed, are a only a few examples commonly discussed in bibliography. Thus, to overcome these limitations, a new model came to fill the gap which is called ABAC [60] and considers many different attributes related both to the subject and the object, in order to grant or deny access to a resource.

2.3.2 Attribute Based Access Control (ABAC)

ABAC defines an access control example where access is granted to users through the use of policies enhanced by attributes. There are various types of attributes (e.g. subject, object, environment) [165]. Unlike RBAC, that defines specific roles describing the privileges associated with subjects, the difference with ABAC is the fact that the policy is expressed by a complex set of rules containing many different attributes. The values of these attributes can take various types (e.g. string, integer etc.). In the case of UCON, attributes may change values during the access. Although the concept itself existed for many years, ABAC is considered as a "next generation" authorization model because it provides dynamic, context-aware and risk-intelligent access control to resources [162]. Moreover, the policies in ABAC can include attributes from many different systems allowing enhanced flexibility and expressiveness.

Considering the attributes they can be of the following types:

- Subject attributes: attributes that describe the user attempting the access (e.g. age, clearance, department, role, job title).
- Action attributes: attributes that describe the action being attempted (e.g. read, delete, view, approve).
- Object attributes: attributes that describe the object (or resource) being accessed (e.g. the object type (medical record, bank account), the department, the classification or sensitivity, the location).
- Contextual (environment) attributes: attributes that deal with time, location or dynamic aspects of the access control scenario policies.

With ABAC you can have as many policies as you like that cater to many different scenarios and technologies and ABAC policies of this type can be expressed in formal languages such as XML [152].

2.3.3 Security Policy Languages

Since, a policy is defined as "a set of rules to administer, manage, and control access to system resources", these policy rules operate as "the binding of a set of actions to a set of conditions, where the conditions are evaluated in order to determine whether the actions are performed". Furthermore, these conditions are defined as "a representation of the necessary state and/or prerequisites that define whether a policy rule's actions should be performed, while this representation need not to be completely specified, but may be implicitly provided in an implementation or protocol".

Policy enforcement (i.e. the execution of a policy decision) is governed by the outcome of policy decisions (i.e. the rule-based evaluation of conditions), which are initiated by internal or external policy requests, leading to the execution of corresponding policy actions (i.e. operations) to affect and/or configure system specific operations and resource management. In order to provide fine-grained policy-based security management, the conditions evaluated within the policy rules must be matched against specific attribute values. Moreover, each rule can integrate a multitude of condition statements that must all be evaluated as "true" for the rule to become active providing a positive (or negative depending on the policy specification) policy decision.

Policy specification languages such as XACML [98], Ponder [30], KAoS [149], WS-Policy [5] and Rei [64] support capturing and evaluating within the policy rules a multitude of conditions, which if supported by the policy specification language may be matched against (i) subject, (ii) object, (iii) action/event and (iv) environmental attributes. These languages are highly expressive and designed to match natural language expressiveness, while still being machine enforceable. The exact values of those attributes can be alphabetic (e.g. object name) and/or numeric (e.g. subject age), while they can be integrated within the policy rules as exact values (e.g. subject rank = corporal), ranges of values (e.g. time = {08:00-16:00} - working hours), or sets of values (e.g. "MSc students" $\cup$ "BSc students"). These are only a few of the potential integration approaches supported by contemporary policy specification languages, while attributes can also be defined across various types, such as single valued (e.g. serial number), multivalued (e.g. color), composite (e.g. full name), or even null value attributes.

According to [67], there are several policy languages that are targeting to security and privacy. They are many and difficult to categorize them all by one characteristic only. Moreover, the organization of large amount of information, especially in the field of the IoE needs explicitly defining, enforcement and management of strategies and access granting. Since policy is considered as a set of rules that describe how to maintain a certain situation, a policy language is a set of syntax and semantics in order to express formally policies.

The categorization of policy languages can be done taking into account several characteristics but the four biggest categories of them are type, intention of use, scope, and design and implementation details. Considering the type of the subcategories are policy languages considering security, accountability, availability, privacy, data carriage, data usage control, and network and device management.

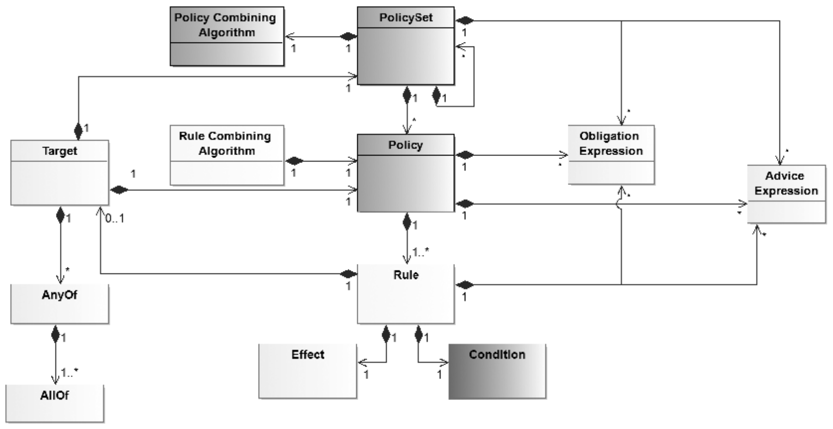


Figure 2.6: XACML Policy Language Model.

Furthermore, in terms of intention of use, the subcategories are those of users requirements, enterprise, and multiple parties interaction.

In addition to the previous, taking into account the scope of the language, the subcategories are data exchange, service requester/provider, agreement descriptions, authorization, access control, and application monitoring. Last but not least, in the category of design and implementation details, there are four subcategories, for which there exist more specific subcategories. In the first sub-category of usability, they are divided into human and machine oriented languages. In the second sub-category of context sensitivity, they are divided into context and non sensitive. In the third sub-category of syntax, they are divided into XML, high-level, functional language, and specific. Finally, in the subcategory of extensibility, they are divided into application specific and general purpose languages.

Considering the widespread area of classification rules for the languages above, the overall classification of a language must be considered via all the four categories and their subcategories. For example, there is XACML, which is a policy language of security and data control type, access control scope, enterprise policies intention of use, machine-oriented, XML based, application independent and context sensitive [98]. The policy language model of XACML is shown below in Figure 2.6. For more information readers can refer to [98].

Another example is, SecPAL, which is security and data control type, but of data exchange and authorization scope, all intentions of use, human-oriented, specific syntax, application independent and context sensitive [11].

2.4 Usage Control

In the IoE, all the application layer communication protocols use access control mechanisms. But, IoE devices can be very different in terms of hardware, communication protocol, software applications, operating system etc. Thus, it is very difficult to create a simple security policy for all of them. Moreover, the rapid change of *People* behavior and the constantly changes in the values of sensors or in the states of actuators lead to the fact that there is need of continuous control and re-evaluation of the attribute values in order to revoke granted access when necessary. The goal of UCON is to provide continuous access control. The net result is a plethora of seemingly ad hoc extensions without underlying intellectual unity [106]. The concept of UCON is comprehensive enough to encompass traditional access control, digital rights management, trust management etc. UCON unifies these areas systematically in a single framework and goes beyond in its scope [105]. UCON, [105], introduces mutable attributes and new decision factors besides authorizations; these are obligations and conditions. Mutable attributes represent features of subjects, object, and environment that can change their values as a consequence of the operation of the system [35]. The novelties that UCON provides differentiate it not only from other ABAC models, but also from CAAC [164], RBAC [127] and TBAC [144] models.

2.4.1 Usage Control Model

UCON model consists of various components which are shown in Figure 2.7 and they are described below [105]:

- `Subjects`: Entities associated with attributes that hold and exercise certain rights on objects. Attributes are properties of the subjects that can be used for the authorization process. Examples of attributes include identities, roles, credits, memberships, security levels, etc. A subject can be a user, a group, a role, or a process. A user is an individual entity that has certain rights on an object. A group is a set of users who holds same rights. A role is a named collection of users and relevant permissions [127]. Groups and roles may have hierarchical relationships. In UCON, there can be *consumer*, *provider* and *identified* subjects.

- `Objects`: Entities that subjects hold rights on, whereby the subjects can access or use objects. Objects are also associated with attributes, either by themselves or together with rights. Examples of object attributes are security levels, ownerships, classes, etc. Object classes are used to categorize objects so authorization can be based not only on individual objects but also on sets of objects that belong to same class [126]. In some cases, objects are associated with attributes together with rights that can be applied on them. Examples of the attributes for objects with rights are credits, roles, memberships, etc. In UCON, objects can be either privacy sensitive or privacy non-sensitive. Objects may have hierarchy on them.

- `Rights`: Privileges that a subject can hold on an object. Rights consist of a set of usage functions that enable a subject's access to objects. The authorizations of rights require associations with subjects and objects. Like subjects and objects, rights can also be divided into *consumer*, *provider* and *identified* rights.

- `Authorization rules`: A set of requirements that should be satisfied before allowing subjects' access to objects. There exist two kinds of authorization rules, the obligation and the rights related. The first is used to check if a subject has valid privilege to exercise certain rights on a digital object. Examples include identity verification, proof of payments, etc. The second is used to check if a subject has an obligation which has to be done after obtaining or exercising rights on a digital object. Examples include metered payment agreement, usage log report etc.

- `Conditions`: A set of decision factors that the system should verify at authorization process along with authorization rules before allowing usage of rights on a digital object. There are two types of conditions: *Dynamic* and *Static*. Dynamic conditions include information that has to be checked for updates at each time of usage, whereas Static conditions do not change. An example of dynamic condition is the number of usage times and an example of static condition is an allowed printer name.

- `Obligations`: Mandatory requirements that a subject has to consider after obtaining or exercising rights on an object. In real world implementations, obligations are agreed before obtaining the rights and at that time obligation-related authorization rules are checked. For example, a consumer subject is obliged to accept metered payment agreements before obtaining the rights

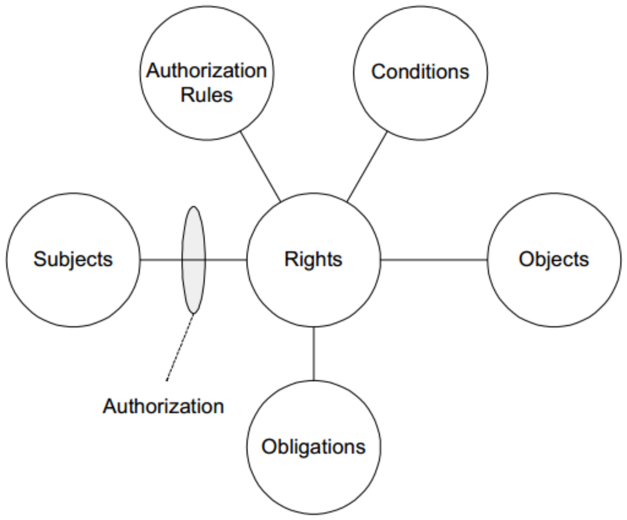


Figure 2.7: Usage Control Model Components.

for the usage of certain digital information. Access control has hardly recognized the obligation concept.

Since mutable attributes change their values during the usage of an object, UCON model allows to define policies which are evaluated before and continuously during the access. In particular, a usage control policy consists of three components that arise from the model above:

- `Authorizations`: Predicates which evaluate subject and object attributes, and also the actions that the subject requested to perform on the object. *Pre-Authorizations* are evaluated when the subject requests to access the object, while *Ongoing-Authorizations* are predicates which are continuously evaluated while the access is in progress.
- `Conditions`: Requirements that evaluate the attributes of the environment. In this case too, *Pre-Conditions* are evaluated when the subject requests to access the object, while *Ongoing-conditions* are continuously evaluated while the access is in progress.
- `Obligations`: Predicates which define requirements that must be fulfilled before the access *Pre-Obligations*, or that must be continuously fulfilled while the access is in progress *Ongoing-Obligations*.

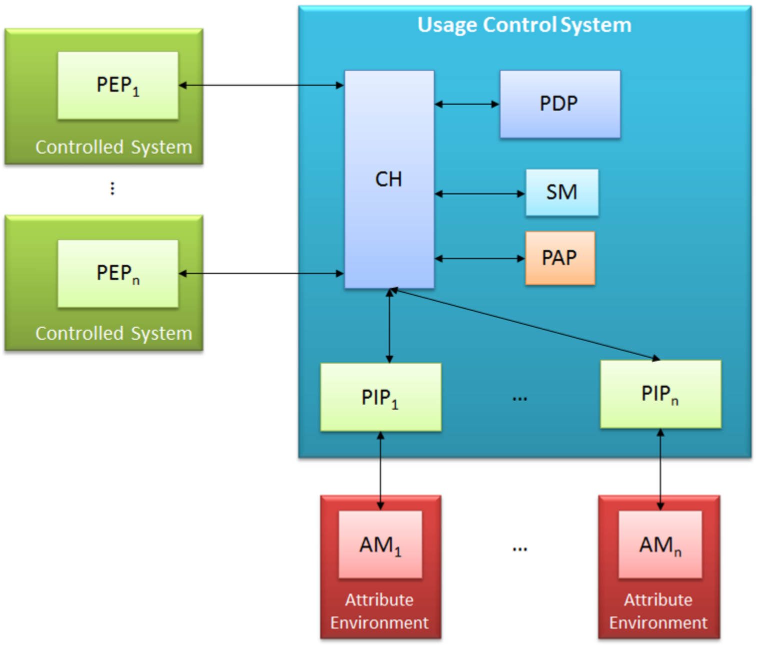


Figure 2.8: Usage Control Framework Diagram.

The continuous evaluation of the policy when the access is in progress aims at interrupting the access when the execution right is no more valid, in order to reduce the risk of misuse of resources. Hence, in the UCON model it is crucial to be able to continuously retrieve the updated values of the mutable attributes, in order to perform the continuous evaluation of the policy and to promptly react to the attributes change by interrupting those ongoing accesses which are no longer authorized.

2.4.2 Usage Control Architecture

The framework of UCON includes three main blocks, shown in Figure 2.8 [85]. In more detail, one of them has to provide the functionalities, another one communicates with controlled systems to check for the requests and the last one is responsible for acquiring the attribute values.

- `UCS`: The most complex block because it contains all the necessary mechanisms for communicating with other blocks, collecting information, and taking the decisions. In order to achieve this functionality UCS has also to provide some actions regarding to the access granting for a session.

- `Controlled Systems`: The components on which the UCON policy can be enforced. Each Controlled System communicates with the UCS issuing the request to access a resource by performing a specific operation on it. These components are called *PEP*s.

- `Attribute Environment`: This component interacts with UCS for attribute value retrieval but it is not controlled by it.

2.4.2.1 Components and Actions of Usage Control System (UCS)

Since, UCS is a complex component of UCON it is worth describing its components, which according to [87], are:

- `Policy Decision Point (PDP)`: Takes as an input an access (usage) request and an access (usage) policy returning the decisions which can be either *Permit*, or *Deny*, or *Undetermined.*
  - Permit
  - Deny
  - Undetermined

- `Policy Information Point (PIP)`: Retrieves attributes related to subject, object and environment of received access requests. Each PIP acts as the interface between the UCS and a specific Attribute Manager (AM) which is a non controlled component that has the values of the attributes that have to be acquired by each PIP. The latter has custom implementation for each specific application, each AM and the type of the attribute that should be retrieved. PIPs communicate with the Attribute Environment through AM which are not part of the UCS [17].

- `Session Manager (SM)`: Database which stores all the active sessions, with the necessary information to perform policy re-evaluations.

- `Context Handler (CH)`: The core of the UCS, responsible for routing messages among the various components. Firstly, it forwards the access request to the various PIPs for attribute retrieval, then the complete access to the

PDP and as a result to return the decision to the PEP. Finally, it receives a notification from PIPs when the value of an attribute changes and forwards the new value to the PDP for policy re-evaluation.

- `Policy Administration Point (PAP)`: Policy storage component. Policies can be included in the request made by PEP. Thus, this component is not mandatory.

In order to evaluate continuously the access requests, UCON provides the following actions [66]:

- `TryAccess`: Each PEP sends a request to the UCS so as to perform an action or access a resource, to be evaluated against a policy. After collecting the necessary attributes from the PIPs, UCS responds with a *Permit*, *Deny* or *Undetermined* decision. If the answer is *Permit*, the response is containing the SessionID for the session that will start. If the answer is *Deny*, the response is delivered to the PEP and the session is tracked in the SM as denied.

- `StartAccess`: Action invoked by the PEP having the SessionID as a parameter. This is the actual start of using the requested service. There is another evaluation from the PDP and, after an affirmative response, CH confirms the session to the SM as active.

- `RevokeAccess`: If the value of a mutable attribute changes, the PIP sends it to the CH for re-evaluation because it might change the policy decision. If this event occurs, the usage has to be revoked. CH informs both PEP and SM that this session is revoked. On the one hand, the SM keeps the session recorded but in an inactive state, whereas on the other hand the PEP blocks the usage to the resource.

- `EndAccess`: When the subject wants to terminate the usage of resources. When it is received by the UCS, it deletes the session details from the SM and communicates to the PIPs that the attributes related to that policies are not needed anymore, unless other sessions are using it.

2.4.3 Usage Control Policy Language (U-XACML)

In Figure 2.9 we can see the model of U-XACML which is the language used by UCON [26]. It is an extension of XACML which is standardized by OASIS [99].

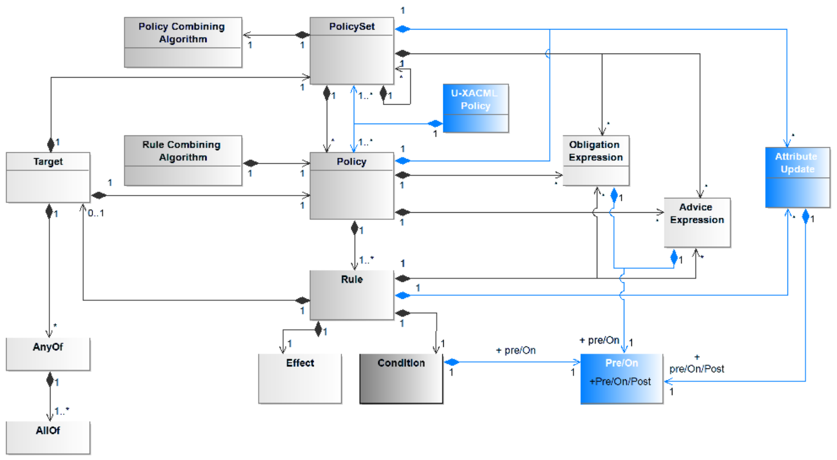


Figure 2.9: Usage Control Policy Language (U-XAMCL) Model.

In order to express the continuous control feature of UCON, U-XACML specifies the time when the access decision must be taken by the clause “DecisionTime” inside the tag “Condition” having as admitted values “pre, on, post”. To express the attribute updates, U-XACML defines also a new element called “AttrUpdates”, that contains a collection of single “AttrUpdate” elements to specify update actions. The time of update is stated by the element “UpdateTime” that has values of “pre, on and post”.

There is a set of attributes associating each of the following: Subjects, resources and environments. This description is done by the following elements: “AttributeDesignator” that indicates who issues the attribute and who is targeting this attribute to. Moreover, “AttributeValue” contains the actual attribute value.

The top-level policy elements are the following:

- `PolicySet`: is an optional element which provides the resulting policy via the combination of several <Policy> elements applicable to the access request. The decision made by UCON must be evaluated continuously when the access is in progress. The U-XACML policy specifies when the access decision is made by the “DecisionTime” in the <Obligation> and <Condition> elements.

- `Policy`: consists of one or more <Rule> elements. An access decision implied by the <Policy> is a combination of the result of the evaluation of each <Rule> it contains. XACML identifies several combining algorithms: deny-overrides, permit-overrides, first-applicable and only-one-applicable. The access decision produced by the <Policy> is accompanied with a set of <Obligations>.

- `Rule`: it has three main parts:
  - `<Target>` which denotes rule's applicability to an access request
  - `<Condition>s` which are predicates over attributes
  - `<Effect>` is the result of the access decision evaluation.

  A Rule returns either "Permit", "Deny" or "Non Applicable" if the <Target> and/or <Condition>s are not satisfied. Authorizations and conditions proposed in UCON are modeled in U-XACML by means of the <Target> and <Condition> elements. <Target> element puts constraints on the immutable attributes only, while <Condition> elements cover mutable attributes of a subject, object and environment.

An example of a U-XACML is shown in the Listing 2.1 below where we can see the items described above. For demonstration purposes the "on" and "post" conditions, since they contain the same content as the "pre" condition, they are not detailed. In Listing 2.1 we also see that the subject-id value must be a string equal to "SUBID" and the attribute that is resource attribute must be an integer greater than zero in order for the policy to result in a "Permit" decision.

2.4.4 Distributed Usage Control

IoT technologies improve our life and increase the quality of it. But a question that should always be taken into serious consideration is the way of protecting the access and sensitive data. Indeed, controlling the access and usage in the area of IoT has become of major importance [137].

Upon the assumption that data are stored in trustworthy places (data providers) we consider that there is a form of access control when there is a request to access this information. The question is how the consumers of the data can use that piece of information [8] [76].

A distributed system is considered as a system that is consisted of actors which are information processing devices and there is an owner of each actor and is responsible for its behavior. Actors are responsible for operations on the data or

```
<Policy xmlns="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17"
PolicyId="policyDep" RuleCombiningAlgId=
"urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:first-applicable"
Version="3.0">
<Description>policyDep</Description>
<Target></Target>
<Rule Effect="Permit" RuleId="rule-permit">
<Target><AnyOf><AllOf>
<Match MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
<AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">SUBID
</AttributeValue>
<AttributeDesignator
AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"
Category="urn:oasis:names:tc:xacml:1.0:subject-category:access-subject"
DataType="http://www.w3.org/2001/XMLSchema#string"
MustBePresent="true">
</AttributeDesignator>
</Match></AllOf></AnyOf></Target>

<Condition DecisionTime="pre">
<Apply FunctionId=
"urn:oasis:names:tc:xacml:1.0:function:integer-greater-than">
<Apply FunctionId=
"urn:oasis:names:tc:xacml:1.0:function:integer-one-and-only">
<AttributeDesignator AttributeId="att1"
Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource"
DataType="http://www.w3.org/2001/XMLSchema#integer"
MustBePresent="true"></AttributeDesignator></Apply>
<AttributeValue DataType=
"http://www.w3.org/2001/XMLSchema#integer">0</AttributeValue></Apply>
</Condition>

<Condition DecisionTime="ongoing">EQUAL TO pre</Condition>

<Condition DecisionTime="post">EQUAL TO pre</Condition>
</Rule>

<Rule Effect="Deny" RuleId="urn:oasis:names:tc:xacml:3.0:defdeny">
<Description>DefaultDeny</Description>
<Target></Target>
</Rule>
</Policy>
```

Listing 2.1: Example of an UXACML Policy.

communication not subjecting to usage control [109]. They can communicate between each other and may have different roles that can change dynamically.

In order to control the way on how data is used, the owners of data providers must define the appropriate UCON policies that include all the requirements that must be satisfied by data consumers that receive data from data providers. These policies must consider the following [13]:

- Data providers interests.
- Data owners preferences.
- Governing laws and regulations.
- Specific agreements between actors.

To this scope obligations are a very helpful tool of UCON in order to manage future requirements [59]. Since monitoring and enforcement are difficult in open infrastructures such as the Internet, and the IoT we assume implementations in infrastructures that already provide a structured form of communication and information distribution in IoT. These infrastructures are the IoT protocols. Also, this is because the information systems in these contexts are easier to control than systems in, say, public P2P networks. Moreover, for some of the actors in these contexts, it can be assumed there might be obligation violation.

With the continuous increasing amount of digital personal data, especially in IoT, we believe that usage control will play a key role on future technologies, particularly in the context of mobile computing and IoE. Any technical solution will likely come in conjunction with organizational, legal, and methodological support. Because most privacy regulations incorporate the notion of "purpose", this must be allowed for in the policy language, possibly based on dedicated ontologies [62]. Heterogeneous systems pose particular problems; it is unclear, for example, how an IoT device can control the usage of the resources. This is a goal that we try to achieve in this thesis.

2.4.4.1 Distributed Usage Control in the Internet of Things

In our framework, we target on a distributed UCON where the data of attributes are stored in trustworthy databases. Only UCS components have access to this gathered piece of information. This framework is based on a fully distributed P2P

UCON system [85]. It has been specifically designed to be implemented in IoT architectures with constrained devices and it is based on a group of smart-nodes which have their own logical architecture that can match UCON, as described in the previous sections.

In IoT most devices are constrained and they cannot host mechanisms to handle and manage access requests mainly due to the lack of computational power and limited storage capabilites.

With this architecture, we can have one separate UCS on each separate node. Furthermore, each node has its own access directly to *local* attributes that are accessed only by this node. Conversely, there are also the *remote* attributes the values of which can be remotely acquired by other nodes. The policies in the distributed UCON framework consider both attribute types.

By using this version of UCON framework, we have the ability to create subsystems that have a standalone UCS with its own PEPs and *local* attributes. At the same time, all of them are connected together in a distributed system which allows access to *remote* attributes [27]. All the nodes communicate with a distributed database that is hosting all the attributes. When every node needs to evaluate a policy that has remote attributes in addition to local attributes, it communicates with the distributed database so as to acquire the values of the remote attributes.

The functionality of this framework is presented in Chapter 3.

2.4.5 Why Usage Control

UCON is an implementation of the ABAC model [165]. Differently from RBAC model [127] where every subject gets only one specific role each time, UCON can evaluate different attributes simultaneously. Apart from the advantages of UCON previously described about attribute mutability and continuity of control, the fact that UCON is based on ABAC provides another advantage over other models: The ability to consider the role of a subject as an attribute, combining the feature of the role in RBAC model with the ABAC model, providing, thus, a generic attribute for each role to take into consideration when evaluating the overall policy as an attribute.

As mentioned above, except from the ABAC and the RBAC models, there is also the TBAC [144] and the CAAC model [164]. TBAC model provides access control via a task-oriented procedure instead of the traditional subject/object one. Furthermore, CAAC model provides again an application specific approach such as TBAC. The main disadvantage of TBAC and CAAC models, is that they are targeting on the application level. Hence, they must be adjusted specifically for each application which does not stand as a general solution to the heterogeneous environment of the IoE. Considering also the additional features of UCON, such as risk and trust awareness [70], it can provide a more general and holistic approach.

Moreover, it is worth noting that, there already exists a distributed UCON model [109] that focuses on how data usage control could be implemented in distributed systems. This is a proof that UCON can fit widely on different types of implementations and architectures which is crucial for the IoE environment that this thesis proposal is targeting at. The ability of UCON to be hosted in multiple machines with different components on each one of them is a powerful tool, but the challenge that still exists, is to be able to create both a local and a global model of UCON, simultaneously, in order to express and enforce local and global policies respectively.

2.5 If This Then That (IFTTT)

IFTTT is a free web-based platform used to create applet chains of conditional statements in IoT settings. Each applet chain, also called *Recipe* or *Applet*, is triggered by changes that happen within various web-services and as a result does specific actions on other web-services.

IFTTT creates chains of simple conditional statements that are called *Applets* and are triggered by changes that happen within web-services.An example is, when a user likes a video on Youtube, to add it to his/her Spotify acccount. After an *Applet* is triggered, there is an action that happens on another web-service. In fact, given a certain set of criteria, IFTTT gathers web-services in one place so that they can easily interact between each other [103]. IFTTT consists of the following structure:

- `Services`: The basic building block of IFTTT. They describe data and actions controlled by an Application Programming Interface (API), and each service has a specific set of *Triggers* and *Actions*.
- `Triggers`: The *"This"* part of the *Applet*, causing thus the triggering of the *Action*.
- `Actions`: The *"That"* part of the *Applet* which is the result of a *Trigger*.

Table 2.2: Comparison of IFTTT and its Alternatives.

Name	**Cost**	**Connections**	**Apps/ Devices**	**Number of Services**	**Access**
`IFTTT`	Free	One-to-One	Yes/Yes	>600	App/Browser
`Microsoft Flow`	Free/Paid	Multiple	Yes/No	226	App/Browser
`Zapier`	Free/Paid	Multiple	Yes/No	>1500	Browser
`Yonomi`	Free	Multiple	No/Yes	200	App
`Stringify`	Free	Multiple	Yes/Yes	70	App/Browser
`Workflow`	Free	Multiple	Yes(iOS)/No	N/A	App

- `Applets/Recipes`: The complete part of an example when a successful *Trigger* that leads to the execution of an *Action*.
- `Ingredients`: The data that are available after the triggering to guide the *Action*.

An advantage of IFTTT is that it can work with various platforms and devices in IoT. In order to create an *Applet*, the only necessary step is to mix *Ingredients* in such a way that they make sense so that the *Trigger* can interact correctly with the *Applet*. Then, there must be a definition of the pieces of information utilized by each *Trigger* and *Action*. A drawback of IFTTT is that it allows only a single *Trigger* and a single *Action*. The same *Trigger* can be used for other *Actions* but not inside the same *Applet*. But, in fact, creating an *Applet* is so easy, that it can be done via the specific application of IFTTT which is available on the major mobile operating systems.

There are various competitors of IFTTT and in Table 2.2 we see a summary of the pros and cons of them. More in detail, the biggest competitor of IFTTT is `Microsoft Flow`. It is free for up to 750 runs/month, it can allow multi step connection and works with many apps including Gmail, Facebook etc. and can be accessed either via apps or browsers. The cons are that, up to now, the apps are around 226 and it does not work with physical devices. But, it also supports Do/While and For/Each loops whereas IFTTT supports only If conditionals. Hence, it is more difficult and complicated to operate. Another example is `Zapier` that works only with apps and not physical devices. It allows multi step connection between devices and has a simple free usage up to 100 runs/month and also paid plans of use. On the contrary, `Yonomi` works only with physical devices and not with apps, but allows multi-step connection between the about 100 compatible devices, it is free and focuses more on home applications. Furthermore, `Stringify` is also free and can host multiple connections but has only about 70 services and cannot be accessed via a browser like IFTTT but only via an app. Finally, there is also `Workflow` that works only with iOS apps and allows multiple step systems. It is again free but there is not a list of compatible services but only the most famous iOS apps are used like Safari, Photo Gallery, Facebook etc.

2.6 Related Work

IoT is a paradigm which includes applications spanning from e-health to industrial controls. IoT architectures are distributed targeting on constrained devices. The different nature of these devices together with the additional complexity that the components of IoE add, makes the introduction of security mechanisms very difficult, especially when considering the requirement of dynamic policy (re-)evaluation.

2.6.1 Related Work Regarding Protocol Security

There are some efforts considering the enhancement of the security of IoT protocols, that are responsible for distributing information. In [81], the authors present a survey about the privacy and security challenges targeting on Smart-Home Environment which is a part of IoT. They focus on the existing solutions and the problems that they have. These solutions are the standard implemented solutions that exist on the basic implementations of the protocols. This means that in the case of protocols, like MQTT or CoAP, they continue to face the problems that we address with the solutions that we propose below.

Yet, no up-to-date solution is targeting on how UCON can be architecturally integrated in the IoT protocol functionality like the ones that we propose below [73], [72], [116]. In addition to the previous, a more efficient solution should try to prove that UCON can fit not only in one protocol, but also in other protocols with different architectures simultaneously, providing interoperability and attribute sharing among different protocols. Thus, such an effort will give the opportunity for the first time to provide one compact and complete security enhancement solution for different protocols together in IoT [116].

2.6.1.1 Related Work Regarding MQTT Protocol

The most significant effort of securing MQTT is called SMQTT [135]. In this variant of MQTT the goal is to augment MQTT protocol by using extra security features based on Key/Ciphertext Policy-Attribute Based Encryption (KP/CP-ABE) using lightweight Elliptic Curve Cryptography. This type of lightweight Attribute Based Encryption, needs extra overhead caused by the time and the computational power that is significant considering the constrained devices used in IoT. SMQTT has to encrypt data continuously. This continuous encryption leads to significantly worse resource usage. Thus, a solution that does not produce such a significan overhead should be favored. Moreover, since Publishers can be very constrained devices, encrypting the data on such devices needs significant computational power or specific hardware components. Hence, such an action may cause battery draining and instability of IoT systems. Moreover, since SMQTT needs specific Publishers and Subscribers in order to (en)decrypt data it cannot offer a security enhancement in the existing or non specifically equiped IoT systems. Finally, our solution can work with any type of Subscribers of Publisher which enhances the generic nature of our solution [73], [72]. The solution that we propose below takes advantage of using computational power provided by the Broker which by default has more computational power than the Publishers and Subscribers. Furthermore, it does not require specific Publishers or Subscribers and it does not alter their functionalities.

Moreover, the authors of [79] propose a solution to securing Smart Maintenance Services. Their goal is to proactively predict and optimize the Maintenance, Repair and Operations (MRO) processes carried out by a device maintainer for industrial devices deployed at the customer. They focus on the MQTT routing information asset and they define two elementary security goals regarding the client authentication. Their solution is based on TLS which is already a basic feature of the protocol. They proposed on how to use it more efficient as a hardware element. Although they claim that the performance impact is not significant, the adoption of an additional hardware component might be critical in the constrained environment of IoT whereas the solution that we propose does not require specific components or additional hardware [73], [72].

Apart from the previous, in [138], the authors present the adoption of EventGuard in order to secure generally Publish/Subscribe overlay services. EventGuard is a dependable framework and a set of defense mechanisms for securing a Publish/Subscribe services. It comprises of a suite of guards to enhance security. But their solution does not target on a specific Publish/Subscribe protocol like MQTT, but general in these type of protocols which means that they are not targeting specifically IoT constrained protocols.

2.6.1.2 Related Work Regarding CoAP Protocol

Over the past few years a number of research works have presented new ways to improve the security of CoAP. Most of these works are focused on reducing the overhead of using DTLS on top of CoAP. In [111], the authors introduced *Lithe* which proposes improvements in the integration between CoAP and DTLS to allow increased performance and more efficient packet sizes and energy consumption. In [148], the authors propose a lightweight security scheme in CoAP using Advanced Encryption Standard (AES) 128 symmetric key algorithm, introducing an object security (payload embedded)-based authentication mechanism with integrated key management. Finally, in [16] the authors introduced RESTful DTLS connections as CoAP resources using Elliptic Curve Cryptography (ECC)-based cryptography, achieving gains in Read Only Memory (ROM) and RAM occupancy. The IETF supports efforts to secure CoAP, like DTLS for CoAP [42], or the Object Security for Constrained RESTful Environments (OSCORE) group which is developing a mode for protecting group communication over CoAP [145]. The new DTLS standard (1.3) adds improvements in performance and security [114] which CoAP can take advantage. However our solution, provides a continuous control monitoring mechanism, that according to the values of critical attributes, can revoke the access when policy is violated [116].

2.6.2 Related Work Regarding Usage Control in the Internet of Things

IoT includes applications spanning from various areas with a large variety of architectures targeting on constrained devices. Although there exist applications of UCON in GRID [87], Cloud [17] systems and SIP-based Multimedia delivery [66], there is only one targeting on IoT [85]. In that work, the authors present a version of UCON called Usage Control in the Internet of Things (UCIoT) that aims to bring the UCON on IoT architectures. As also the authors of [109] state, UCON can be distributed which means that the UCIoT framework can deal with heterogenous and distributed architectures of connected devices. Their work mainly focuses on an implementation of UCIoT in a smart home environment. They present specific policies alongside experiments on testbed. They do not, though, state how this framework (UCIoT) can be applied to the several application protocols. Their implementation in a P2P environment does not state how the UCON framework deals with each protocol. The solutions that we propose in the next chapter address this lack of addressing these protocols and how UCON can work alongside them.

Considering, obligation standardization and interoperability over devices and services in the area of IoE, the framework described above considers an application of UCON targeting there [85], but the authors there focus more on implementing a specific version of UCON that focuses on IoT, whereas our work does not need any specific modification on UCON framework and can be used not only on smart-home environments. Furthermore, considering the interoperability, with the vast variety of IFTTT *Applets* that exist, the solution that we propose targets on enabling a very wide range of applications instead of only revoking the access. The authors propose a distributed model of the standard UCON framework, discussing a smart home use case.

The focus of these works, however, is not centered on obligations, they only exploit in an appropriate way the authorization constructs. Considering IFTTT an effort to create simple policy algorithms for IoT is presented in [94]. This work focuses more on the policy specification and how it could be predicted according to the user features and the specific IoT domain. This work does not study the way the policies are written or the obligation format and context but focuses more on the part of how to create a policy according to what is happening in a specific IoT domain. Furthermore, enforcement mechanisms are not considered in their analysis.

In [153], the author describes the development of a specific framework that can modify the concepts of the IFTTT platform so as to provide services that can be used in order to make home automation systems more secure. This work, though, creates specific services in the IFTTT platform and also depends only on the security mechanisms of IFTTT without any control on the installed environment and, also, no continuous way of evaluation or policy enforcement. In [140], the authors describe also some possible security and privacy risks in the *Applets* of IFTTT. In our case, in order to control whether our system has been compromised or not we can always compare the amount of executions of the *Applet* that we expected with the actual ones that really happened from the history of the IFTTT platform. Also, via the handling of the data included in an obligation we can always have control of what data are being shared in the IFTTT platform.

Concerning the XACML model for policies, in [31], the authors present a description of obligation expression based on examples in Grid and networking security. The work is very specific to this limited environments and does not provide a way to formalize the obligations semantic. Finally, in [18], the authors present a variation of XACML specific for obligation extraction. Firstly, they propose a method of creating a new file for obligations in XACML which is produced by the initial file but they do not give a clear description of how the obligation part should be constructed so as to be easily readable by both the PDP and the PEP. Though interesting, their approach is not based on standard XACML, differently from our solution that integrates in XACML 3.0 policies without requiring any modification to the standard architecture and workflow.

Additionally, they try to define a set of relations between obligations and parameters to describe them but they do not provide any formalized way of expressing them. Also, they do not consider continuous evaluation with obligations that could be fulfilled before, after or during an action. As a matter of fact, they do not provide examples, timings or use cases on how their approach can be applicable in enforcing the obligations within the PEP.

2.6.3 Related Work Regarding Efficient Policy Management of Usage Control in the Internet of Everything

The area of IoT includes various protocols and applications that target to a wide range of constrained environments. Thus, the complexity to create security enforcement tools that can handle policies in such environments increases. It becomes more complex when the security policies have to take in to account dynamic environments.

Multiple challenges related to information security have been solved through application of Computational Intelligence (CI) and problem solving approaches. The main motivation for this is the ability of such methods to automate processing tasks and facilitate faster decision making, which have been completed before mostly manually by domain experts [129]. With growing complexity of policy management involving large number of attributes and Object/Subject properties, one needs advanced models capable of both efficient timely handling of data processing as well as delivering accurate and reliable results over time. Therefore, so-called Hybrid Intelligence (HI) can be utilized to mitigate weaknesses of standalone methods and bring optimized solutions under constraints [1]. The basic principle of HI is to use multi-stage solution capable of re-training and adjusting the whole decision model when new Objects/Subjects are added, properties changed, etc.

2.6.3.1 Decision Making Algorithms

Decision Making process has been proven to be a very challenging task in the information security field, mainly because of the large amount of information which need to be taken into account and the demand of a deeper understanding of the potential trade-offs which the decision makers have to consider in order to reach the optimal solution. To this end, a number of techniques have been developed over the years, aiming at overcoming the limitation of human processing capabilities and providing to the decision makers the possibility of structuring complex problems and evaluating all the available alternatives. These methods belong to a category of algorithms known as Multi-Criteria Decision Analysis (MCDA) and are briefly presented below.

A comprehensive survey on the MCDA and their various categories was presented by Greene et al. in [53]. Even though their study is focused on the geographic information systems, they also provide an analysis on the various MCDA techniques, their main characteristics and their distinction. Two main categories which can be identified are the Multi-attribute decision making methods (MADM) [161] and the Multi-objective decision making methods (MODM) [4], where the first having a single objective evaluate a number of different criteria and attributes and the second are used in problems with multiple, often conflicting, objectives.

Another level of distinction is the one among the non-compensatory and the compensatory approaches. The methods belonging to the first category do not consider the possibility that the benefits on some attributes can overbalance shortfalls of others [84]. On the other hand, compensatory methods allow the aforementioned trade-off giving thus a level of freedom to the decision maker.

An additional category is the one of the weighting methods, where usually an expert of the application environment is asked to provide a certain level of preference or weight to each of the criteria, before an aggregation takes place. The most known method belonging to this category is the Analytic Hierarchy Process (AHP) [119] [121] which provides a way to structure into an hierarchy the criteria and to compare them pair-wise by assigning to them relative weights with respect to a final goal. Finally, worth to be mentioned are the outranking methods whose outcome is a ranking of the alternatives based on a concordance-discordance principle, meaning that an alternative is better than another if the majority of the attributes reinforce this claim (concordance) and if the minority cannot offset it (discordance) [15].

Chapter 3

Distributed Usage Control and Internet of Things Communication

This chapter presents our work towards enhancing IoT security via UCON. Firstly, we used the standard UCON to add it in the MQTT protocol. After identifying the limitations of UCON and the advantages that distributed UCON can offer, we used it for our further efforts. Hence, we present the addition of UCON in MQTT protocol followed by the addition of distributed UCON in CoAP protocol. Then, we present the addition of distributed UCON to both protocols simultaneously. Finally, we present UCON interoperability and obligation standardization via expressing enforcing UCON obligations through IFTTT.

3.1 Addition of Usage Control in IoT Application Layer Protocols

In this section we present our work of adding UCON in IoT protocols providing the architectures, the workflow, use case scenarios and timing evaluation. We will discuss firstly how UCON can be installed in a couple of protocols and then how it can work in a more complex environment of coexisting ecosystems that are communicating via different protocols.

3.1.1 Usage Control in MQTT Protocol

This part describes the architecture of including UCON in MQTT, presenting first the model, then the operative workflow and the performed implementation.

3.1.1.1 System Model

As previously mentioned, MQTT protocol is based on the Publish/Subscribe model, thus the entities participating to the protocol can act either as *Publishers* or *Subscribers*. *Publishers* could be sensors or other devices which collect and provide specific data, when available, periodically or even as a stream. *Subscribers* are instead entities that register to the *Broker* to receive, when available, specific data or set of information grouped under a *Topic*. The *Broker* acts as middleware and coordinator, managing the subscription requests and dispatching data to *Subscribers*, when made available by prosumers.

The MQTT protocol supports ID and password-based authentication for both *Publishers* and *Subscribers*. The enforcement is performed on *Broker's* side, which keeps track of the ID and authentication password of authorized *Publishers* and *Subscribers*. However, we argue that this authentication model is too simplistic and coarse grained, making it impossible to check the right to access information over time. In fact, once a *Subscriber* has been authorized, the subscription remains valid until the *Subscriber* explicitly invokes an `unsubscribe` for the topic(s) it was registered for. The same goes for *Publishers* which keeps the right to publish continuously or on demand, till they have valid credentials. In real applications, several features might imply a condition for a subscription to decay, or for a publication to be denied. Detected *Publisher* malfunction or corruption, conditions on time spans in which a subscription should be allowed, and *Subscriber* reputation, are just few examples of aspects on which a more complex policy should be enforced.

To enforce policies with similar conditions to the aforementioned ones, and to have the possibility of revoking a subscription, usage control has been added to the MQTT logical architecture.

In Figure 3.1, we depict the logical architecture of the proposed framework.

As shown, the UCS is physically integrated in the *Broker* Device, i.e. the physical machine that is hosting the *Broker* software, which enables the MQTT protocol. It is worth noting that we consider in this example three abstract PIPs, which are conceptually grouping the PIPs reading attributes related to the subject (PIP_S), to the resource (PIP_R) and to the environment (PIP_E). The PEP is (par-

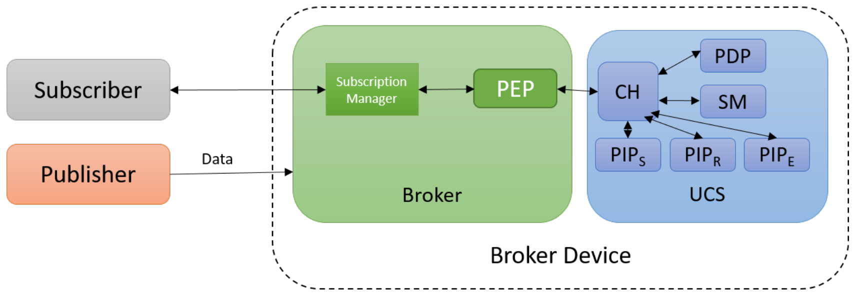


Figure 3.1: Usage Control in MQTT Architecture.

tially) embedded in the *Broker*, to dynamically control the subscription events. In particular, the PEP will intercept the subscription events and interact directly with the *Broker* subscription manager, deleting and inserting the entries for *Subscribers* from the list of authorized ones, according on the UCS decision. In such a way, the PEP ensures that no *Subscribers* can register by avoiding the enforcement of the usage control policy. Since the PEP is embedded in the *Broker*, the proposed architecture remains compatible with any implementation of MQTT *Subscribers*. The only requirement is that the *Subscriber* is configured to access with username and password, otherwise the connection will be refused by the *Broker*.

3.1.1.2 Operative Workflow

In Figure 3.2, we report the envisioned workflow. For the sake of simplicity, we will consider a simple system made out of a *Broker* and a single *Publisher* and *Subscriber*.

The workflow is initiated by a subscription request from the *Subscriber* to the *Broker*. This request is intercepted by the the PEP, which interprets it, so as to take the credentials of the *Subscriber* that are needed in order to create and send the request to the UCS for evaluation. Hence, the PEP invokes the `TryAccess` sending to the CH request and policy. The request is eventually filled by attributes retrieved through the PIP, then is sent, together with the policy, to the PDP for evaluation, which should return a Permit or Deny decision. In case of Deny, the subscription request is dropped and the *Subscriber* will be notified, as if a wrong username/password has been inserted. In case of Permit, the SM creates the session and sends its ID to the PEP (via the CH) which is informed about this decision and performs the `StartAccess`. Supposing a permit decision has been received, the

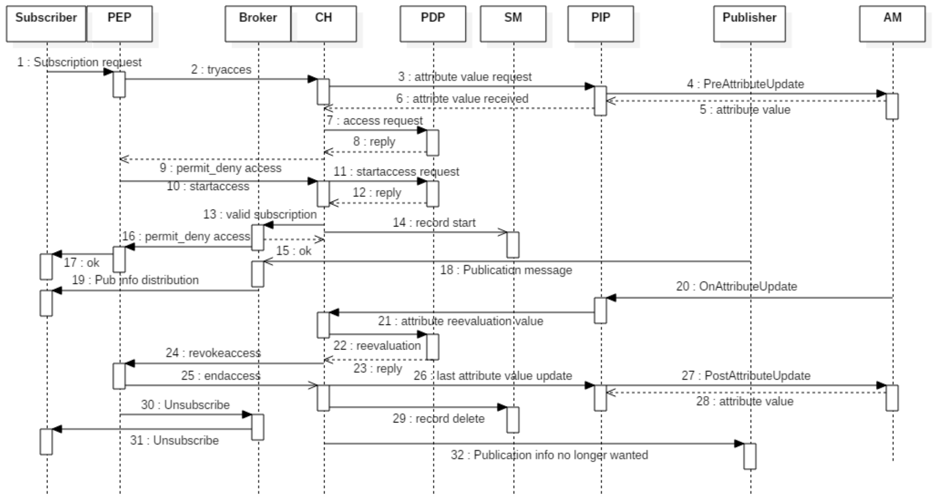


Figure 3.2: UCON in MQTT Workflow Diagram.

Broker informs the *Subscriber* about the successful subscription and starts to send data related to the topic when available, eventually stimulating *Publishers* in an idle state.

To illustrate the `revoke` workflow, we suppose that one of the attributes relevant for the *Subscriber* policy changes its value `(OnAttributeUpdate)`. This causes the PIP to send this new attribute to the CH that forwards it to the PDP for reevaluation. Supposing that the value of this attribute leads to a conclusion that this session must be revoked (Deny decision), the CH invokes the `RevokeAccess` on the PEP, also informing the *Subscriber* that the access is no longer granted `(RevokeAccess)`. The termination of the access could happen also if the *Subscriber* is no longer interested to the data, invoking the `Unsubscribe`. The unsubscribe triggers the PEP to send an `EndAccess` to the CH. The latter informs the PIP to take the last value of the attribute `(PostAttributeUpdate)`. Also the UCS informs the *Broker* that the *Subscriber* is no longer subscribed and forces the unsubscription of this specific *Subscriber* from the *Broker*. Moreover, the SM is also informed that this session is over so that the record should be archived or deleted Finally, if this *Subscriber* is assumed to be the only one that was interested to the *Publisher*, the *Broker* informs him to stop data publication due to fact that there is no more any interest from any *Subscriber*.

We point out that the simplification of considering a single *Publisher*/*Subscriber* does not harm generality. In fact, the protocol is not modified and by using multiple *Subscribers*/*Publishers* do not introduce any additional criticality, since both MQTT *Broker* and UCS can support multiple components natively.

3.1.1.3 Implementation of Usage Control in MQTT

As previously mentioned the UCS is a Java-based configurable framework, easy to integrate in any system with a Java runtime environment. The software used to implement the *Broker* is the open source MQTT *Broker* Moquette[1]. Though not largely used as the Mosquitto[2] *Broker*, Moquette is easier to integrate with the UCS framework, since they are based on the same programming language. The *Broker* has been partially modified to include in it the PEP functionalities. In particular, the subscription request is intercepted by hooking the subscription handling method, as to invoke `TryAccess` and `StartAccess` and waiting results before allowing or denying the subscription. If a Deny decision is received, the *Broker* will return a *wrong user/password* message to the *Subscriber*.

If there is a policy violation, the `RevokeAccess` is invoked. Hence, the PEP calls the Unsubscribe function so as to prevent the *Subscriber* from receiving messages, while the `EndAccess` is invoked to remove the session details on the UCS side.

In Figure 3.3, is depicted the architecture of our testbed. In one RaspberryPi[3] (central in Figure 3.3) we run the *Broker* which includes the PEP, and the UCS as Java ARchive (JAR)s. The code of the *Subscriber*[4] and the *Publisher*[5] were running unmodified in different Raspberries.Furthermore, additional tests have been performed by having the *Subscriber* host in an Android application called *MyMQTT*, which can be accessed through Google Play. Hence, the *Subscriber* code can be almost completely executed in the same device of the *Publisher*. Moreover, it is or the latter can be a small sensor that gives the data to the *Broker* as shown in Figure 3.3. Since the framework is general, none of these configurations affects the functionality or requires any modifications to the framework.

[1] https://github.com/andsel/moquette
[2] https://mosquitto.org
[3] http://raspberry.org
[4] https://github.com/pradeesi/MQTT_Broker_On_Raspberry_Pi/blob/master/subscriber.py
[5] https://github.com/pradeesi/MQTT_Broke_On_Raspberry_Pi/blob/master/publisher.py

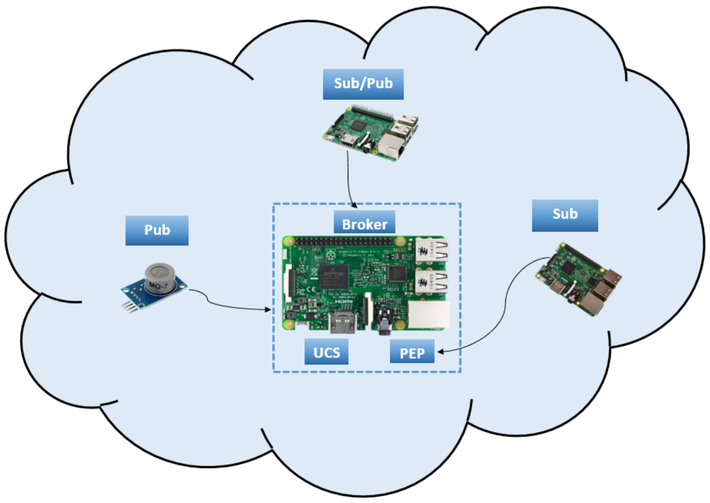


Figure 3.3: UCON in MQTT Testbed Logical Representation.

3.1.1.4 Experimental Evaluation of Usage Control in MQTT

To demonstrate the viability of the proposed approach, the overhead introduced by UCON has been measured in a simulated and in a real environment. The framework has been tested in two different environments: the first one is a virtual machine installing Ubuntu 16.04 64-bit, equipped with an Intel i7-6700HQ with 8 cores enabled, 8GB-DDR4 RAM running in 2133MHz. The second one is a Raspberry Pi 3 with a Broadcom ARRMv7 Quad Core Processor running on 1.2GHz and 1GB of LPDDR2 RAM on 900MHz, running official Raspbian as operative system. The *Publisher* and the *Subscriber* were installed in two other Raspberries.

The complete set of results is in Table 3.1. All values have been extracted as the average times computed on 10 runs of the framework in every setting. The first column describes the title of the timings which are all described in milliseconds. The second column describes the timings when the Raspberries are used, and the third one the scenario where we used the Desktop-PC.

In Table 3.1, the detailed timings are reported, considering a policy with a single attribute. In Figure 3.4 and Figure 3.5 are reported the performance variation at the increase of the number of attributes used in the policy. As shown, the timing behavior is almost linear to the amount of attributes, which is expected, due to the longer time needed to collect a larger number of attributes and for the evaluation

Table 3.1: Timings for comparing results between Raspberry and Desktop experiments.

Event Timings (ms)	**Raspberry**	**Desktop**
Total Tryaccess Time	770	91
Total Startaccess Time	169	26
Total Subscription Time	969	121
UCON Subscription Part	939	118
MQTT Subscription Part	30	3
Total Endaccess Time	211	27
Unsubscribe Time With UCON	213	27
Unsubscribe Time Without UCON	2	0
Revoke Duration in *Broker*	216	27
Revoke Duration on UCON	455	41

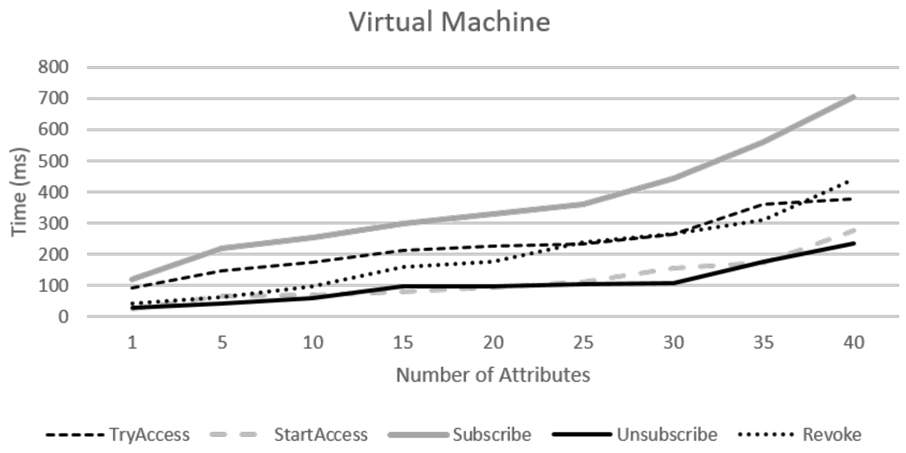


Figure 3.4: Timings on the Simulated Testbed.

performed by the PDP. However, in the real case, even considering 40 attributes, the timings are still acceptable for most of applications. Moreover, it is worth noting that policies with a large number of attributes such as 40 are quite unusual [85].

As expected, the low computational power of the Raspberry alongside the existence of a real network among the MQTT components, explains the longer timings than in the simulated environment. However, also considering a limited amount of

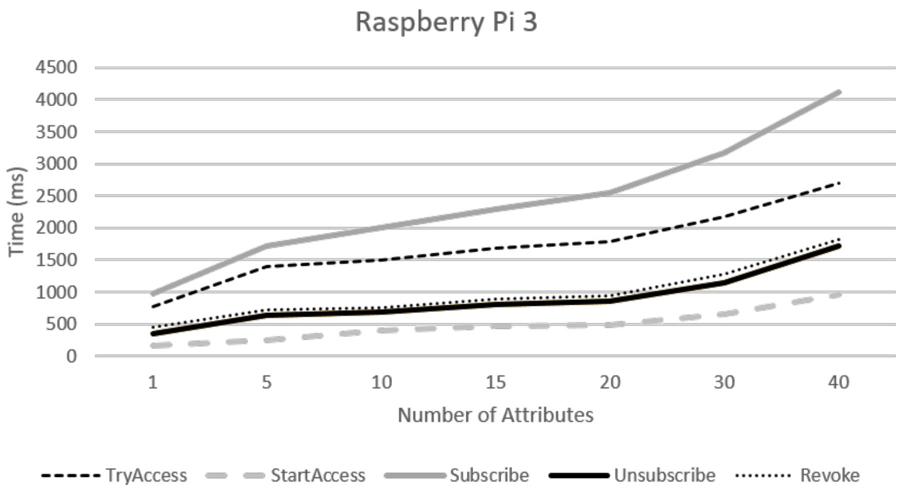


Figure 3.5: Performance on the Real Testbed.

attributes which is usual as mentioned above the overhead is slightly bigger than one second.

Considering the subscription time, we see that there is some overhead caused be UCS. This is not considered as a constraint because, since the *Broker* provides a buffer, we can still send all the published messages between the time of the request and the actual acceptance of the *Subscriber*. This causes no packet loss to the *Subscriber* and high QoS. Furthermore, the most significant time is the one of the revocation. This time is in fact the actual time in which the policy is violated and should be minimized. As shown, this time is equivalent to 216ms in the real use case and 27ms in the Virtual Machine, considering a policy with a single attribute. For several applications, this time can be considered as negligible. As shown, the time between a non-valid value is taken and revocation of the access is very small. Finally, it is worth mentioning that in the ongoing phase, i.e. after a successful `StartAccess`, no delay is introduced by UCON while delivering messages to the *Subscribers* independently also of the number of attributes.

3.1.2 Distributed Usage Control in CoAP Protocol

CoAP protocol follows the Client/Server model. Each *Client* can either request and/or provide data to the CoAP system. *Clients* could be very constrained devices

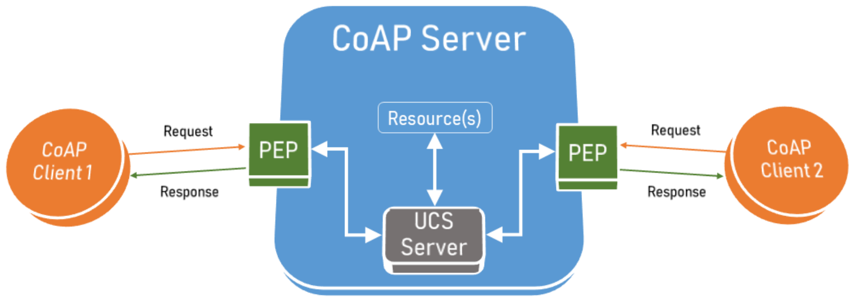


Figure 3.6: Architecture Diagram of Usage Control in CoAP Protocol.

whereas the *Server* has to provide enough computational power to support all the communications and actions of the CoAP system. The registration and management of the *Clients* is performed by the *Server* that assigns unique tokens to each *Client*. Yet, this model is very simple and does not provide any mechanisms for checking the access during time. All the *Clients* remain connected as soon as they firstly register correctly. In reality, there might occur cases in which sensitive information should not be delivered to specific *Clients*. To achieve such continuous monitoring of the access rights of the *Clients*, the CoAP architecture has been enhanced by the addition of UCON.

In Figure 3.6 we present the architecture of the proposed enhancement in CoAP architecture. The UCS is in the same device with the *Server* of CoAP. UCS can communicate also with other UCSs that can provide the values of remote attributes. The component that performs this communication is CH. There are various PIPs that are receiving the values from the various AMs regarding subject, resource and environmental attributes. The PEP is also installed in the *Server* and every new request that comes from a *Client* is firstly sent from the *Server* to the PEP for being evaluated by UCS. For every session of the UCS there is a unique sessionID assigned and stored in the SM. This unique sessionID is matched with the unique token assigned to each *Client* by the CoAP *Server*. The PEP enforces the decision of the UCS by interacting accordingly with the *Server* that manages the subscriptions. Thus, we ensure that there is no chance that a *Client* can be registered in the system without being monitored by UCON. Furthermore, we manage to achieve our goal without making any modification to the *Clients*.

3.1.2.1 Operative Workflow of Distributed Usage Control in CoAP

In Figure 3.7, we report the workflow of an instance of UCON to a system that uses the CoAP protocol. For the sake of simplicity, we will consider a simple system made out of one *Client* that publishes information (CLIENT_P) and another one *Client* that requests for information (CLIENT_R) which is marked with red color on the right part of Figure 3.7.

The workflow starts with the request for access by the CLIENT_R to the *Server* (Task 1). When the request arrives to the *Server* it initiates the communication with the local PEP (PEP_L) (Task 2). At this point, we should mention that the components of the local UCS (UCS_L) are marked with green color and have the (L) identification to their names, whereas the components of the remote UCS are marked with light blue color and have the (R) identification to their names. As soon as the request is received by the PEP_L, it communicates with the UCS_L by performing the *TryAccess* action to the CH component of the UCS_L, which is marked in the workflow as CH_L, containing the request and the policy (Task 3). The request has to be enhanced with the attribute values that the CH requests from the various PIPs that take them from the specific AMs (Tasks 4-5). If the value cannot be provided by a local PIP (PIP_L), the CH_L communicates with a remote CH (CH_R) of a remote UCS (UCS_R) (Task 6). CH_R is responsible to acquire the attribute value from PIP_R (Task 7-8) and return it to the CH_L (Task 9). Then all the above are sent to the local PDP (PDP_L) to be evaluated (Task 10) and the PDP_L replies with the result to the CH_L (Task 11). Considering the result is either *Permit* or *Deny*, the request is approved or not accordingly and the CLIENT_R is informed about this (Tasks 12-14).

Then, in the case of a *Permit* in the previous request, the CLIENT_R performs another request to obtain data, which in our case will be provided by the CLIENT_P (Task 15), after the *Server* communicates with PEP_L (Task 16) and gets a *Permit* on that request. For this to happen, PEP_L must perform the *StartAccess* to the CH_L which again passes through evaluation by the PDP_L as previously (Tasks 4-11). Supposing that CLIENT_P starts sending data to the *Server* (Task 20) that must be delivered to CLIENT_R, the *Server* distributes them without any interference by UCON (Task 21). In the meantime, UCS_L is performing a continuous re-evaluation of the policy (Task 22). In the case that there is a policy violation, the access of CLIENT_R should be revoked (Task 23). The CH_L informs the glspep_L (Task 24) and the latter informs the *Server* (Task 25). The

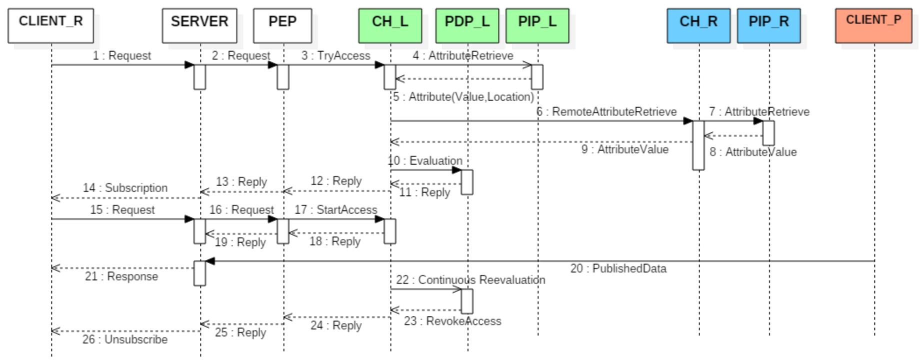


Figure 3.7: Workflow Diagram of Usage Control in CoAP Protocol.

Server deletes CLIENT_R from its authorized *Clients* and informs it about this fact (Task 26).

3.1.3 Usage Control in Other Protocols

In the previous sections we proved that both UCON and distributed UCON can be added in the workflow of the protocols taking as an example one protocol of each communication pattern (MQTT for Publish/Susbcribe and CoAP for Client/Server). Accordingly UCON can be added in the other protocols mentioned above since they must follow one of the mentioned communication patterns. Since the pattern does not change, by following our previous work we can add UCON in each protocol simply by adjusting the previous procedure to the specific components and architecture of each protocol.

3.2 Application of Usage Control in Combination of IoT Application Layer Protocols

To the best of our knowledge, up to now the efforts for policy enforcement and continuous monitoring of the access in IoT application layer protocols are still limited. In this part, we present our work on increasing the security of both CoAP and MQTT protocols with dynamic policy enforcement of UCON policies together with the ability of sharing attribute values for access evaluation purposes between protocols [116]. We demonstrate the general methodology that proves the ability

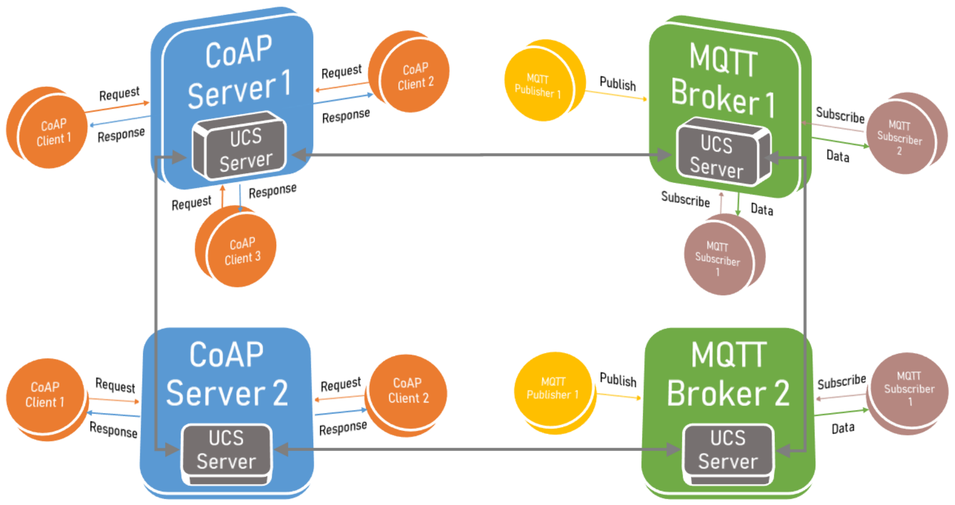


Figure 3.8: Architecture Diagram of the Proposed Framework.

of integrating UCON in a seamless way without modifying the protocols. In order to demonstrate the viability of our approach we present also a real implementation of our framework together with performance evaluation.

3.2.1 Architecture

Taking advantage of the distributed version of the UCON framework [85], we created a similar architecture in our framework in order to combine simultaneous control of accesses using different IoT application layer protocols.

As shown in Figure 3.8, we consider a system of interconnected subsystems using different IoT protocols. These standalone subsystems include a set of smart-devices communicating via the same instance of CoAP or MQTT protocol. All of these subsystems work standalone with a separate instance of UCS that is installed accordingly to the protocol each time, enhancing thus their security and continuous monitoring of the access. Thus, all the devices of every system are connected to the same instance of UCS by communicating with their own PEPs. For example, we can have the systems ($Ci,\ i \in \mathbb{N}_{>0}$) that use the CoAP protocol and the systems ($Mj,\ j \in \mathbb{N}_{>0}$) that use the MQTT protocol. For the sake of simplicity, in Figure 3.8 we present only a couple of subsystems for each protocol. Because of the different architecture of each IoT protocol, the PEP component of UCON has to be specifically adjusted to each protocol, whereas the UCS does not change but can be executed in different components according to the protocol used. Every system has their own *local* attributes. In the meantime, each system *(Ci/Mj)* for their local evaluation of access requests may need access to the values of *remote* attributes that belong to another subsystem *(Cx,Mx)*. This communication between different instances of UCS for sharing attribute values gives us the ability to control different protocols without the necessity of the sensors that provide the attributes to use the same protocol for communication. At this point, it is worth mentioning that the use and functionality of both protocols works as standard despite the addition of UCON.

3.2.2 Experimental Evaluation

In this section we introduce a use case for the proposed framework and present experiments in both simulated and real environments to help demonstrate the results.

3.2.2.1 Example of Use Case

For this use case we consider a scenario of a MQTT system (system M) and a CoAP system (system C) within a smart home. Both systems have a UCS installed (UCSm and UCSc respectively). Figure 3.9 describes the topology of our use case scenario.

In this scenario, a smart-vacuum-cleaner performs an access request to UCON in order to operate. The policy states that it is allowed to operate only if the power consumption and noise levels are within a certain threshold. Some appliances like a smart-TV, a smart-meter (that measures power consumption) and a smart-speaker together with the vacuum cleaner belong to the same system, communicating via MQTT protocol (System M). Furthermore, we consider a smart washing machine, a smart thermostat and a smart noise sensor that are also inside the same home but communicate via CoAP protocol and they belong to a CoAP System (System C). In order for the vacuum cleaner (*Subscriber*) to start cleaning, it must first connect to the *Broker* of system M to subscribe to the power consumption values which are gathered and stored by the smart meter (*Publisher*). The *Broker* receives this request and forwards it to the UCSm. UCSm realizes that in the policy regarding authorization of subscriptions contains a noise level attribute that cannot be retrieved locally by a PIP. Since this value is not local to the system M but belongs to the UCSc of System C, the UCSm contacts the UCSc and asks for access to the

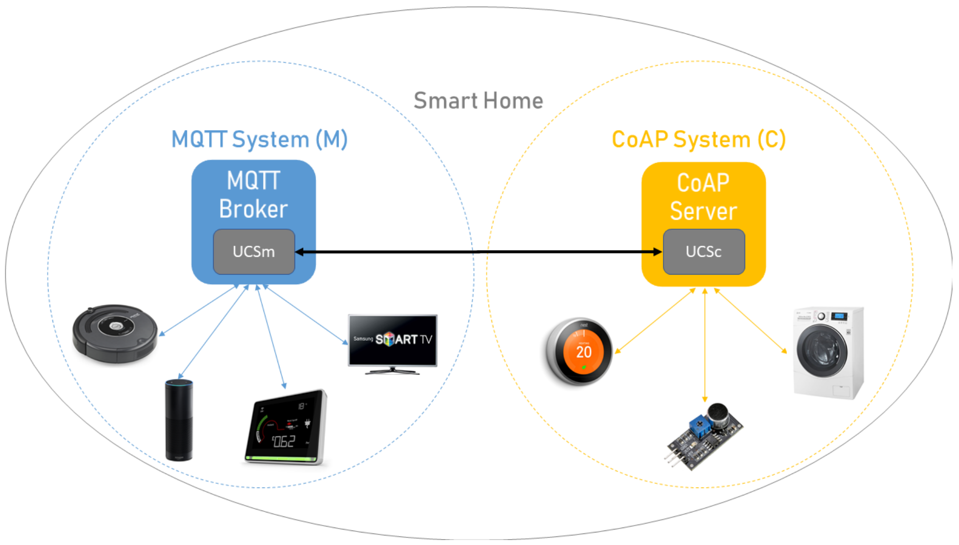


Figure 3.9: Use Case Diagram.

value of the noise level from the noise sensor.

After receiving the value from the UCSc, the UCSm evaluates the request and forwards the result to the *Broker*, which, in case of “Permit”, allows the vacuum cleaner to retrieve the data coming from the smart meter. In parallel, the UCSm continuously checks all the attributes and evaluates the policy. In the event that the value of the noise level or the power consumption rises above policy-defined threshold, the policy is violated and the access of the vacuum cleaner to the data of the smart meter is revoked, which also leads to the cancellation of the subscription from the MQTT system. Thus, without this information the vacuum cleaner will halt its task.

On the other hand, the opposite scenario is happening on System C. The smart washing machine (*Client* C1) wants to receive the data coming from the smart noise sensor (*Client* C2) of System C. In this instance the policy states that the smart washing machine is able to receive the data only if the power consumption is below a certain threshold. The CoAP *Server* of System C forwards this request to UCSc that checks the request conformance with the policy. However, the value of the power consumption belongs to UCSm, which means that the UCSc has to retrieve the attribute value remotely. Thus, UCSc communicates with UCSm, acquires this value, evaluates the request and forwards the result to the *Server* of System C. If the result is “Permit”, the *Client* C1 retrieves the data and operates normally. If at any given point the value of the power consumption or the noise level rises above the defined threshold, the policy is violated and the access of C1 to the data is revoked by the UCSc, which leads to the halt of the washing machine task.

3.2.2.2 Implementation

The UCS framework was implemented as a Java application. The software used to implement the *Broker* of the MQTT protocol was *Moquette*[6]. This *Broker* is based on the same programming language as the UCS framework which helped making sure that they could be integrated in the same device. Regarding MQTT Subscribers and Publishers we used off the shelf Python-based implementations. These ran as standard, without any modifications. Only the Moquette Broker was partially modified so as to host the PEP and call the UCS when invoking the *TryAccess, StartAccess, EndAccess* actions and wait the response from UCS. In the case of negative response, the Broker responses with *wrong user/password* message.

The software used for the CoAP system was *Californium*[7] which is also developed in Java like *Moquette*. *Californium* is a well-known implementation of CoAP and provides implementation of both *Clients* and *Servers*. Following the same pattern of the MQTT implementation, the Clients of CoAP ran without any modification whereas the *Server* was partially modified in order to host the PEP and the UCS. When a *Client* wants to observe some resource on a *Server*, the PEP calls the UCS invoking the same actions as previously in the case of the Moquette *Broker*. In the case of a negative response, the *Server* returns a message to the *Client* that it was not allowed to observe that resource. If, in the case of both protocols, there is a policy violation, *RevokeAccess* is invoked by the UCS to the PEP and on both cases the session is terminated and the *Subscriber* or the *Client*, depending on the protocol, is removed.

3.2.2.3 Testbed and Timing Evaluation

To demonstrate the viability of the proposed approach we examined the overhead of UCON in a real environment. The framework was tested in a virtual machine with Ubuntu 18.04 64-bit powered by an Intel i7-6700HQ using only 4 of its logical CPUs and 4GB of RAM. We considered a scenario containing one MQTT and one CoAP system, respectively. In this scenario we intercepted requests of each system

[6]https://github.com/andsel/moquette
[7]https://github.com/automote/Califorium

Table 3.2: Timings in Milliseconds (ms) for Both Protocols.

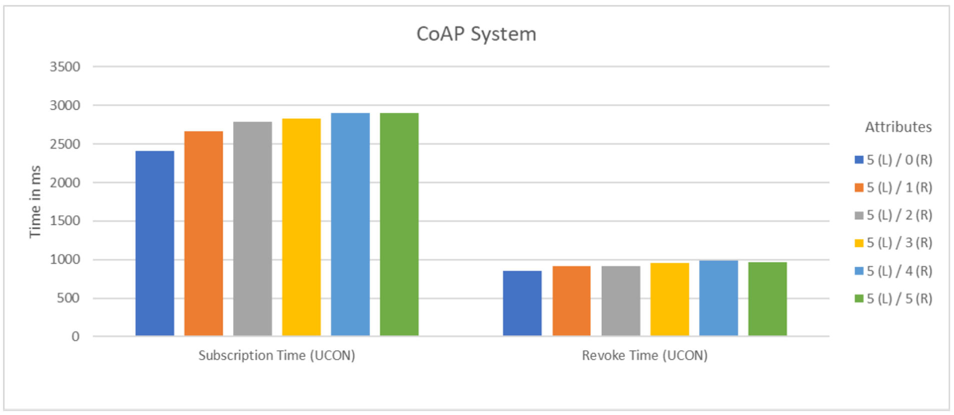
Protocol	**CoAP**						**MQTT**					
Time (ms) / No. Attrs (Local/Remote)	5/0	5/1	5/2	5/3	5/4	5/5	5/0	5/1	5/2	5/3	5/4	5/5
Subscription Time (UCON)	2407	2660	2783	2831	2901	2905	2455	2708	3065	2882	2884	2971
Subscription Time (Protocol)	1	1	1	2	2	2	3	4	4	3	3	3
Total Subscription Time	2408	2661	2785	2832	2903	2907	2458	2712	3070	2885	2887	2974
Revoke Time (UCON)	852	917	919	959	985	964	949	980	1155	1069	1058	1064
Unsubscribe Time (Protocol)	$\simeq 1$	$\simeq 1$	$\simeq 1$	$\simeq 1$	$\simeq 1$	$\simeq 1$	$\simeq 1$	$\simeq 1$	$\simeq 1$	$\simeq 1$	$\simeq 1$	$\simeq 1$
Total Revoke Time	853	918	920	960	986	966	949	980	1156	1070	1059	1065

considering a constant number of 5 local attributes and a variable number of remote attributes coming from the other system that varies from 0 to 5. In Table 3.3, we report the detailed timings for each one of the setups described above taking the median of 5 experiments per setup. Firstly, we present the scenario that the CoAP system has the local attributes and the MQTT has the remote attributes and vice versa.

The process of a *Client* to subscribe to a *Server* for acquiring data in CoAP is called observation and in MQTT is called subscription. Hence, whenever we want to refer to both operations simultaneously we will use the term subscription for both of them. We appose the subscription time only for UCON which consists of the summary of the times of the *TryAccess* and *StartAccess* actions. We also provide the subscription time that is needed only for the protocol and then we give the total sum of subscription times which is the sum of the previous two times. Considering the subscription time, we recognize that the time in each protocol is very low, especially for CoAP given that CoAP is more constrained as shown in Figure 3.10.

Also, we notice that there is some overhead caused by UCON. But, this overhead of UCON is happening only at the subscription time and there is no overhead in the functionality of each protocol when distributing information after granting permission. Moreover, the ongoing evaluation is happening in the UCS without causing any overhead to the protocol functionality. For unsubscribing (*RevokeAccess*) we consider again two different times for each protocol. The first time is the time of the execution of the *RevokeAccess* action for UCON and the second one is the time for the unsubscription of the components in the protocols (Subscriber and Client for MQTT and CoAP respectively) as shown in Figure 3.11.

Again the time of UCON is near one second which produces some overhead comparing to the approximately 1ms of the protocols, but yet again this is not crucial considering real IoT scenarios and it happens only in the revoke time and



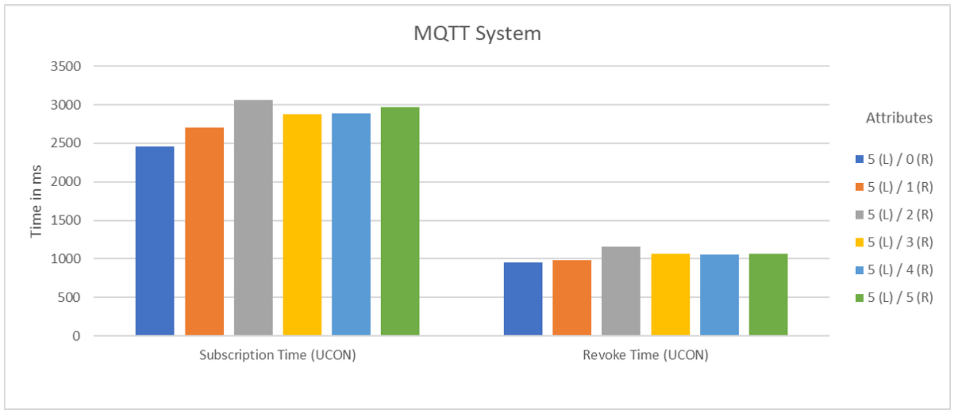
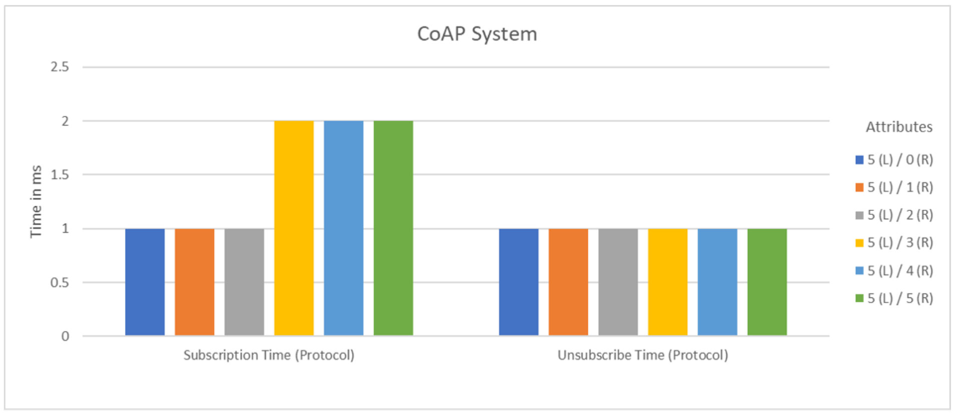


Figure 3.10: Subscription Timings for Both Protocols.

not during the sessions of the protocols. Finally, when there are not any remote attributes the timings are significantly lower, but as the number of remote attributes increases the timings of UCON also increase although not significantly.

3.3 Usage Control for Interoperable IoT Devices and Services

As shown above, in order to enhance security and privacy in IoT ecosystems, UCON has been a tool able to provide continuous access control inside IoT protocols in a seamless way[73]. Apart from mutable attributes, UCON introduces new decision factors besides authorizations; these are obligations and conditions [105]. The main feature of UCON is to handle Obligations are part of U-XACML policy



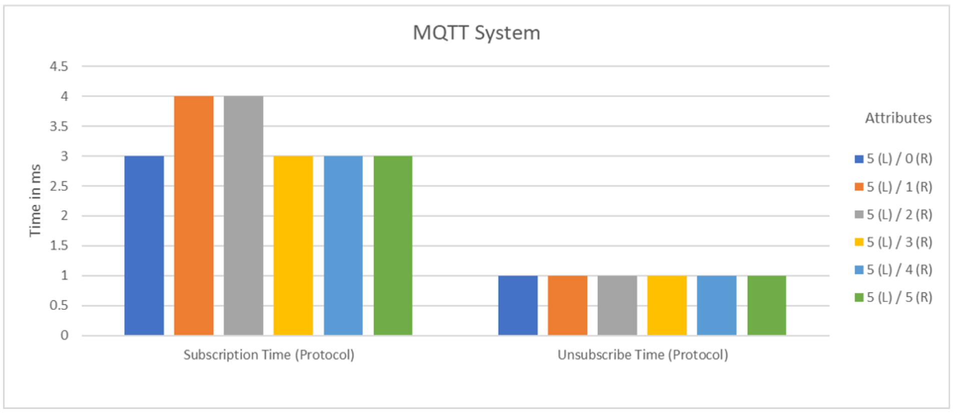


Figure 3.11: Revoke Timings for Both Protocols.

language that is used in UCON [26]. U-XACML has a very high expressiveness potential, hence fitting the flexibility requirements of complex environments, such as IoE ones. Since the third version of XACML, which is also the base of U-XACML there is a tag for describing obligations which are mandatory actions that have to be performed in conjunction with the policy enforcement [26]. Obligations are targeting either the PEP or the PIP. Currently, XACML does not provide a standard for the description of obligation semantic regarding the obligations targeting the PEP. While the intention is the one of non imposing constraints on format of the represented information, the direct consequence is that every obligation management engine has to be developed ad-hoc. Having a standard semantic for representation which is meaningful at least in an application macro-environment, such as IoE, can push developers and policy editors to use common expressions to represent and enforce obligations. In particular, it is possible to express obligations as commands for an inter-operable service or platform used by a multitude of IoT/IoE devices, such as IFTTT[8]. As stated in Chapter 2, IFTTT is a free web-based platform used to create applet chains of conditional statements in IoT settings. Expressing triggers through UCON obligations will allow the device receiving the policy evaluation decision to easily execute the obligations, without hard-coding the actual obligation interpretation, demanded to the specific UCON applet.

This section presents the application of IFTTT triggers to express and enforce UCON obligations that are targeting the PEP. We will describe a novel architecture where the standard UCON framework is combined with the IFTTT platform services. The proposed architecture is designed to be independent of the specific device implementation, specific application and transport level communication protocols. Thus, the proposed framework does not alter the XACML model and workflow, whereas we enhance its capabilities by proposing a standardized way of expressing obligations, reusing pre-existing components, being thus non intrusive.

In the following paragraphs, we will discuss the full operative workflow, detailing both UCON and IFTTT operation parts, proposing two relevant use cases and a set of performance experiments to demonstrate the viability of the proposed approach.

3.3.1 Enforcing Usage Control Obligations via IFTTT

This subsection describes the proposed framework for evaluating UCON policies and enforcing obligations via IFTTT including the workflow, the implementation and the process of including IFTTT *Triggers* in the standard XACML.

3.3.1.1 Architecture

Our goal is to create such an obligation format that the PEP can execute IFTTT *Applets* without the need of specific PEP per *Applet*. To this aim, we propose a framework that is implementing the PEP on a smart-device so as to perform access requests and receive the responses to and from the UCS respectively. If in the response of the UCS there is an obligation, the PEP has to extract and enforce it by performing the corresponding *Trigger* to an IFTTT *Applet*. The components of the proposed framework are shown in Figure 3.12. Firstly, the UCS has to be installed

[8] https://www.ifttt.com/

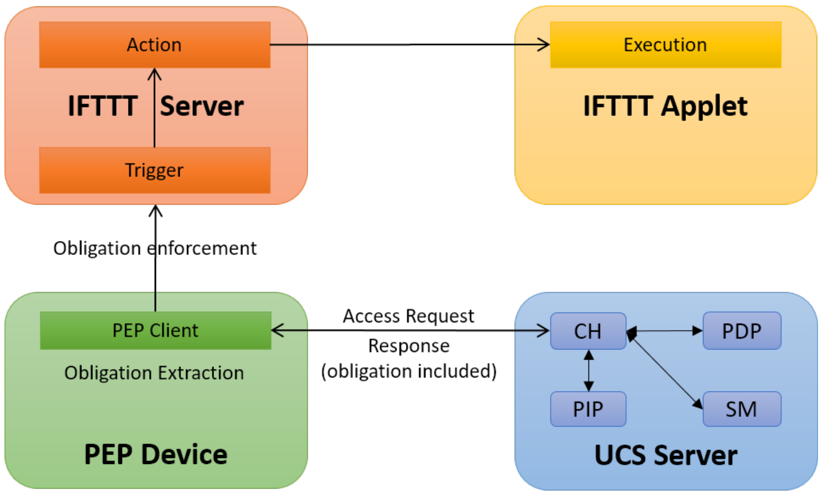


Figure 3.12: Logical Architecture of Usage Control in IFTTT.

on a smart-device which can be any appliance, computer, smartphone that is able to run an operating system capable of installing and running third party applications. Such a device can be a RaspberryPi or a smart-TV. Then, UCS has to communicate with the device that hosts the PEP. Other smart but not very powerful IoT devices, such as a smart temperature or light sensors, can be used from the various PIPs in order to acquire information about attribute values by the AMs and provide it to the UCS. The PEP can either reside in the same device as the UCS or in a separate device. The interpretation of the obligation coming from the UCS, is to the values that are necessary for the IFTTT *Trigger* and the enforcement is to perform the triggering of the *Applet*. The triggering happens via a web request from the PEP with the IFTTT by making a web request that enforces the obligation. This means that the PEP must have access to the Internet so as to communicate with the IFTTT servers. Then, the IFTTT server is responsible to execute the *Applet* when receiving the *Trigger* of the obligation. After that, the necessary information will be extracted from the IFTTT server and the *Action* service will be executed.

3.3.1.2 Operative Workflow

The complete workflow of our framework is presented in Figure 3.13. This figure describes the communication between the PEP and UCS from sending the request until the obligation enforcement and the triggering of the *Applet*. For better under-

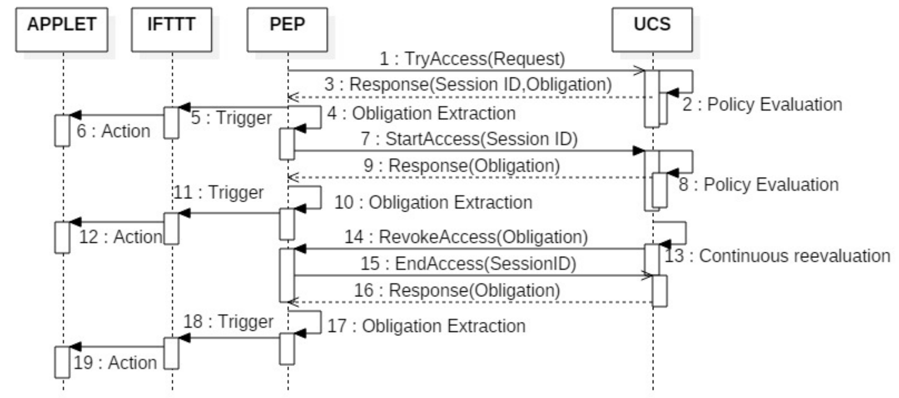

APPLET
IFTTT
PEP
UCS
1 : TryAccess(Request)
2 : Policy Evaluation
3 : Response(Session ID,Obligation)
4 : Obligation Extraction
5 : Trigger
6 : Action
7 : StartAccess(Session ID)
8 : Policy Evaluation
9 : Response(Obligation)
10 : Obligation Extraction
11 : Trigger
12 : Action
13 : Continuous reevaluation
14 : RevokeAccess(Obligation)
15 : EndAccess(SessionID)
16 : Response(Obligation)
17 : Obligation Extraction
18 : Trigger
19 : Action

Figure 3.13: Sequence Diagram of the Proposed Framework.

standing we will describe the workflow in two parts. These parts are i) the communication between the PEP and the UCS (tasks 1-3, 7-9, 13-16) and ii) obligation enforcement for by the PEP to IFTTT (tasks 4-6, 10-12, 17-19).

3.3.1.2.1 Communication between PEP and UCS

As shown in Figure 3.13, the PEP primarily initiates communication with the UCS by performing the *TryAccess* action for evaluation of the request (task 1). Then, the CH component of the UCS receives the request and the values of the attributes from the various PIPs . All the previous, are sent to the PDP for evaluation where the answer is *Permit* or *Deny* according to their compliance with the policy that arrives together with the request (task 2). If the answer is *Deny*, the PEP is informed about it (task 3). But, if the answer is *Permit*, the SM starts keeping a record of the session by assigning a unique ID to it and the PEP is informed that the access was initially granted and retrieves the session ID (task 3). In both cases, if there is an obligation in the response, the PEP has to extract and enforce it. The PEP, then, starts the actual usage of the resources by performing the *StartAccess* action to the UCS for the session with the specific ID (task 7). After another evaluation from the PDP (task 8), if the answer is *Permit* or *Deny* there may be an obligation in the response to the PEP, whereas the PEP is responsible for interpreting and enforcing it (task 9). For more information about the procedure inside UCS component, readers can refer to [76]. Moreover, while a session is in progress, there is a continuous re-evaluation of the session (task 13). In the case of policy violation, the UCS performs the *RevokeAccess* and the sends the appropriate message to the PEP (task 14). On the contrary, if the subject wants to terminate the session while it is on progress, the PEP has to inform the UCS about it by performing the *EndAccess* action (task 15) and receive the answer (task 16). Both in *RevokeAccess* or an *EndAccess* the message from UCS to the PEP may include, as previously, an obligation so after Task 14 or Tasks 15,16 the Tasks 17-19 of the obligation happen. We can see that obligations can be performed after every action of UCON, and they must include all the necessary information so that the PEP can trigger the *Applet*.

3.3.1.2.2 Communication between PEP and IFTTT

The PEP must not only communicate with the UCS, but also to extract the obligations and enforce them by triggering the IFTTT *Applet*. About obligation enforcement, firstly the IFTTT *Applet* has to be created in the IFTTT platform and the *Trigger* has to be a web request service. The *Applet* is executed by making the web request from the PEP to the IFTTT platform and, if the data received are correct, the platform performs the *Action*. The type of the *Action* depends on what the creator of the *Applet* selected and is not controlled by the UCS. The role of the PEP is to extract the information related to the obligation. For the tasks that may include obligations, the PEP must firstly extract the information included in the obligation (tasks 4,10,17). The next step is to enforce the obligation by performing the corresponding *Trigger* of the IFTTT *Applet* that has to run. To do so, the PEP has to create and send the appropriate web request to the IFTTT server (tasks 5,11,18). The correctness of the web requestis verified by the IFTTT server and it is not controlled by the PEP (tasks 6,12,19).

3.3.1.3 Obligation Standardization

According to the OASIS standard [99], obligation, as part of a policy for access control, is a XACML tag that describes when the obligation will be triggered. If it is triggered on a Permit or on a Deny it must be included in the appropriate policy rule. Furthermore, in U-XACML, the format of the obligation does not change compared to XACML[26]. But, in U-XACML, according to the time of execution, the corresponding obligation *Pre, OnGoing* must be included in the appropriate condition (Pre, Ongoing) respectively.

In our case, we consider the enforcement of obligations that have not attribute updates targeting the PIPs, but obligations that target the PEP and include the necessary data of an IFTTT *Trigger*. When an obligation targets the PEP, the payload that the PEP has to extract and enforce is included in the “ObligationId” as a string variable that has no specific type. In our framework, this string has the form of a JavaScript Object Notation (JSON) structure that includes the names and the values of the variables that are necessary to perform the IFTTT *Trigger*.

As an example, we consider a couple of obligations that happen a) after a successful set of *TryAccess* action followed by a successful *StartAccess* ($Oblig_1$) and b) after a *RevokeAcess* ($Oblig_2$). In Listing 3.1, we can see a simplified version of a policy written in U-XACML focusing on the obligation part. The first obligation example is an expression of ($Oblig_1$) and the second obligation part is an expression of ($Oblig_2$). Both are included inside the corresponding policy rules. We can see the *ObligationId* string, expressed in such a format of JSON structure so that it can be included in the U-XACML file without issues including the IFTTT payload.

The string of the obligation, which is in JSON format, includes the specific values for the *Trigger* of the IFTTT to happen. The *Trigger* is a web-request. For this action we selected the *Webhooks*[9] service when we created the *Applet* in the IFTTT platform. Webhooks service provides the ability for a web request to be the *Trigger* of the *Applet* but, generalizing, any other IFTTT service could be used supposing that the PEP is accordingly altered. The *Webhooks* services gives to each IFTTT acccount a unique *Key* that should be included in every request for identification purposes. Since the *Key* remains the same for all the different *Applets* of each user of the same *Webhooks* service account, the only way to distinguish each application between each other, is done via the unique *EventName* that every instance of the *Webhooks* service must have. *Webhooks* gives also the opportunity to include some payload variables in the web-request. The *Payload* can include values of attributes, plain text, or everything other information acquired by the UCS or the PEP. In total, in order for the obligations to include the IFTTT data for the *Trigger* as shown in Listing 3.1, they must include the *(Key, EventName, Payload)* values. Summarizing, for the two types of obligations mentioned above, the requirements and decision that has to be issued by the UCS to the PEP, are the following:

-TryAccess$\wedge$StartAccess$\rightarrow$Permit$\rightarrow Oblig_1$,

$Oblig_1$ = (Key1, EventName1, Payload1)

[9]https://www.ifttt.com/maker_webhooks

```
<Policy xmlns="urn:oasis:names:tc:xacml:3.0:core:schema:wd−17"
PolicyId="policyIfttt" RuleCombiningAlgId="..." Version="3.0">
<Description>Description</Description>
<Rule Effect="Permit" RuleId="rule−permit">
<Target>Target</Target>
<Condition DecisionTime="pre">Condition</Condition>
<Condition DecisionTime="ongoing">Condition
<ObligationExpressions>
<ObligationExpression FulfillOn="Permit"
ObligationId="{\"EventName\":\"ucon_oblig_enforc\",
\"Key\":\"bhOEWZ5qcbgMoa_w4−Nny_\",
\"Value1\":\"UCON_request_for_access_happened\",
\"Value2\":\"with_result\",
\"Value3\":\"PERMIT\"}"/>
</ObligationExpressions>
</Condition>
<Condition DecisionTime="post">Condition</Condition>
</Rule>
<Rule Effect="Deny" RuleId="rule−deny">
<Description>Description</Description>
<ObligationExpressions>
<ObligationExpression FulfillOn="Deny"
ObligationId="{\"EventName\":\"ucon_oblig_enforc\",
\"Key\":\"bhOEWZ5qcbgMoa_w4−Nny_\",
\"Value1\":\"UCON_REVOKE_happened\",
\"Value2\":\"because_the_result_was\",
\"Value3\":\"DENY\"}"/>
</ObligationExpressions>
</Rule>
</Policy>
```

Listing 3.1: Simplified Policy Example Including Obligations.

-RevokeAccess→Deny→$Oblig_2$,

$Oblig_2$ = (Key2, EventName2, Payload2)

Every time that the PEP executes correctly the web request, the *Applet* should run and users should see this in the IFTTT panel and also monitor that the *Action* happened. The execution can also be monitored the control panel of the IFTTT platform account related to the *Applet* that was executed.

3.3.2 Experimental Evaluation

In this section, we present two relevant use cases for application of the proposed framework. Furthermore, a set of experiments to evaluate the performance over-

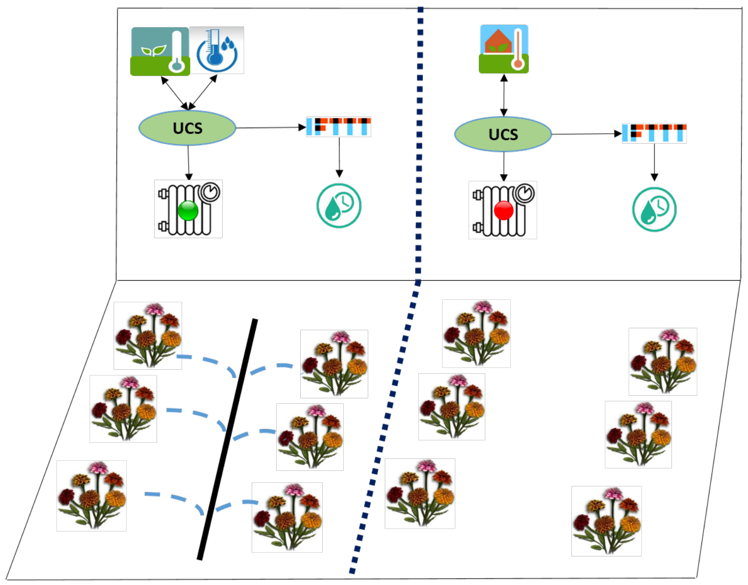


Figure 3.14: Example of Usage Control Obligation via IFTTT in a Smart-greenhouse Installation.

head introduced and demonstrate the viability of our approach is reported.

3.3.2.1 Examples of Use-Cases

The first example is advanced management through policy enforcement for remote urban farming in smart greenhouses. One of the motivations to consider this use case is that there are several pre-existing examples of IFTTT *Applets* designed[10] to manage watering and other smart devices in a greenhouse, which can be exploited to enforce UCON obligations. The representation of the scenario is shown in Figure 3.14.

The policy here for UCON is to monitor via smart-sensors various attributes of a greenhouse, e.g. humidity, temperature through the proposed framework and provide access to a smart-heating device to perform a scheduled heating of the plants. The obligations are targeting to the watering schedule of the smart-watering device that is responsible for watering the plants. The goal in this example is to maintain the temperature and humidity levels of the greenhouse to the desired values via controlling the smart-heating with UCS and change the schedule of the smart-

[10]https://ifttt.com/greeniq

watering device with IFTTT via the obligations coming from UCS. More in detail, the PEP is in the smart-heating device and asks for permission to operate on the smart-greenhouse. The PEP communicates with the UCS in order to request access to operate. The UCS monitors the values of the humidity, temperature and provides the answer. There are also other smart-sensors that can be used such as the availability of electricity/batteries etc. After a successful request, in the response sent by the UCS after the *StartAccess* action, there is an obligation including a schedule for the smart-watering device to operate less frequently. When the PEP receives this obligation, it has to enforce it by performing the web-request to the IFTTT platform containing the schedule for the smart-watering device. When the IFTTT executes the *Applet* the smart-watering device is operating on the defined schedule. In the meantime, if the weather is too hot or the smart-heating device is operating for too long, the UCS, during the continuous re-evaluation procedure, receives this information from the smart-temperature sensor. Thus, understands that there is a policy violation that may affect the plants inside the smart-greenhouse. So, the UCS performs the *RevokeAccess* to the PEP (smart-heating device) to stop operating. An obligation is also included that provides a schedule targeting the smart-watering device which has to operate more intensively so that the temperature decreases and the humidity level increases. This obligation is extracted and enforced by the PEP and is sent to the IFTTT platform. Summarizing, in the case of the obligation after the *StartAccess* the smart-heating device operates and the watering schedule is less intensive. In the case of the obligation after the *RevokeAccess* the smart-heating device stops operating and the watering schedule is more dense so as to lower the temperature and provide better conditions in the greenhouse.

Another example is smart management through policy enforcement for controlling a smart-office. The goal of this scenario is to continuously monitor the presence of the people inside an office and the air quality of the room while they are inside. The motivation in this example is the optimized use of the appliances that control the convenience of the people inside an office (such as air quality, temperature etc.). Frequently we face the situation that when there are several people in an office during meetings, the air quality is not optimal disturbing, thus, the meeting and make the people feel annoyed. The policy in this example is the control by the UCS of the access of the smart-heating device operation inside the room as scheduled and the operation of the ventilation system by the IFTTT platform through obligations. The control of the access is based on the continuous monitoring of attributes such as temperature and air quality provided by smart-devices

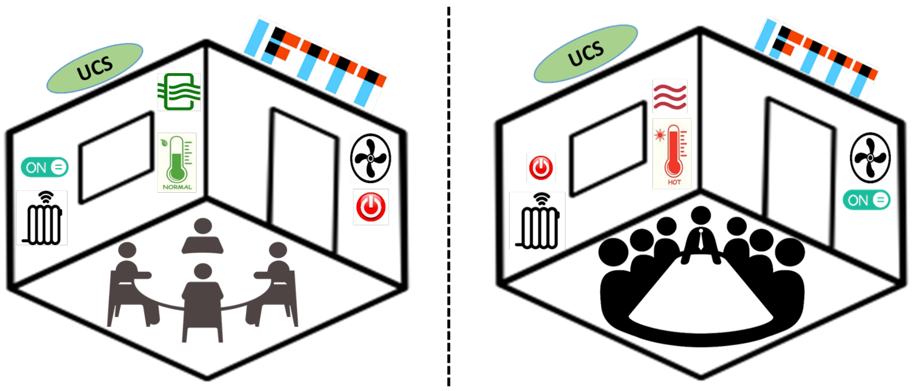
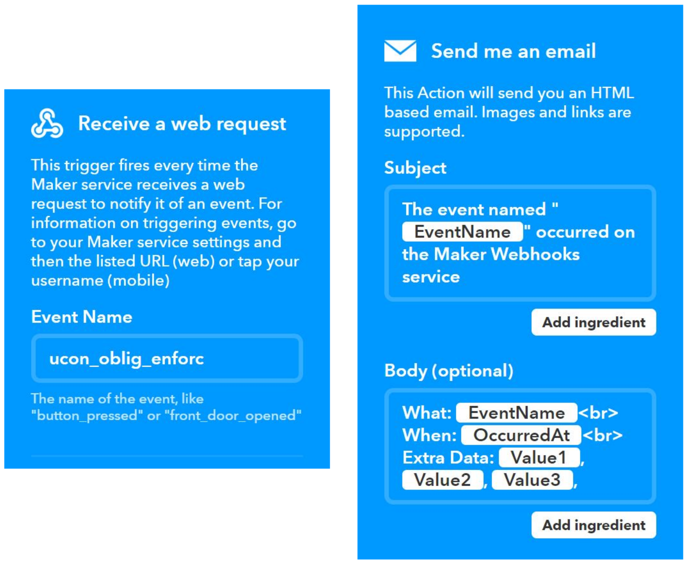


Figure 3.15: Example of Usage Control Obligation via IFTTT in a Smart-office Installation.

installed inside the office. The UCS can be in this case installed in a RaspberryPi that also provides sensors about air-quality[11]. The representation of this scenario is shown in Figure 3.15 where there are shown two cases of this scenario. On the left there is the case with a few people inside the office. The PEP is installed in the smart-heating device and request for access from the UCS to operate as scheduled. The UCS monitors the attributes of the temperature and air-quality and provides the answer. In the response of the UCS after the *StartAccess* there is an obligation for the IFTTT to force the stop of the ventilation system. On the right there is the case with too many people inside the office. In this case, there is a policy violation because the attributes indicate that the condition inside the office is not comfortable. Then, independently of the schedule of the smart-heating, UCS issues the *RevokeAccess* action to force the smart-heating device to stop. The obligation that comes together with the *RevokeAccess* and must be enforced, forces, through the execution of the IFTTT *Applet*, the smart-ventilation system to start operating. In this example, the first obligation comes with the response after a successful *StartAccess* and forces through the IFTTT the smart-ventilation system to stop. The second obligation comes with the *RevokeAccess* and forces through the IFTTT the smart-ventilation system to operate.

[11]http://bit.ly/2MlxI4i



Receive a web request

This trigger fires every time the Maker service receives a web request to notify it of an event. For information on triggering events, go to your Maker service settings and then the listed URL (web) or tap your username (mobile)

Event Name

ucon_oblig_enforc

The name of the event, like "button_pressed" or "front_door_opened"

Send me an email

This Action will send you an HTML based email. Images and links are supported.

Subject

The event named " EventName " occurred on the Maker Webhooks service

Add ingredient

Body (optional)

What: EventName
When: OccurredAt
Extra Data: Value1 , Value2 , Value3 ,

Add ingredient

Figure 3.16: Example of IFTTT Applet Structure.

3.3.2.2 Implementation

The UCS comes as a JAR or Web application ARchive (WAR) file, to be deployed on the device(s) intended to evaluate the access decisions. The *Trigger* was the *Webhooks* service, whereas the *Action* service was the “Send me an email” service of IFTTT platform that sends an email to the owner of the account of the *Applet*.

In Figure 3.16 there are shown the two services of the *Applet*. On the left, there is the *Trigger* part which provides a box for specifying the unique *EventName* of the *Webhooks* service. On the right part there is the *Action* service that includes the subject and the body of the email structure. On both the subject and the body of the email plain text can be combined with data coming from variables. These data may be included in the obligation (e.g. EventName, Value1-3) as they are described in Listing 3.1. The data may be also variables that the IFTTT platform provides, such as the “OccuredAt” in Figure 3.16 that gives the timestamp of the execution of the *Applet*. For validating that either the account of the *Applet* or the system that enforces obligations have not been compromised and there are no flooding attacks, users can compare the number, data and time of *Applet* executions in the IFTTT control panel with obligations ran in the PEP. For this *Applet* we consider two obligations that must be filled with details according the standardization of the previous section. The first one happens after a succesful *StartAccess* action (*Oblig*$_1$) and the second one happens after a *RevokeAccess* action (*Oblig*$_2$). The *Key* is obtained by the *Webhooks* service and the unique *EventName* is set up in the *Trigger* service as shown on the left part of Figure 3.16. There is the possibility of either executing the same instance of the *Webhooks* service with a different *Payload*, or creating two difference instances for each obligation (*Oblig*$_1$, *Oblig*$_2$). In the first case, which is the one used in this work, the *EventName* is the same and the *Payload* only changes, and in the second case both the *EventName* and the *Payload* change. In Listing 3.1 there are the examples of the obligations. It is worth noticing that the *Payload* includes three values. Whenever the UCS sends an obligation to the PEP, this obligation must include the *Key*, the *EventName* and the values. The link that the web-request has to be sent to has the following format in order for the *Trigger* to be successful.

- https://maker.ifttt.com/trigger/A/with/key/B

  A: $\leftarrow$ *EventName*

  B: $\leftarrow$ *Key*

This is necessary because in the case of creating multiple *Applets*, both the IFTTT and the PEP must distinguish them. The three values that are included in the obligation, explain in plain text what happened in UCON part and show the difference between *Oblig*$_1$ and *Oblig*$_2$. Hence, every time that the *Applet* is executed, the recipient of the email can distinguish which obligation has been enforced.

3.3.2.3 Testbed and Timing Evaluation

For evaluating the viability of our framework, we selected to create the following testbed. We used a virtual machine with Ubuntu 18.04 installed on a PC with constrained settings in terms of the enabled CPU cores and the amount of RAM used. In particular we used 2 cores of an i7-6700HQ CPU and 1GB of RAM. In this virtual machine we ran both the UCS and the PEP applications. When receiving an obligation, the PEP was forced to trigger an IFTTT *Applet* that was created for this scope. We have selected to study the performance of our framework by monitoring both the timings that UCON actions happen and the timings that our system needs to extract, create and send the web request of the obligation. However, since we cannot interfere with the time that actually the *Applet* is executed in the IFTTT platform or the synchronizing settings of the email recipient, we do not consider

Table 3.3: Timings in milliseconds (ms) over the attribute number.

Time (ms) / Attribute No.	1	5	10	15	20	25	30	35	40
Total TryAccess Time	312.6	387.8	380.6	358	380	393.6	433.4	476.2	477.8
Total StartAccess Time	79	94.2	76.8	98.2	90.6	127	106.2	131.4	169.8
Permit Enforcement Time	739.8	728.8	688.2	675.4	666.8	590.8	523.6	512.2	559.6
Total RevokeAccess Time	52	70.2	72.8	92.6	103.8	121.4	115.6	170.4	146.4
Revoke Enforcement Time	685.6	714.2	715.4	701.6	729.8	719	713.6	734.2	727

them in our timing evaluation. Although the execution of the *Applet* is related to the traffic of the IFTTT servers, we did not face in any experiment times larger than 1-2 minutes.

Our evaluation is based on the number of the attributes that our system has starting from a simple case with one attribute until a case with 40 attributes. We executed every experiment five times and took the average out of them, while increasing the number of the attributes to multiples of 5. The results of the execution are shown in Table 3.3. In this table we can see that the number of attributes increases the time that the UCS need to evaluate the *TryAccess* and *StartAccess* request. Both these timings if added together lead to the summary that from the time that the PEP sends the request via the *TryAccess* until the time it has back the response of the *StartAccess* by the UCS, the time varies from 400ms in the case of one attribute until 650ms seconds in the case of 40 attributes whereas it increases in a linear way. Nevertheless, we observe that the average time that the PEP needs to extract and enforce the obligation is independent of the number of attributes and equal to 450-700ms in every case. This is something that was expected since the attribute values are used only by the UCS for the re-evaluation but to summarize we can identify that in any case the overall time from the time that the PEP starts a request until the time that the obligation *Trigger* has happened is a bit more than one second.

In addition, we can see that the time for UCS to execute the *RevokeAccess* varies from 50ms to 150ms from 1 to 40 attributes respectively. The time that the PEP needs to extract end enforce the obligation in this case, remains similar to the one of the previous case and between 600-700 ms.

In Figure 3.17, we show the total timings for firstly handling a request and secondly handling a revoke. In the first case, which is marked with a dashed line, we report the time that from sending the *TryAccess* from the PEP to the UCS, until the time that the obligation is executed by the PEP to the IFTTT. We can identify that the total time does not change much when the attribute number is increasing

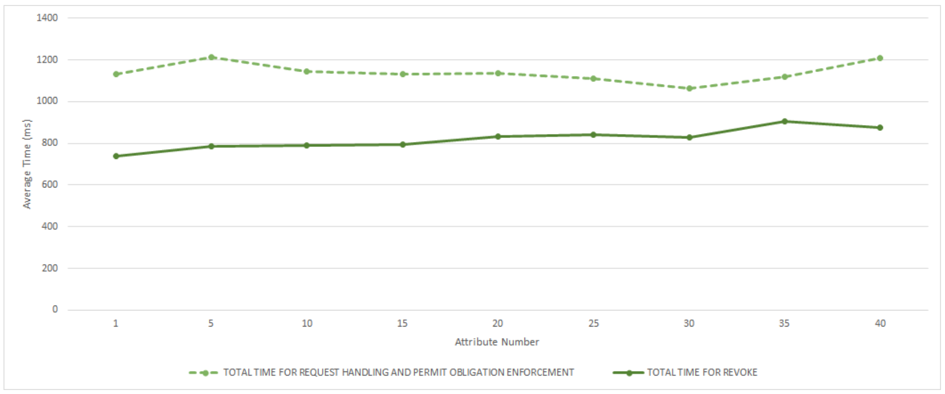


Figure 3.17: Timings for Handling a Request and a Revoke of Access.

which reveals a stability in the timings aroun 1100-1200ms. In the second case, which is marked with a continuous line, we report the time that the *RevokeAccess* is triggered until the time that the obligation is executed by the PEP. Again, we can see that the increased number of the attributes does not change significantly the timings.

Summarizing, obligations is an access and usage control tool which enables a capillary control that goes beyond evaluating the right to perform or not an action. However, in the XACML 3.0, specific semantic for standardized obligation representation and enforcementwhen the obligation is targeting the PEPs is not considered. This section shows a way to define an obligation semantic that is specific to IoT environments, using IFTTT *Triggers* to be enforced by PEPs that are directly connected to IFTTT *Applets*. We have reported two specific use cases which motivate our work and reported performance results to demonstrate the viability of the proposed approach, detailing how the proposed solution is not disruptive for the standard UCON workflow, enabling seamless integration [74].

3.4 Limitations of Usage Control in the Internet of Things

The previous sections of this chapter presented our work towards UCON and how it can be added in the protocols' architecture. Following each one of the implementations and architectural structure regarding each group of protocols (Publish/Subscribe or Client/Server) it is possible to create the architectural model of UCON in each one of the protocols aforementioned.

However, there are also limitations in large-scale heterogeneous systems such as IoT [73] [72] [116], where the number of attributes incorporated within the deployed security policies can grow exponentially, as is also the case with the complexity of those policies. This increases the demand for resources but also sets significant limitations in run-time efficiency and scalability of UCON.

Furthermore, in such systems, the number of the attributes which need to be evaluated grows continuously. Hence, the possibility of mistakes and conflicts during the policy development increases. In the case of an application environment that consists of several services, where there are many subjects that want access to these services and the corresponding resources, the time and computational cost needed for the initial evaluation or re-evaluation of the policies increase radically. This is due to the fact that each session is evaluated for a specific subject object pair, and re-evaluated continuously in the same manner.

The scalability issues that arise can be faced by limiting the number of attributes to be evaluated. This could happen via combining the attributes by extracting a certain policy for access to a set of services, which require the same constraints in order to provide access to resources. Moreover, for every group of services the amount of attributes for re-evaluation can be reduced by combining them into a role following the pattern of RBAC models. Each user role will be re-evaluated by UCON, in order to provide or not access to the resources of a specific group of services.

The proposed enhancements in the existing UCON model, arise by the combination of the benefits provided by both these two approaches, where attribute based aggregation is utilized both for subject roles and object groups. Consequently, these aggregated values are incorporated within the predefined security policies, reducing the required resources for policy evaluation and accordingly increasing the scalability potential of such deployments.

Moreover, we see that even upon adopting and using the distributed UCON framework for IoT, we find occasions that certain subsystems may not be aware of the overall policy satisfaction on the whole ecosystem that they belong to. A policy that might result to "Permit" decision locally, may result to a global policy violation of the ecosystem. This should be controlled by higher tiers in the ecosystem, so that there is an overall management that controls the overall behavior of it. Then, according to the available resources, the next in hierarchy holons may have a separate management inside their subsystem and so on and so forth. Thus, we introduce a hierarchical alternative of distributed UCON in order to build such policies, so that via forcing attribute updates, higher holons may interfere to lower in hierarchy holons and impose ceratin activities to the lower tiers.

Chapter 4

Hierarchical Usage Control

In this chapter we describe the hierarchical architectural model of UCON.

Since, IoE ecosystems are composed of smaller structures that control separate type of devices and they are isolated from each other, they can also be split into smaller independent structures for security reasons, in order to avoid being compromised. Sometimes, it is possible to create smaller systems, for increasing speed of connection between devices and limit information load. Finally, there can be split subsystems of a more general system due to limited connection coverage.

Hence, we understand the necessity of having a unique framework that can control the universal access on the resources over complex IoE SoS. This gives the ability not only to control the access over complex systems, but enforce policies independently from the protocol, the communication pattern or the hardware of the devices.

In this chapter, we propose the security enhancement of complex IoE structures, which we consider as SoS by providing a hierarchical version of UCON [76] based on distributed UCON [109].

Since our goal is to use our model on the IoE area, we express IoE ecosystems as SoS, hierarchically structured by IoT subsystems that may use different devices, communication protocols or patterns. UCON has been proven above to be adaptable in the area of IoT [85] and that it can work simultaneously with IoT protocols [116]. It has been found in the literature that, in IoT, the structure of the SoS can be hierarchical but there is no tool to enforce policies throughout the whole SoS [7] [117] [55]. With this work, we aim to provide dynamic policy enforcement towards more secure data distribution on complex SoS of IoT subsystems. Moreover, we enhance the interoperability between the various systems, by sharing attribute values via UCON. This work presents a hierarchical alternative of UCON targeting IoE SoS. We highlight that the presence of UCON does not impact how the various IoT systems work, which means it can be supported in various protocols as presented in our previous studies [73], [116]. To further enhance the viability our approach, we provide a relevant use-case scenario.

4.1 Hierarchical Systems

Hierarchical theory is a promising area of general systems theory [136]. This theory deals basically with the decomposition of a system into subsystems forming a hierarchical structure and is, therefore, on method of dealing with complexity. These subsystems or infimals are coordinated by a supremal in such a way as to obtain original system objectives. Hence, hierarchical theory is applicable to systems with a natural hierarchical structure or whose dimensionality is so high as to present computational difficulties. Thus, it would be particularly appropriate for use in public and societal systems problems.

Hierarchy theory, as well as empirical evidence, suggests that complexity often takes the form of modularity in structure and functionality [159]. Therefore, a hierarchical perspective can be essential to understanding complex ecological systems. But, how can such hierarchical approach help us with modeling spatially heterogeneous, nonlinear dynamic systems like landscapes, be they natural or human-dominated?

The theoretical basis for the spatially explicit hierarchical modeling approach is called Hierarchical Patch Dynamics Paradigm (HPDP), which emerges out of the integration between hierarchy theory and patch dynamics [159] [158]. Hierarchy theory emerged from a diversity of studies in various disciplines, including management science, economics, psychology, biology, ecology, and systems science [134], [133], [107], [88]. It has been significantly refined and expanded in the context of evolutionary biology and ecology by a series of books published in the past two decades [89], [123], [124], [2]. The concepts of "levels" in organization and "hierarchy" are not recent at all [156]. Much of the theory is only pertinent to nested hierarchies in which lower-level components are completely contained by the next higher level, although some general attributes are found in both nested and non-nested hierarchical systems [150].

According to hierarchy theory, complex systems have both a vertical structure that is composed of levels and a horizontal structure that is composed by holons.

A holon is a model-component acting as an autonomous system giving directions to "lower" components and the other side looking "up" and serving as a part of a "higher" holon[1]. Hierarchical levels are separated by different characteristic rates of processes (e.g. behavioral frequencies, relaxation time, cycle time or response time). Higher levels are characterized by slower and larger entities whereas lower levels by faster and smaller entities. Generally speaking, the relationship between two adjacent levels is asymmetric: the upper level exerts constraints (e.g. as boundary conditions) to the lower level, whereas the lower provides initiating conditions to the upper. On the other hand, the relationship between subsystem "holons" at each level is relatively symmetric in that they interact in both directions. The interactions among components within the same holon are more strongly and more frequently than those between holons.

Thus, it appears more than necessary to point out that hierarchy, as used in the scientific context, does not always refer to a system that is rigidly controlled by overwhelming top-down constraints and in which bottom-up effects generated by local interactions are insignificant. Certainly, hierarchy theory does not suggest this, either. As discussed earlier, hierarchy theory emphasizes both top-down and bottom-up perspectives. While dominance hierarchies do exist in natural, social, and engineered systems [101], the local dynamics of, and interactions among, components are fundamental to the very existence of any functioning hierarchies. Indeed, the relative importance or relationship between top-down constraints and bottom-up forces in determining system dynamics is a key to understanding most if not all complex systems. Neither does hierarchy theory imply inflexibility or a lack of diversity and creativity. On the contrary, an appropriate hierarchical, dynamic structure not only provides opportunities for diversity, flexibility, and creativity, but also for higher efficiency and stability that are difficult to obtain in non-hierarchical complex systems.

4.2 Hierarchical Usage Control Architectural Model

This section presents the architectural model of hierarchical UCON.

This model consists of several UCS that can exchange attribute values so that they can behave as a complete IoT SoS whereas, in the meanwhile, each subsystem can host its own SoS that it is the highest tier etc.

[1] https://www.holon.se/folke/kurs/Distans/Ekofys/Recirk/Eng/holarchy_en.shtml

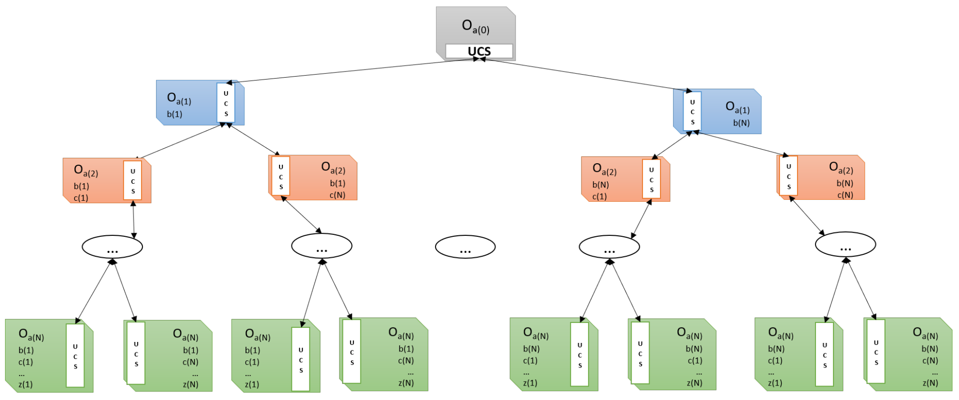


Figure 4.1: Diagram of the Hierarchical Usage Control Architectural Model.

4.2.1 Descending from the Top Holon to the Bottom Holon

As presented in Figure 4.1, we consider as a starting point, an system with a UCS that has an overall usage control role on the whole ecosystem (System $O_{a(0)}$).

This system apart from the overall view that it has, it can also get an overall view of every other subsystem of his SoS. These systems must declare that they are one level lower in hierarchy and, in the meantime to have individual identification inside their same level. So, since they are one tier lower than the central one they should be (System $O_{a(1)}$). Then we add their own identification and for the second tier in hierarchy and it becomes in total (System $O_{a(1)b(1)}$) for the first of this tier until the last one (N) of this tier (System $O_{a(1)b(N)}$). In accordance with the previous, as we move inside every holon, we have a new subsystem of individual holons that composite their own SoS.

Supposing that we reach the last tier holon in the hierarchy we reach to a point that each UCS is now controlling only access requests by devices and it is the lowest level. A UCS system of this level could be described as System $O_{a(W)b(X)c(Y)....z(Z)}$.

4.2.2 Ascending from the Bottom Holon to the Top Holon

The last description in the previous section indicates the whole path that someone should follow in order to find all the higher holons that control it. More specifically, a System $_{a(W)b(X)c(Y)....z(Z)}$ indicates the following path.

- z_Z: This is the "Z" subsystem of the "z" system.

- $\mathtt{c}_{\mathtt{Y}}$: This is the "Y" subsystem of the "c" system that is higher in hierarchy of the "z" system.
- $\mathtt{b}_{\mathtt{X}}$: This is the "X" subsystem of the "b" system that is higher from the "c" system.
- $\mathtt{a}_{\mathtt{W}}$: This is the "W" subsystem of the "a" system which belongs to the highest holon (System $\mathrm{O}_{\mathrm{a(0)}}$).

In fact, the value "W" indicates the overall hierarchy levels after the highest one which is System $\mathrm{O}_{\mathrm{a(0)}}$. So, if for example, "W" equals to 3, it means that there are 3 lower levels from the highest one.

In order to provide an in-depth monitoring of every holon inside an ecosystem, we need to provide information about each component of each subsystem to the higher, in hierarchy, systems. Then, each holon of the upper level will create an provide information about its status to the higher holons and so on and so forth. Thus we will reach the point, that the highest holon in the hierarchy of the system will be aware of the whole situation of the ecosystem.

This can be achieved by the expression of the status of every component to the higher holon via an attribute value. This value will be shared with the attributes of the various UCSs of the same subsystem that will collect them and provide a new attribute of its situation to the higher subsystem that it belongs until we reach the highest holon. Taking advantage of the distributed UCON that is presented in the previous chapter, each node can share an attribute to the other nodes about its situation. Hence, the higher holons can have a detailed view of the situation in every SoS that belongs to them.

Let us consider the scenario that an access request results to a "Permit" decision from the UCS of a lower holon for its own local policy. We may face a situation that this permitted access should be revoked because, although locally it is accepted, globally it causes a change to the whole ecosystem that is not permitted. So the highest holon may enforce an attribute update on its decision from the UCS that will be transfered by each holon until it reaches the node that the device should be revoked. Thus, we achieve less payload which in the area of IoE is crucial, and also we achieve interoperability between different UCSs.

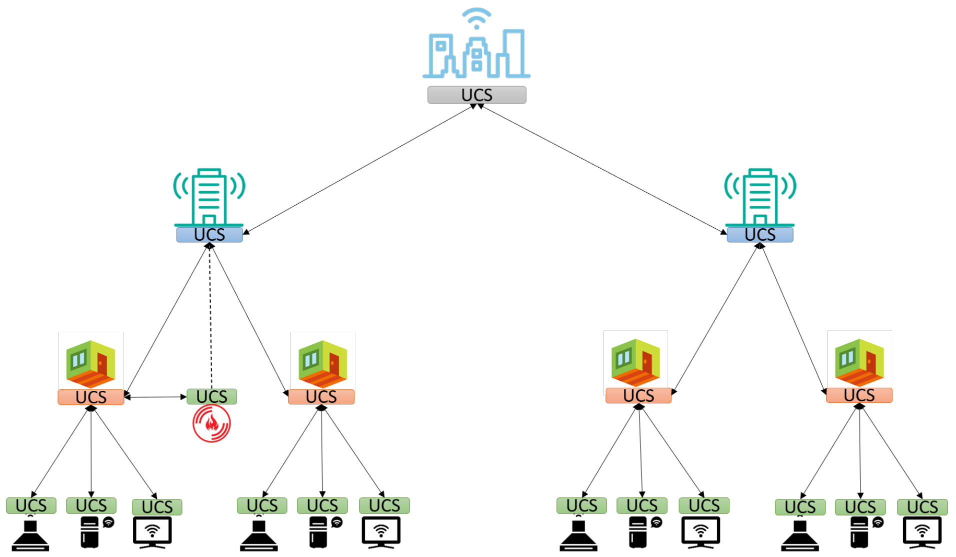


Figure 4.2: Use Case Example of the Hierarchical Usage Control Architectural Model.

4.3 Use Case Example of Hierarchical Usage Control

As an example we will present how our model fits on a smart-neighborhood scenario. We consider as our system a small part of a smart-city consisting of a set of subsystems that are the smart building. Now, every smart building can be a separate system consisting of several subsystems which in this case are the smart-appartments. Moreover, each apartment can be a separate system that consists of several subsystems that are the rooms of the apartment etc. This scenario is shown in Figure 4.2. On this way, we have created a top-tier SoS consisting of several other subsystems that can form their own SoS and so on and so forth.

In the meanwhile every separate SoS has its own local attributes, but the other entities of the same SoS can have access on them as remote attributes. Every UCS provides its local attributes to the central distributed database that can be accessed from the other tier nodes.

On this use-case we examine the hierarchy towards the overall eco-friendly behavior of the smart-neighborhood in terms of energy consumption. More in detail, the highest holon in hierarchy is the UCS of the overall neighborhood that collects the attributes of the energy consumption of all the smart-building and, in the meanwhile, has its own local attributes that control the energy consumption of the road lightening etc. The next hierarchy holon is the UCS of the smart-building that has its own SoS and its own local attributes that affect the power consumption of the public places in the building, (e.g. elevator, stair lights etc.). The next holon in hierarchy is the UCS of the smart-apartment that again can have a separate SoS whereas the next holon is the smart-room.

Every device in every UCS that belongs to the same SoS can have its own policy about power consumption. But it has to refer its overall power consumption in the higher hierarchically holon so that the latter can have an overall evaluation of the consumption of its SoS. This holon has to make available this value on the distributed database so that the higher than it can manage the higher tier power consumption etc.

As an example we consider a smart-vacuum cleaner that wants to operate. The request is evaluated by the UCS of the room that may result to “Permit”. As a result the attribute value of the overall consumption in the distributed database about this room is updated. The same happens to the attribute value of the overall consumption of the smart-building etc. On this way, we reduce the number of the attributes that have to be distributed amongst the several UCS reducing thus the payload.

Supposing that the central and highest in hierarchy holon is receiving an overall power consumption that should not be allowed. Then it has to find which is the least energy efficient next in hierarchy holon (in our case the smart-building) and update its max allowed energy consumption. Then again the smart-building has to update the max allowed energy consumption of the smart-apartment and when we reach the device level which is the lowest holon in the hierarchy, the UCS that the vacuum cleaner belongs to, has to revoke the access of the vacuum cleaner because, although it may locally could operate in first place, then due to the overall behavior of the whole ecosystem, it must stop operating.

More details on how the various UCSs communicate and exchange attributes were already presented on the previous chapter. The only difference with this model is that we consider the overall behavior attributes of the various SoS (e.g. the overall power consumption) and the ability to be controlled by the higher holon SoS via the tolerance attributes (e.g. max energy consumption allowed).

Chapter 5

Policy Management in Usage Control over the Internet of Everything

This section is about complex policy management and simplification in UCON. More specifically, it is about about handling the access over complex environments such the one of IoE where policies can be very complex. This happens via service and attribute grouping, leading us to move from ABAC to RBAC models.

5.1 Enhancing Usage Control for Performance: An Architecture for Systems of Systems

Modern interconnected systems of systems, require scalable and efficient security mechanisms, for controlling a very large number of access requests in a future with billions of heterogeneous devices connected to the Internet. The evaluation of access requests to certain pieces of information and services commonly relies on dedicated policies [93], which incorporate object, subject, and environmental attributes.

As presented earlier in this thesis, a limitation of access control is that the access request is only checked once, at the initiation, which highlights the lack of capabilities related to checking alterations on the values of attributes during a session which is a feature of UCON [76]. Yet, the examined environments carry inherent limitations in terms of both computational and communications capacity. Accordingly, corresponding optimizations must be implemented to the original UCON design, seeking to maintain operational efficiency at run-time, but also further security objectives related to resilience. Such optimizations must be initially integrated architecturally, and further enhanced within the components of the deployed policy based management systems.

In this part, we build on the results previously presented in this thesis, in order to mitigate the limitations of policy management in UCON. Namely, the current UCON architecture, requires the complete re-evaluation of access permissions per user-asset-session triplet, both at the initiation and at runtime. This, has been experimentally proven to be very time consuming especially when the number of users, assets, sessions and policy attributes increases [73],[72], [45]. Accordingly, we describe the developed architectural optimizations to UCON architecture, seeking to positively affect run time efficiency, scalability, and resilience against active attacks. In order to achieve that, a service group functionality is introduced to the existing model alongside with a dynamic role allocation subsystem. Thus, the right of access will be granted to a user, based on his allocated role for each group of services and not for one service at a time. The integrated optimizations improve the performance of the model, while increasing its resilience by allowing the mitigation of specific types of active attacks that are based on request flooding.

Architectures of this nature can be described in three abstraction levels, maintaining consistency and completeness. These levels are the (i) architectural model and components, (ii) protocol and interface, and (iii) implementation. So, we present and discuss the suggested architecture in all three levels in the next paragraph, highlighting the integrated optimizations to UCON and the corresponding affects.

5.1.1 Accompanying Information about Risk Aggregation

Large-scale applications create a challenging field in regard to access and usage control. The number of the attributes which need to be evaluated grows continuously and hence, the possibility of mistakes and conflicts during the policy development increases. Therefore, the architectural model that will be presented in this section considers the risk level that each attribute encapsulates, and aggregates these values for policy decisions. For example, if a subject wants to access a classified document and the policy takes into account the role of the subject, then it is possible to assign different level of risk to different roles, e.g the administrator of the system comes with a low level of risk while a new-hired employee with a high level of risk. Risk Aggregation methods are presented in this section for the purpose of better understanding. They were used for fulfilling the scope of this thesis but the research regarding risk aggregation is out of the scope of this thesis.

So, this model shall be a qualitative risk model for systems that make use of UCON, and its goal is to aggregate the risk values of the attributes into one single value, that will characterize the total risk of a given request. In order to achieve the aggregation, the model exploits the AHP [120]. Having the total risk value the security administrator has the possibility to define policies which are based only on this value or, as it will be explained later in this section, policies of any other granularity level. In order to make the functionality of the model clearer a set of definitions must be given [86]. For this work, we assume that there is no interdependence between the attribute for the aggregation procedure.

- *Full Policy*: A policy considering the attributes as they are extracted when acquiring the attribute values but not yet aggregated.
- *RA-Policy*: A risk aware policy is a policy which is written by considering the risk level of aggregated attributes. It has generally a smaller number of attributes with respect to the correspondent Full-Policy. Hence, it is easier to define and evaluate.
- *Initial Request*: A generated request enriched with the related attributes extracted.
- *Aggregated Request*: A request automatically computed by our framework, starting from an initial request, translating it to the aggregation level required by the current RA-Policy.

The framework is based on a reverse tree structure which is depicted in Figure 5.1 [86]. The total risk value, which was calculated by the aggregation of the attributes' risk values, forms the root of the tree. The upper levels consist of several blocks which represent groups of attributes that are related to each other. For example, a possible group could be the attributes related to environmental factors, such as the location or the time of the request. The leaves of the tree represent the attributes that participate in the Full Policy, whilst the Total Risk value is the one being considered by the RA-Policy.

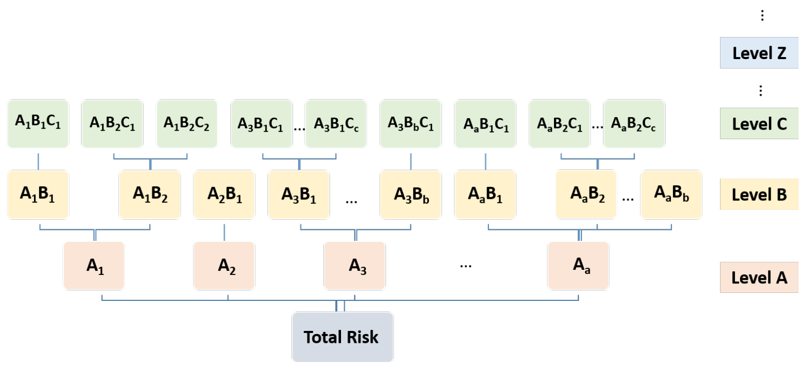


Figure 5.1: Total Risk Reverse Tree.

As stated above, the method used for the aggregation of the risk values of the attributes is the AHP. This method demands the definition of three elements: the *goal*, the *criteria* and the *alternatives*. Regarding the risk-aware model the *goal* is to characterize the total risk of the given request, the *criteria* are the various attributes and the *alternatives* are the possible risk levels (i.e. Low Risk, Medium Risk, High Risk). A set of comparison matrices is created, where an expert on the specific field of the usage control application environment, defines a level of preference among the criteria, stating by this way the relevance of each criterion with respect to the goal.

A comparison matrix is $N \times N$, where N is the number of the alternatives. Each element of the matrix takes a value in the interval [1,...,9] which defines the importance of an element in comparison with another one. Let us consider the previous example of accessing a classified document. Regarding the attribute of the role of the subject, it is reasonable to assume that the administrator of the system can be assigned with a lower level of risk than a new employee. The comparison matrix which represents this statement is shown in Table 5.1. The meaning of this matrix is that if the value of the role is the *administrator* then the value of *Low Risk* is considered to be 7 times more relevant than the *Medium* and 9 times more relevant than the *High* or *Unacceptable Risk*. On the contrary, if the value of the role is *new employee* then the *High Risk* alternative will be valued more than the others as shown in Table 5.2.

Finally, regarding the integration of the risk-aware framework to UCON, there is no need for any modification of the original model. The only requirement is the addition of a set of PIPs, which will acquire the risk values from the AHP blocks.

Table 5.1: Comparison Matrix of the Alternatives for the Administrator.

Administrator	**Low**	**Medium**	**High**	**Unacceptable**
Low	1	7	9	9
Medium	1/7	1	3	5
High	1/9	1/3	1	1
Unacceptable	1/9	1/5	1	1

Table 5.2: Comparison Matrix of the Alternatives for a new Employee.

New Employee	**Low**	**Medium**	**High**	**Unacceptable**
Low	1	1/4	1/9	1/9
Medium	4	1	1/9	1/9
High	9	9	1	1
Unacceptable	9	9	1	1

The proposed architecture is shown in Figure 5.2 [86], where the attributes are grouped into two sets. Each one of the sets will be aggregated using AHP and the results of these aggregations will be the input to a final AHP problem which will compute the single total risk value.

Having this architecture, it is also possible to define policies of different granularity levels, although it must be noted that excessive aggregation levels can affect the expressiveness of the policy, as discussed earlier [86]. For example, a policy can be defined by using only the single value of total risk, such as *"Subject can access object if the total risk of the request is at most medium"*, or combine this value with attributes either coming directly from the AMs or coming as outcome from any AHP block, such as *"Subject can access object if the total risk is at most medium and the time of the request is within the working hours"* or *"Subject can access object if the total risk is low and the risk of the environmental group of attributes is medium"*. Thus, this model is totally configurable and adjustable to the requirements of the application environment.

5.1.2 The Proposed Architecture

In this paragraph, we present the architecture for enhancing the UCON model, in two abstraction levels, namely: (i) the architectural model and its components, (ii) protocol and interface. The aim of this architectural enhancement is to improve

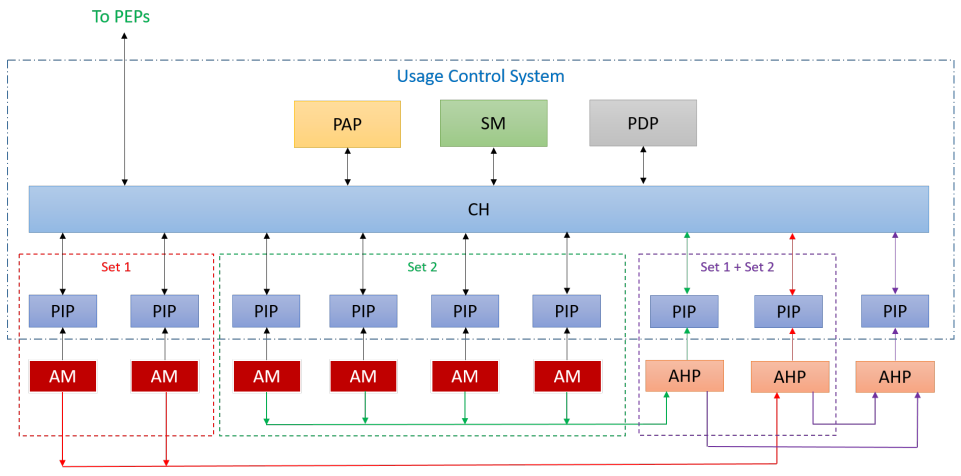


Figure 5.2: Risk Aware Usage Control Architecture.

the existing UCON model in terms of performance and efficiency. To this end, a service group functionality has been introduced in the current architecture. Alongside the dynamic user role allocation, this functionality gives the possibility for a faster access evaluation and response. Policy attributes are aggregated integrating criticality and risk metrics, allowing for the mapping of service groups but also for the allocation of distinct roles across these groups to every subject. Accordingly, the extraction of the service groups and the current user role (for each group) at run-time is achieved by the Group Handler (GH), and in accordance to the current attribute values. For example, considering that an application environment consists of ten services, the architecture for the enforcement of UCON policies proposed in [77], has to evaluate the subject's request for each one of them. On the contrary, this architecture, after grouping the services, will grant access to these groups in accordance to the predefined policies, and the dynamically allocated user roles, which are independently calculated for each group. Hence, if a user has access to a group, in accordance to his role for this group, and makes a request for a service belonging in this group the evaluation will be faster, improving the run-time efficiency.

5.1.2.1 The Architectural Model and its Components

The suggested architecture of the UCS remains unaltered to the one presented in Section 2.4.2 used in Chapter 3, with the exception of the introduction of a GH

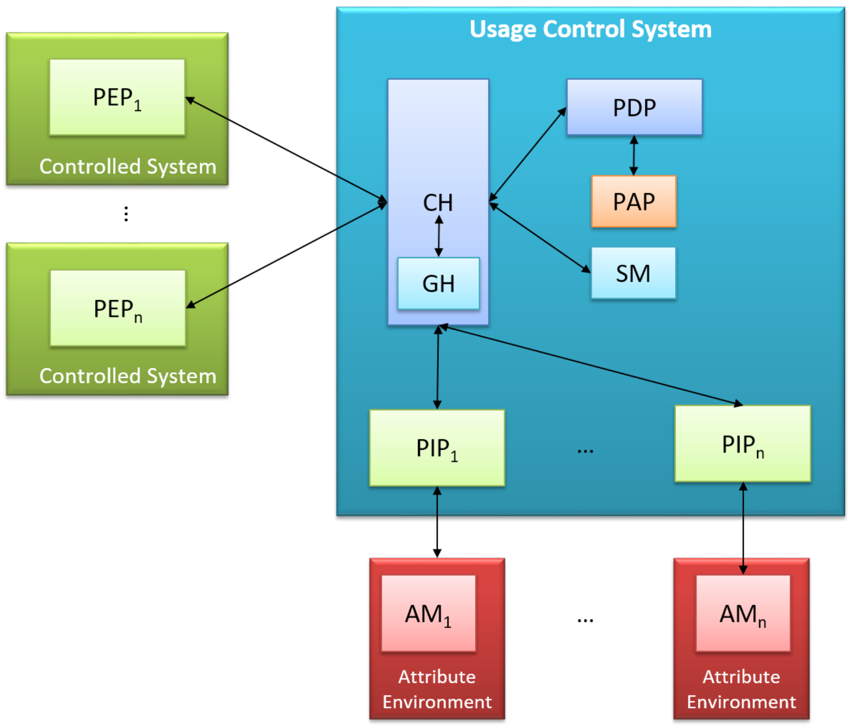


Figure 5.3: The Proposed Architectural Model.

as an internal sub-component of the CH, for the purpose of providing high-level compatibility with prior studies and implementations. The components of the architecture and their interconnections are in Figure 5.3. The actions used by the PEP to interact with the UCS in order to perform an access request, a start/end of usage of resources are the same as in the UCON model described earlier. The same applies for the actions used by the UCS to interact with the PEP in order to revoke access when needed.

The proposed architecture consists of the same components as in UCON with the addition of the GH. The discrete services provided by these components are:

1. **Existing components of UCS:** They have all the same architecture and they provide the same services as described in 2.4.
2. **CH-Context Handler:** The CH operates as the controller of the other components, and is responsible for the management and supervision of the session initiation and session re-evaluation processes.

- **GH-Group Handler:** This sub-component of the CH is responsible for the computation of both the service groups and subscriber roles that correspond to a session, in accordance with the risk aggregation model describer earlier, where the aggregated values of the corresponding attributes, are mapped into such roles and groups. In respect to the services, this computation can be done apriori and in the simplest form integrated as a Look up Table, although the GH can also incorporate the capacity for empirical environmental observation for dynamic service group management at run-time. As for the computation of the user roles, this is done at runtime in two occasions, the initiation of a session for a specific service group and the re-evaluation of access for a specific service group, but not on a per-session basis as in the original model.

5.1.2.2 Protocol and Interface

In this part we provide the sequence diagrams for the session initiation and re-evaluation processes, discussing the operations and providing corresponding examples. For the rest of this paragraph *Series of steps* refer to Figures 5.4 and 5.5, which provide the sequence diagrams during the initiation and operation phases in the following scenarios.

1. **Session establishment:** *Series of steps: 1-3-4-5:*
   In the initial steps of every session establishment request, the PEP translates the request into a *TryAccess* message towards the CH, which includes the unique identifier (Service ID) of the service that the subscriber requests access to. Consequently, the CH extracts the service group which corresponds to the given identifier, in accordance with the service grouping established during deployment, based on the risk aggregation method described earlier. Furthermore, the CH seeks to establish whether the subscriber has initiated similar request for this service, by querying the SM for active *TryAccess* entries. Provided that the SM replies negatively, therefore this request is not part of an active DoS attack, in step-3 the CH requests from the SM a notification about active sessions for the examined subscriber within the same service group. Given that no such sessions are identified, in step-4 the CH retrieves the required attributes from the PIP, extracts the subscriber's role that corresponds to the examined service group, and requests a policy evalu-

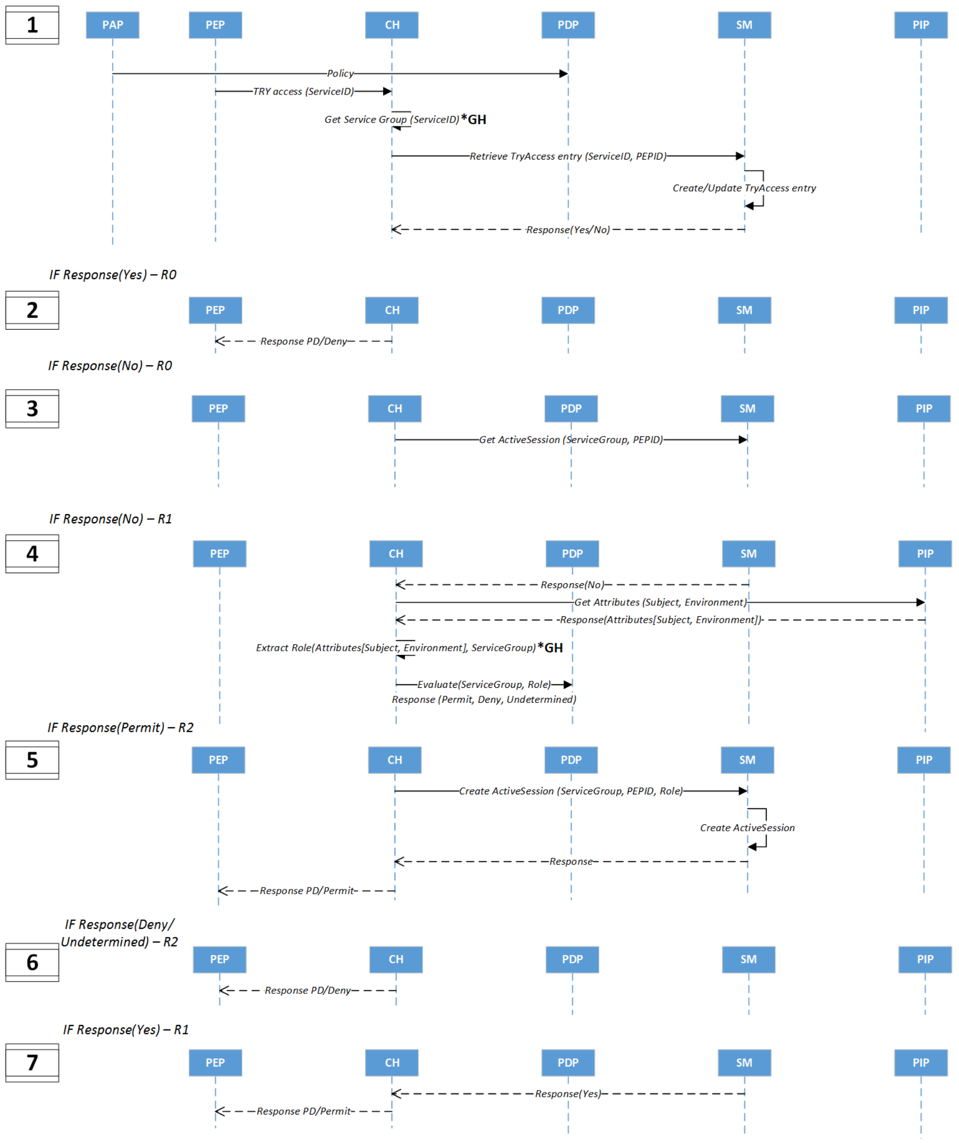

1
PAP
PEP
CH
PDP
SM
PIP
Policy
TRY access (ServiceID)
Get Service Group (ServiceID)*GH
Retrieve TryAccess entry (ServiceID, PEPID)
Create/Update TryAccess entry
Response(Yes/No)
IF Response(Yes) – R0
2
Response PD/Deny
IF Response(No) – R0
3
Get ActiveSession (ServiceGroup, PEPID)
IF Response(No) – R1
4
Response(No)
Get Attributes (Subject, Environment)
Response(Attributes[Subject, Environment])
Extract Role(Attributes[Subject, Environment], ServiceGroup)*GH
Evaluate(ServiceGroup, Role)
Response (Permit, Deny, Undetermined)
IF Response(Permit) – R2
5
Create ActiveSession (ServiceGroup, PEPID, Role)
Create ActiveSession
Response
Response PD/Permit
IF Response(Deny/ Undetermined) – R2
6
Response PD/Deny
IF Response(Yes) – R1
7
Response(Yes)
Response PD/Permit

Figure 5.4: Initiation Phase-Sequence Diagram.

ation from the PDP, based on the service group and extracted subscriber role. Further, in step-5, given that the permission is granted, the CH requests from the SM to initiate a corresponding session and send a permission notification to the dedicated PEP.

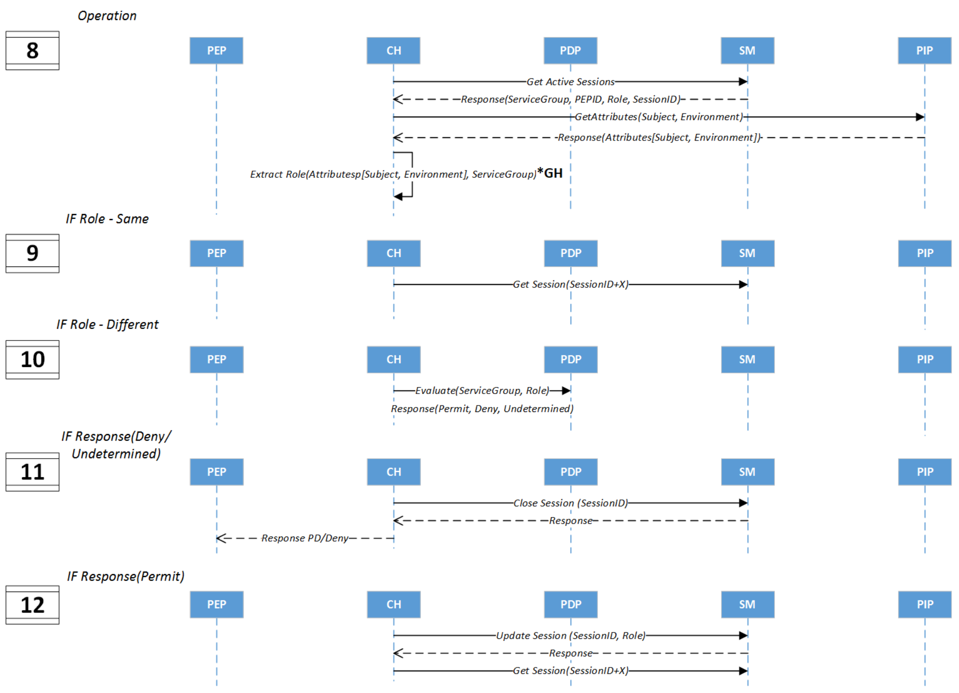

Operation
8
PEP
CH
PDP
SM
PIP
Get Active Sessions
Response(ServiceGroup, PEPID, Role, SessionID)
GetAttributes(Subject, Environment)
Response(Attributes[Subject, Environment])
Extract Role(Attributesp[Subject, Environment], ServiceGroup)*GH
IF Role - Same
9
Get Session(SessionID+X)
IF Role - Different
10
Evaluate(ServiceGroup, Role)
Response(Permit, Deny, Undetermined)
IF Response(Deny/ Undetermined)
11
Close Session (SessionID)
Response
Response PD/Deny
IF Response(Permit)
12
Update Session (SessionID, Role)
Response
Get Session(SessionID+X)

Figure 5.5: Operation Phase-Sequence Diagram.

2. **Denial of Service avoidance:** *Series of steps: 1-2*
   In this scenario the activities executed for step-1 are identical with those described for the *session establishment* scenario. Yet, given that the SM reports that *TryAccess* entries are still active for the same subscriber-service pair, (i.e. the time to live has not expired) this request is recognised as part of a DoS-Request-flooding attack, and the request is immediately denied in step-2. This improves the resilience of the UCON architecture, in comparison to the original UCON [77].

3. **Initial session denial:** *Series of steps 1-3-4-6:*
   In this scenario the activities executed for step-1, step-3, and step-4 are identical with those described for the *session establishment* scenario. Yet, given that the request is evaluated as "Deny" by the PDP, the PEP is notified accordingly by the CH. It must be noted that in this scenario, the *TryAccess* entry in the SM remains active for the corresponding time to live, leading to the previously described *Denial of Service avoidance* scenario, if an identical

request is delivered within this time to live.

4. **Request for the same service group:** *Series of steps 1-3-7/;*
In this scenario the activities executed for step-1 and step-3 are identical with those described for the *session establishment* scenario. Yet, given that the requesting subscriber has and active/permitted session for the examined service group, the CH immediately evaluates the request as "allow" notifying the corresponding PEP in step-7. This improves both the efficiency and scalability of the usage control architecture, in comparison to the original UCON.

5. **No attribute change:** *Series of steps 8-9:*
During the session re-evaluation phase, the CH requests the ActiveSessions entry from the SM. Accordingly, the CH requests from the SM the specific information for the first-in-queue session. Based on these information, and the timely values of the corresponding attributes from the PIP, the role of the subscriber is re-evaluated. Given that the role has not been changed, no further action is taken and the CH proceeds to the next-in-queue session, as described in step-9.

6. **Attribute change with permission:** *Series of steps 8-10-12:*
In this scenario the activities executed for step-8 are identical with those described for the *No attribute change* scenario. Given that a change occurred in the subscriber's role, the CH requests and new access evaluation from the PDP, in step-10, and updates the corresponding session entry of the SM in step-12, given that permission is granted by the PDP.

7. **Attribute change with denial:** *Series of steps 8-10-11:*
In this scenario the activities executed for step-8 and step-10 are identical with those described for the *Attribute change with permission* scenario. Yet, given that the policy evaluation result by the PDP is *Deny*, the session in the SM is closed and the corresponding PEP is notified, as described in step-11.

5.1.2.3 Test Case Scenario

The test case which has been utilized for the initial evaluation of the proposed architecture, and its comparison with the original UCON, is presented in figure 5.6. The test case refers to the cloud service deployment of a state owned airport

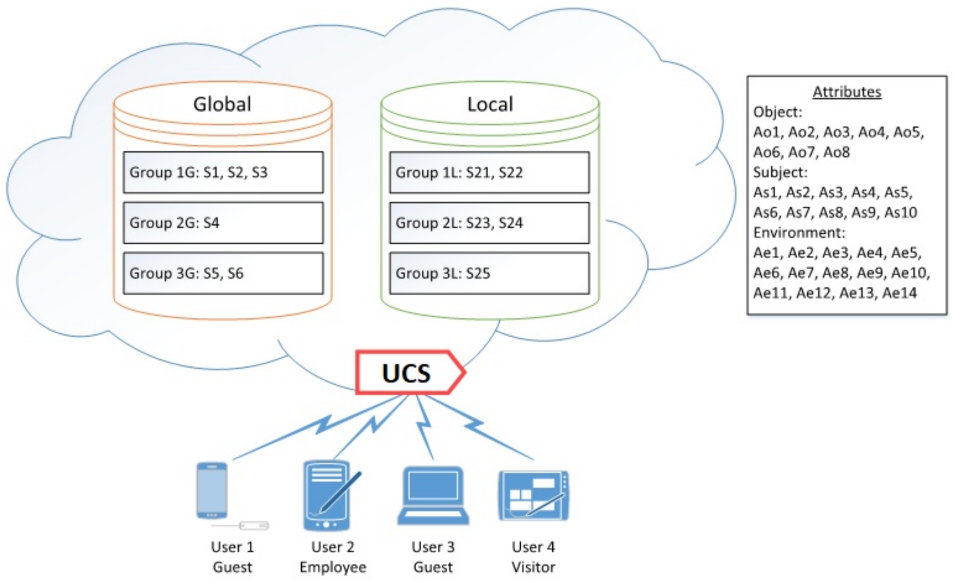


Figure 5.6: Exemplified Test Case Scenario.

operator, which is distinguished between a global deployment (with three groups of services, whose instances are available across all the managed airports) and a local deployment (with three groups of services in dedicated local instances per airport). A set of object, subject, and environmental attributes have been defined for the definition of the corresponding policies, while four distinct types (roles) of users have also been established.

Moreover, we present the results from one of the executed scenarios within this test case. In this, one of the operators' employees registers and seeks to obtain access for services S1, S2, and S3 of service group 1G. We executed the registration process for this scenario with the original UCON, and the Enhanced-UCON architecture presented in this section, for policies with 1, 5, 10, 15, 20, 25, 30, 35, and 40 attributes. Each test was conducted for ten repetitions, and the average times for the evaluation are presented in table 5.3 and figure 5.7. The table presents the elapsed time, in milliseconds, for each of the services, the total time, and the percentage of improvement. The test environment for this scenario was a virtual machine installing Ubuntu 16.04 64-bit, equipped with an Intel i7-6700HQ with 8 cores enabled, 8 GB DDR4 RAM.

The results highlight a significant improvement in terms of run-time efficiency, as both the number of micro-services and attributes (incorporated within the se-

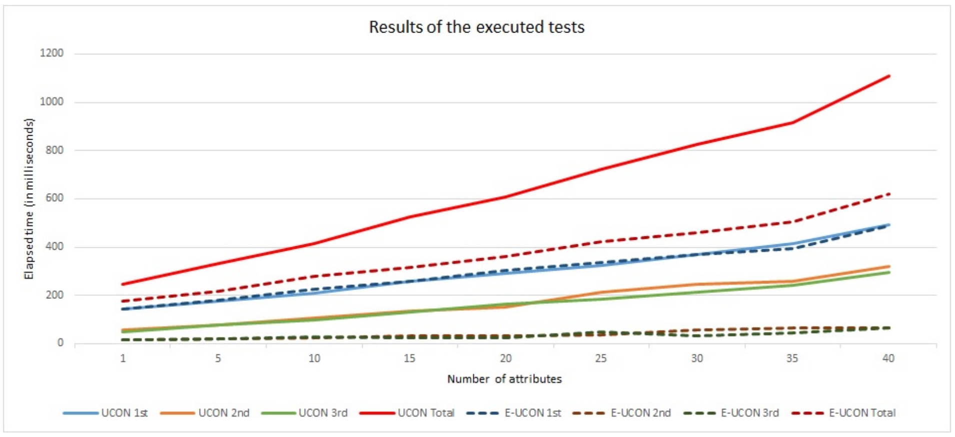


Figure 5.7: Results of the Executed Tests.

curity policy) increase. This improvement is not affected by the characteristics of the services towards which the access request is directed, as the services belong to the same group, for which the users role remain unaltered. A small degradation is noticeable for the initial service registration in low attribute policies, but this is quickly replaced by significant improvement of up to approximately 85%. In total the average performance, across all tests and repetitions, decreases by 1.346% for the first service, while for the second it improves by 77.195%, and for the third by 77.785%. The overall average improvement for three services, across all tests and repetitions, has been 39.154%.

5.1.2.4 Summary

In this part we presented an Enhanced-Usage CONtrol (E-UCON) architecture, where the standard functionality of the model is extended in order to support groups of services and users. This extension aims to improve the model in terms of performance and run-time efficiency, and to provide the scalability required from the application domain. The aforementioned improvements, result from the fact that the right of access will be assigned to user roles towards groups of services and not only in one service at a time, which reduces the evaluation time and the computational requirements. Furthermore, the proposed architecture improves the standard model in terms of security, as it gives the possibility of recognizing and preventing active attacks, such as specific types of Denial of Service based on request flooding. Finally, in this part we presented a method of simplifying the writing of security

Table 5.3: Results of the Executed Tests.

Number of attributes	1	5	10	15	20	25	30	35	40
Original UCON-times in milliseconds (ms)									
1st service	141.1	175.3	210.6	256.9	291.2	322.9	367.5	415	493.9
2nd service	56.6	76.9	105.6	134.8	153	211.8	245.7	256.3	318.8
3rd service	48.1	77.6	96.1	132.1	162.4	185.6	211.3	242	294.7
Total time	247.8	331.9	414	525.5	608.9	721.9	827.3	915.7	1110.4
Enhanced UCON-times in milliseconds (ms)									
1st service	142.7	179	223.4	259	303.1	335.2	368.3	394.8	487.6
2nd service	16.7	17.8	23.4	30.1	34	37.3	57.2	63.3	64.7
3rd service	14.4	19.4	28.2	24.2	24.1	48.9	33	44.3	65.5
Total time	175.4	216.9	277.1	315.3	362.5	422.2	461.1	504.2	618.9
Optimization percentage-%									
1st service	1.134	2.111	6.078	0.817	4.087	3.809	0.218	-4.867	-1.276
2nd service	-70.495	-76.853	-77.841	-77.671	-77.778	-82.389	-76.720	-75.302	-79.705
3rd service	-70.062	-75.000	-70.656	-81.681	-85.160	-73.653	-84.382	-81.694	-77.774
Total	-29.217	-34.649	-33.068	-40.000	-40.466	-41.515	-44.264	-44.938	-44.263

policies through the aggregation of the risk values related to individual attributes, is also integrated in the UCON model.

The experiments show that the aforementioned enhancements result in significant improvements in performance and evaluation time, especially in realistic deployments with multiple micro-services governed by complex or semi-complex policies.

5.2 From Attribute-Based Access Control (ABAC) to Role-Based Access Control (RBAC)

Constrained nodes [14] and constrained dynamic networks [33] are commonly deployed within a variety of application domains, due to operational constraints that arise from technical, physical, financial, regulatory and other limitations. IoT [54], emergency response [71], military operations [47] and remote ecosystem monitoring [57] are common areas where such systems and networks are frequently utilized, seeking to provide connectivity and access to services.

The constrained nature of such networks is characterized by low achievable throughput, high packet loss and packet loss variability, asymmetric link characteristics, and limits on reachability over time. Furthermore, due to their inherent dynamicity, they present high rate of change and unpredictability within the network topology graph, but also uncertainty regarding predominant traffic flow models, which are primarily bound to scenario specific parameters.

The continuous proliferation of connected devices, deployed services, and information (generated, stored, or in transit, also potentially classified) across the aforementioned application domains, along with the requirement for fine-grained policy-based security management, increases both the complexity and size of the deployed security policies [102]. Accordingly, the specification and enforcement of security policies is a challenging task, seeking under such limitations to provide suitable monitoring, control, and audit solutions, in order to oversee data flows across communication and control links but also access to services.

Providing security policy based management across these systems is a challenging task [48] [49] [50] [51] . A recommended terminology for policy-based management is provided in RFC-3198 [155], and the necessary concepts are utilized in the sections below. To this scope, this work describes a mechanism for the specification and enforcement of security policies which has been developed to provide effective fine-grained security policy-based management, still respecting the operational constraints of constrained distributed dynamic systems. The mechanism is utilized at the edge nodes deployed within the proximity network of such systems, while the computational burden is transferred to the policy specification and pre-deployment phases for the initialization, and to higher tier nodes during run time. We present it in two distinct phases, namely the initialization phase in preparation for policy deployment, followed by a comprehensive description of the processes involved in run-time operation. During run-time, the computational burden is transferred to high-tier nodes, while low-tier nodes apply risk aware policy enforcement based on a compromise solution.

5.2.1 Mechanism for Security Policy Based Management and Enforcement in the IoT

This mechanism is about the more efficient security policy-based management and enforcement within constrained distributed and dynamic systems, such as the IoT. The mechanism is utilized at the edge nodes deployed within the proximity network of such systems, while the computational burden is transferred to the policy specification and pre-deployment phases for the initialization, and to higher tier nodes which provide more computational power during run time.

The described operations have been developed in accordance to the following requirements:

- Policy rules capture multiple pre-conditions that are mutable in run-time.
- Each pre-condition is associated with a criticality and a freshness metric.
- The possible types and ranges of the pre-conditions are limited only by the specification language.

Accordingly, the mechanism operates under the following assumptions:

- High-tier nodes have the required resource capabilities for the enforcement of fine-grained security policy-based management, in accordance to the traditional Conditions $\rightarrow$ Rules $\rightarrow$ Capabilities paradigm.
- Low-tier nodes enforce security policy based management, based on the following assumptions:
  - Assets can be classified into a predefined number of classes in accordance to a delimited number of axioms and their specific data/object-property values. These axioms are defined in order to establish the attributes of those assets, allowing their classification in accordance to the similarities of the preconditions governing their specific action requirements.
  - In each subject we can assign a dynamic role per class in accordance to the values of its attributes. The role is inferred at high-tier nodes (who in run-time enforce dynamic ABAC per asset), and be delivered to low-tier nodes.
  - The utilized classification mechanism must provide membership functions for the classification of new assets, or the re-classification of assets with mutable attributes, without requiring continuous re-training for minor changes in addition to the periodic maintenance.
  - The utilized classification mechanism must allow for the forced assignment of a specific asset within a pre-determined class (even a singular class) based on a criticality metric, in order to support tailored security policy based management for specific assets.
  - The assigned subject role must be able to be re-inferred in run time, both as a periodic process and as an event-triggered process.

  - The security administrators must be allowed to precisely define the permissible margins for subject role assignment.

Accordingly, there is a risk integrated by the security administrators in order to support fine-grained security policy based management at the constrained edge nodes deployed at the proximity networks of constrained distributed dynamic systems, such as those described earlier but the risk technologies and research are out of the scope of this thesis. However for better clarification, risk is instantiated in the following processes:

- Grouping assets into classes.
- Assigning roles to subjects in accordance to the run-time values of mutable attributes, allowing for constrained elasticity to the exact permissible values.
- Defining security policies in a per-class/per-role basis.
- Specific design trying to avoid possible coherence issues.

The following subsections present the developed mechanism for the satisfaction of these requirements. We divided the presentation of the mechanism into two distinct phases, namely the *initialization phase* in preparation for policy deployment, followed by a comprehensive description of the processes involved in *run-time operation phase*.

5.2.1.1 Initialization Phase

In Figure 5.8 we depict the process which the proposed mechanism uses during the initialization phase, in order to classify the assets, establish permissible subject roles and define the necessary policies per role so as to regulate access. It consists of six steps enumerated below. In the defined process, steps 2, 5 and 6 are cyclical in order to ensure policy completeness, as described bellow in detail.

1. **Create Asset Taxonomy**

   This first step refers to create a taxonomy of the assets sufficiently detailed for the creation of the policies. This taxonomy can be formalized depending on the required granularity level. The inputs can come from a lot of parameters such as the following:

   - Asset identification process.

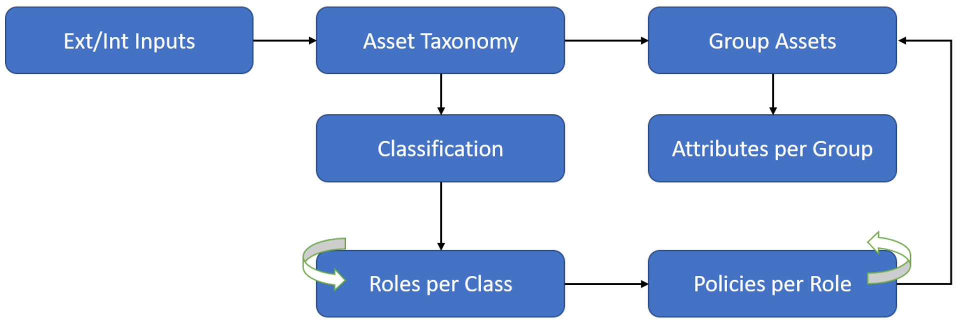


Figure 5.8: Process Diagram for the Initialization Phase.

- Earlier risk analysis results.
- Contemporary threats and attacks.
- Published vulnerabilities.
- Internal historical data.

2. **Select Asset Group**

   This is the first one of the cyclical group of steps in this initialization phase where there should be a policy separation between nodes. This separation depends not only on the inputs of step-1 but also on the conditions of the system on that moment. This process is cyclical because all the children nodes of a parent node have been selected and there have been specified suitable policies for them.

3. **List Critical Group Attributes**

   In this step, the responsible person for the security has to create each asset as an individual between the previously defined classed and select which are the critical attributes for it and give it a unique identifier. Furthermore, there must be a specific range of the value of every attribute associated also with a criticality level.

4. **Apply Multinomial Assets Classification**

   Multinomial classification is applied to the assets belonging to the defined groups, in order to identify classes based on the similarity of their attributes and criticality values. There are several methods for the asset classification [130] but their analysis is out of the scope of this thesis.

Table 5.4: List of Critical Group Attributes.

Attributes > Asset Identifier ∨	Y1	Y2	...	Yn
X1	F(X1,Y1)	F(X1,Y2)	...	F(X1,Yn)
X2	F(X2,Y1)	F(X2,Y2)	...	F(X2,Yn)
X3	F(X3,Y1)	F(X3,Y2)	...	F(X3,Yn)
...	...	...	...	...
Xk	F(Xk,Y1)	F(Xk,Y2)	...	F(Xk,Yn)

Table 5.5: Definition of Accepted Roles of each Subject per Class.

Class	Accepted Role
Class A	**Role-A1**, Role-A2, **Role-A3**
Class B	Role-B1, Role-B2, **Role-A3**
Class C	**Role-A1**, Role-C2, Role-C3

5. **Define Accepted Subject Roles for each Class**

   This steps defines all the roles that each class can accept, accompanied by the privileges and the actions that each subject accordingly has and can perform to the class. There is no specific number of the roles per class and it is highly depended on the application environment and its requirements. Moreover, there is no interdependency between roles and classes. The same role can be assigned to many classes without the prohibition of the policy to be identical. An example is shown in Table 5.5

6. **Policy Rules Definition per Role**

   In this step, we have to define the policy rules for every role and for every class that this role is included. As we stated in the previous steps, there can be different policy rules for the same role if it belongs in different classes. The rules are defined by taking into consideration all the attributes of each subject and also the environmental ones. After evaluating the values of this attributes we give a role to a subject for every asset on the class. We can consider an example for the classes and the roles, given in Table 5.5 using one environmental (EnvAtt) and one subject (SubAtt) attribute. Then we can assign the roles as follows:

   - If EnvAtt=E1, SubAtt=S1 then for Class A the subject gets the role "Role-A1".

- If EnvAtt=E2, SubAtt=S2 then for Class A the subject gets the role "Role-A2".

In this step we have to mention that we must include such mechanisms to avoid policy conflicts but this is out of the scope of this thesis.

5.2.1.2 Run-time Phase

This phase includes two steps. The first one (Step 7) is about extracting the role and the second one (Step 8) is about assigning the access rights for each role based on the policies about it per class.

7. **Role Extraction**

   In this step, we have the arrival of a request from a subject to access one of the assets of a class that was created in step 4. For this request and this subject we have to assign a role to it for this class. This role has to be transferred from the high-tier nodes to the lower ones and is calculated based on the run-time values of the attributes as we described before. This assignment can be handled as a Multi-Criteria Decision Making (MCDM) problem. In such a problem, we have to define a goal and we have to choose the most appropriate option among the alternatives that meet our scope. In our case, the goal is to extract the role and as alternatives all the accepted roles for every class as we presented them in step-6. There are several multi-criteria decision analysis methods but those methods are out of the scope of this thesis. The procedure for the extraction of the role is not happening only once, but also periodically so that we can ensure continuously the validity of the roles. The time period to re-check the validity of roles is application and criticality depended.

8. **Assignment of Access Rights**

   Finally, in this step after having extracted the the role, the higher-tier nodes will send the answer to a PDP in the low-tier nodes. This PDP will then match the assigned role to the subject that made the request and it will assign to it the appropriate rights. Then, if the same subject with the same role make another request for a different asset inside the same class there will not be a new communication between the low and the high-tier nodes unless there is a trigger for role re-evaluation such as expiration of attribute value freshness or policy violation. Such a case was presented in section 5.1 [45, 46].

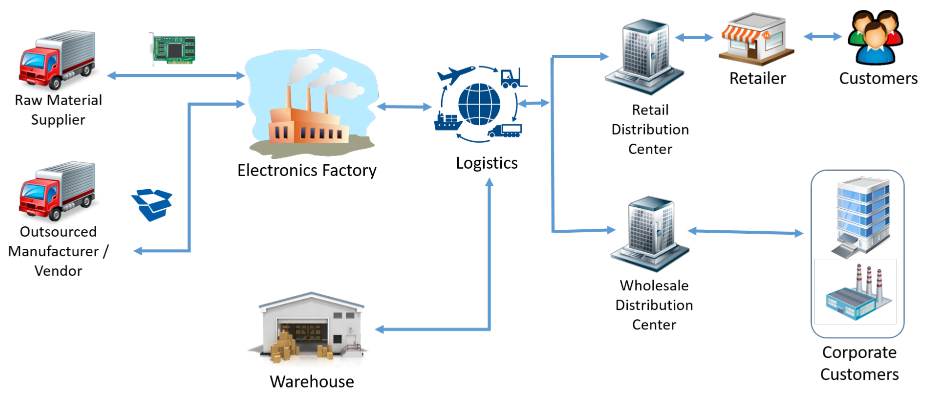


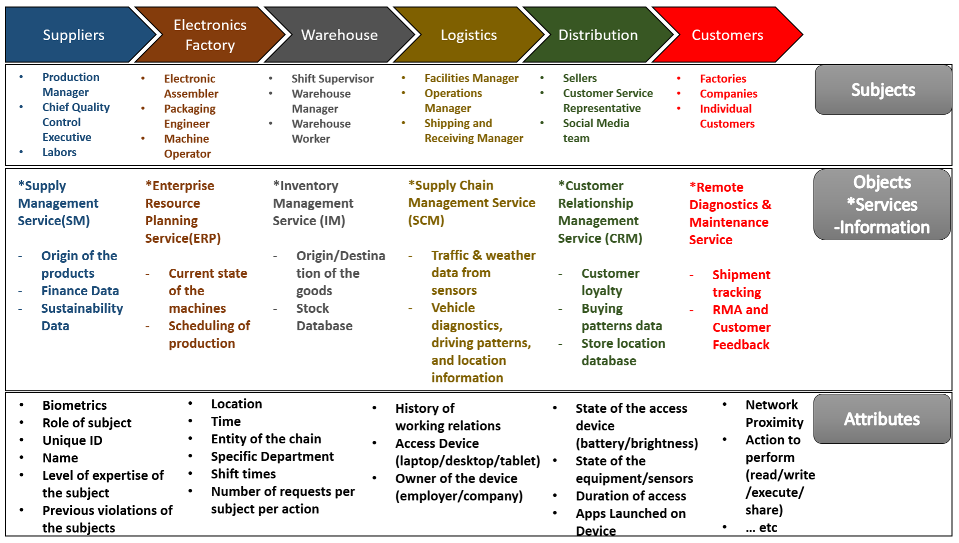
Figure 5.9: Use Case of a Digital Supply Chain of an Electronics Factory.

5.2.2 Use Case Example: A Digital Supply Chain Scenario

The revolution of Industry 4.0 is fundamentally reshaping the traditional manufacturing processes. The new manufacturing paradigm encompasses a full application of IoE with sensors placed in every step of the production, automated procedures, worldwide spread networks and advanced analytics of big data. The natural consequence of this change is to redefine also the well-known and standardized process of a supply chain to a more digitized environment. Digital Supply Chains (DSC) form an ecosystem, fully transparent across the entire organization, from the stage of the product development until the purchase and the delivery. One of the main characteristics of a DSC is that all data, regarding the management and operational processes, the quality control, the production planning etc. are available real-time to all the participating entities. In a recent study [41], 33% of the participating companies have already started to move towards a more digitized supply chain, whereas 72% of them plan to do so within the next five years.

An example of a DSC of an electronics factory is depicted in Figure 5.9. The key players of the chain are:

1. *Suppliers* who are responsible for providing to the factory all the necessary materials for the production of the final product.
2. *Factory* which materializes the production considering the demand of the market and the time schedule of the delivery.
3. *Warehouse* where the final product is stored, waiting for transportation.



Suppliers	Electronics Factory	Warehouse	Logistics	Distribution	Customers	
• Production Manager • Chief Quality Control Executive • Labors	• Electronic Assembler • Packaging Engineer • Machine Operator	• Shift Supervisor • Warehouse Manager • Warehouse Worker	• Facilities Manager • Operations Manager • Shipping and Receiving Manager	• Sellers • Customer Service Representative • Social Media team	• Factories • Companies • Individual Customers	Subjects
*Supply Management Service(SM) - Origin of the products - Finance Data - Sustainability Data	*Enterprise Resource Planning Service(ERP) - Current state of the machines - Scheduling of production	*Inventory Management Service (IM) - Origin/Destination of the goods - Stock Database	*Supply Chain Management Service (SCM) - Traffic & weather data from sensors - Vehicle diagnostics, driving patterns, and location information	*Customer Relationship Management Service (CRM) - Customer loyalty - Buying patterns data - Store location database	*Remote Diagnostics & Maintenance Service - Shipment tracking - RMA and Customer Feedback	Objects *Services -Information
• Biometrics • Role of subject • Unique ID • Name • Level of expertise of the subject • Previous violations of the subjects	• Location • Time • Entity of the chain • Specific Department • Shift times • Number of requests per subject per action	• History of working relations • Access Device (laptop/desktop/tablet) • Owner of the device (employer/company)	• State of the access device (battery/brightness) • State of the equipment/sensors • Duration of access • Apps Launched on Device	• Network Proximity • Action to perform (read/write /execute/ share) • … etc		Attributes

Figure 5.10: Digital Supply Chain - Test Case Details.

4. *A logistics company* which is responsible to transport in the best and safest way the products with respect to a specific time schedule.

5. *Distribution* of the products, which can be either to retailers or directly to other corporate customers such as factories or companies.

6. *End customers* who demand a continuous information flow about the status of their order from the time of the purchase until the final delivery.

The transparent ecosystem of a DSC presupposes the continuous use and exchange of both data and services. This process, if not secured, may lead to security breaches and information leakage to unauthorized parties, both within and outside the DSC. Figure 5.10 shows some examples of subjects, services and information per entity of the aforementioned supply chain. Furthermore, the last row of the table includes a number of exemplary attributes which can be monitored through the security policies, providing thus fine-grained security policy based management. The scenario which has been utilized for the evaluation of the proposed solution, refers to the DSC of Figure 5.9. In an environment like this, each entity manages certain resources locally and at the same time it has to share information with the other entities. In the following part we present a specific test case for the proposed mechanism within this scenario.

5.2.3 Test Case

The test case which has been utilized for the evaluation of the proposed model refers to a company belonging to the corporate customers of the supply chain shown in Figure 5.9. With respect to the mechanism given in section 5.2.1 the actions taken for each step are the following:

5.2.3.1 Initialization Phase

- **Step 1**: The information security administrator of the company, having conducted an asset identification process and also having analyzed previous incidents within the company, created the Asset Taxonomy. This taxonomy includes assets such as various Data (i.e Financial Documents, Customer Information, Customer Reviews etc.), Services (i.e Customer Relationship Manager, Remote Diagnostics and Maintenance etc.) and Sensors within the company's HQ (i.e temperature sensor, CCTVs, smoke detector etc.).

- **Steps 2, 3**: The selected asset group and the one used throughout the evaluation is the Invoices belonging to the Financial Data of the company. The asset group contains a total number of 150 invoices. Each one of the invoices is correlated with 25 attributes.

- **Step 4**: Having the assets and the values of their attributes this step initiates. In this step the assets are classified based on their similarities in distinct classes. If two assets have the same values for a number of specific attributes, they will be classified in the same group.

- **Step 5**: The security administrator considering the several operations which take place within the company, defines 3 roles per class and assigns to each one of them specific privileges. The roles and the corresponding privileges are given in Table 5.6.

- **Step 6**: Finally, the security administrator defined several sets of policies in order to regulate the access to the asset groups. These policies are based on the possible attributes that the subject can have, as well as on a number of environmental attributes. The evaluation of the policies result into the assignment of a role to a subject for the class he/she requested access to. For this test case, for Class A an example of the policies that have been defined are:

Table 5.6: Roles and Privileges per Class.

Class A	Manager Read, Modify, Share	Employee Read	Intern No access
Class B	Manager Read, Modify, Share	Sys Admin Read, Modify, Share	Guest No access
Class C	Sys Admin Read, Modify, Share	Employee Read, Modify	Intern No access
Class D	CEO Read, Modify, Share	Employee Read, Modify	Guest Read
Class E	IT Read, Modify, Share	Stuff Read	Intern Read

1. *Policy 1*: If the subject belongs to the Accounting and Finance Department, his identifier is within the 4893XXXX range, the time of the request is between 6-9am and the connection type is through an Ethernet cable then the role of the subject is *Manager*.
2. *Policy 2*: If the subject belongs to the Marketing Department, his identifier is within the 8849xxxx range, the time of the request is between 5-7pm and the connection type is through an Ethernet cable then the role of the subject is *Employee*.
3. *Policy 3*: If the subject belongs to the Production Department, his identifier is within the 5634XXXX range, the time of the request is between 12-2am and the connection type is wireless with WEP encryption then the role of the subject is *Intern*.

5.2.3.2 Run-time Phase

Steps 7, 8: In this step we must meet with the prerequisite to assign in all the attributes a numerical value that is inside the given scale for it so as to conform with the steps described in 5.2.1.2.

For the purposes of this test case we assume the following requests coming from two different subjects:

Subject A: belongs to the Marketing Department, his/her Identifier is 48934583, the time of the request is 10am and he/she is connected to the network via an Ethernet cable.

Subject B: belongs to the Accounting and Finance Department, his/her identifier is 56349812, the time of the request is 10am and he/she is connected to the network via Wi-Fi.

The final step of the procedure is to actually assign the roles to these subjects based on their attributes. As previously mentioned in the methodology, the security administrator, based on a number of factors (i.e. the importance of the asset), is able to define how rigorous the acceptable distance will be in order for a role to be assigned.

Out of all the possible roles that can be assigned, the first one that satisfies the requirements will be assigned to the subject, unless the security administrator set a different rule.

Consequently, the policy administration and enforcement subsystem (provisioned in a high-tier node) transfers these roles to a PDP in the lower-tier nodes. This component is responsible for storing the access rights per role, match the role to the predefined access rights, and assign them to the subject upon request.

Chapter 6

Conclusion

IoT pervasiness is everywhere these days and shall be higher in the next years. IoT security is a top concern as we anticipate more vulnerabilities and attacks from cyber-criminals, we face another issue that the devices that are sold are not secure by design and that patching of IoT devices has been proven to be difficult [147]. Moreover, attacks on IoT devices has been found as both simple and destructive [157].

IoE does not speak only about networked connection of "Things" like IoT, but it includes the *People* and how they interact in an interconnected ecosystem, the *Data* management and all the *Processes* that the previous create [63]. Thus, creating more secure mechanisms to continuous monitor and enforce policies in this area has crucial importance towards this new era. Furthermore, these mechanisms should be able to deal with the obstacles of heterogeneity, differences between devices and protocols for communication and type of operating system. In addition to the previous, these mechanisms should not add extra payload on the endpoint devices, and not interfere with the protocol specifications and functionalities. Finally, since IoE is a complex ecosystem consisting of several SoS, these mechanisms should be able to take into account behaviors that affect not only a local subsystem, but also the whole IoE ecosystem.

So, the goal of this thesis, as stated in the start, was to define and evaluate a generic policy enforcement architectural model that is based to a extended and adapted version of UCON framework, targeting to provide continuous access control and policy enforcement on the IoE. This model shall target to provide a hierarchical variation of UCON that will continuously enforce policies and control the access to the resources of complex IoE systems. This model shall be able to provide a holistic solution independently from the type, application protocol, policy attribute number or language between all components of an IoE environment. i.e. when attributes refer to the behavior of people, which increases policy complexity, UCON can cope with it seamlessly.

Firstly, we identified that current security mechanisms for IoT protocols are mainly focused on ensuring standard security properties such as message confidentiality and integrity, together with authentication. Moreover, to the best of our knowledge, we found out that up to now the efforts for policies enforcement, which would ensure much more flexible, expressive and effective properties, are still quite limited. We discovered also, that the correct component to integrate UCON with, was the communication protocol level, because this the area where access decisions are made and policies can be enforced regarding the violation of access rights. So our target was to enhance the security of the communication protocols by using UCON.

As a first step, we selected MQTT protocol on which to add UCON and enable, thus, the dynamic enforcement of UCON policies. We have presented a general methodology which allows to integrate UCON in a seamless way, without requiring protocol modifications. We included also an implementation has been presented, with performance evaluation to demonstrate the viability of the approach.

The next step was to check that our solution can be integrated not only in one protocol, but can be generic, independently of the protocol. Since IoT protocols follow two communication patterns (Publish/Subscribe and Client/Server) and MQTT is following the Publish/Subscribe one, we selected as next step to add UCON in CoAP protocol. For this reason, we used a distributed version of UCON which enables attribute sharing between UCSs.

We identified then, that we can take advantage of the attribute sharing ability of distributed UCON and that achieve the increase of security of both CoAP and MQTT protocols with dynamic policy enforcement of UCON policies together for access evaluation purposes between protocols. We have presented the general methodology of our approach that proves the ability of integrating UCON in a seamless way without modifying the protocols accompanied by an implementation of our framework that together with performance evaluation demonstrated the viability of our approach.

Realizing that IoE is an area that devices are meant to interact with each other, we tried to create a way that a specific action on one device, may cause an interaction to an other device. But, instead of changing the components of UCON every time for a new device, we took advantage of the obligations part of that come with the response from the UCS and force the PEPs to do specific actions.

Obligations is a powerful access and usage control tool which enables a capillary control which goes beyond evaluating the right to perform or not an action. However, in the XACML 3.0 specific semantic for standardized obligation representation and enforcement is not considered. Thus, in order not to change for every action the PEPs, we have proposed a way to define an obligation semantic that is specific to IoT environments, using IFTTT triggers to be enforced by PEPs that are directly connected to IFTTT *Applets*. We have also reported two specific use cases which motivate our work and reported performance results to demonstrate the viability of the proposed approach, detailing how the proposed solution is not disruptive for the standard UCON workflow, enabling seamless integration.

Since, in IoE there are structures that may be considered as complex ecosystems, composed of smaller structures that control separate type of devices, we realized the necessity of having a unique framework that can control the universal access on the resources over complex IoT SoS. This gives the ability not only to control the access over complex systems, but enforce policies independently from the protocol, the communication pattern or the hardware of the devices. For this, we presented an alternative of UCON SoS based on the distributed UCON, that is hierarchically structured. With this solution, we our goal was to cover the fact that access requests that could be permitted in a small local IoE subsystem, may be rejected by the higher in hierarchy holon due to policy violation in the whole system, or, more globally, to the whole IoE ecosystem. We also highlighted that the presence of UCON does not impact how the various IoT systems work.

Finally, we understood that an ABAC model like UCON inside IoE could generate large amounts of attributes for evaluation. Moreover, IoE environments are constraint and large amounts of attributes may create limitations in performance.

Thus, we presented an Enhanced-Usage CONtrol (E-UCON) architecture, where the standard functionality of the model is extended in order to support groups of services and users. This extension aims to improve the UCON model in terms of performance and run-time efficiency, but also to provide the scalability required from the application domain. The mentioned improvements, result from the fact that the right of access will be assigned to user roles towards groups of services and not only in one service at a time, which reduces the evaluation time and the computational requirements. Furthermore, the proposed architecture improves the standard model in terms of security, as it gives the possibility of recognizing and preventing active attacks, such as specific types of Denial of Service based on request flooding.

Extending our research towards the attribute and policy simplification, we presented a mechanism for the enforcement of risk-aware security policies in distributed dynamic and constrained environments by giving a comprehensive description of its functionality considering two phases, the initialization and the runtime. The first one includes the preparation for the policy deployment and processes such as the classification of the assets, the establishment of permissible subject roles and the definition of policies per role take place. The second phase includes the proposed mechanism that exploits a compensatory multi-criteria decision making algorithm, influenced by Technique for Order Preference by Similarity to Ideal Solution (TOPSIS), which by considering the actual values defined in the security policy, the run-time values of the subject at the time of the request or any upcoming reevaluation and the criticality of the assets, computes the optimal compromise solution and assigns a role to a subject.

Summarizing, our work has presented a way on which UCON can be applied in the area of IoE via being added in the level of communication protocols. We also included various modifications and alternatives of UCON so as to be adapted in a better way and achieve better performance. So we presented a way that we can create a SoS of UCSs that can be inserted in a IoE ecosystem, and enforce not only local, but also global decisions continuously, in a seamless way. This thesis covers the gap of continuous access control in the area of IoE by using UCON model. We have presented also some enhancements that try to make our solution more efficient especially regarding the resource limitations in IoE SoS.

However, we saw that there are limitations in our work, such as the risk introduced when grouping services and attributes. Another limitation is the necessity of continuous update and check in the shared database of distributed UCON framework. Also, we have to take into consideration the task of reducing the overhead and the time needed for UCON actions in comparison to the protocol (un)subscription times. Moreover, we shall consider enhancements in the actual UCON model itself, so that it can adapt better to newer techniques and environments.

6.1 Future Research Directions

As future work, in the area of IoT protocols, we plan to create evaluation scenarios in real applications so as to define and enforce policies in a real applicative setting. Furthermore, we intend to make UCON framework a default feature of the IoT protocols. The main goal is to try and insert UCON in the standards of the protocols and also try to mediate the related limitations described in the previous paragraph.

Regarding obligation standardization and IFTTT, we plan to perform an implementation on a larger testbed with a real parental control scenario with several coexistent IFTTT *Applets* and a wider number of attributes. Furthermore, we plan to extend the standardization effort, giving a formal definition of the grammar to be used for defining IFTTT obligations.

As for the simplification of policies, attribute and service grouping, we intent to perform further enhancements related to (i) credential management, (ii) trust, and (iii) task delegation that will cover the aforementioned limitations and will be integrated and tested within E-UCON. This work includes the developing of an extended and heterogeneous test-bed for experimentation, which will be utilized in order to evaluate the performance of the proposed enhancements in different and more demanding use cases. Another goal is to create a model for transforming ABAC models into RBAC ones and simplify the evaluation procedure and the policy complexity.

Acronyms

ABAC Attribute-Based Access Control. 5, 6, 11, 18, 32, 33, 36, 46, 95, 110, 122, 124

Ack acknowledgement. 23

AES Advanced Encryption Standard. 51

AHP Analytic Hierarchy Process. 54, 97–99

AM Attribute Manager. 40, 63, 64, 74, 99

AMQP Advanced Message Queuing Protocol. 19–21, 25, 26, 29, 31

API Application Programming Interface. 47

CAAC Context Aware Access Control. 32, 36, 46, 47

CH Context Handler. 40, 41, 57, 58, 63, 64, 75, 101–105

CI Computational Intelligence. 53

CoAP Constrained Application Protocol. 3, 8, 9, 13, 19–23, 30, 31, 49, 51, 55, 62–70, 121

CON Confirmable. 23

CPS Cyber-Physical System. 2

CPU Central Processing Unit. 31

DDS Data Distribution Service. 21

DTLS Datagram Transport Layer Security. 3, 20, 21, 23, 31, 51

ECC Elliptic Curve Cryptography. 51

GH Group Handler. 100–102

HI Hybrid Intelligence. 53

HPDP Hierarchical Patch Dynamics Paradigm. 89

HTML Hypertext Markup Language. 27, 28

HTTP Hypertext Transfer Protocol. 3, 4, 19, 21, 22, 26, 27, 29–31

IETF Internet Engineering Task Force. 3, 21, 26–28, 51

IFTTT If This Then That. 6, 9, 10, 18, 19, 47–49, 52, 55, 73–85, 122, 124

IoE Internet of Everything. 1, 2, 4–7, 9–12, 14, 16–18, 30, 34, 36, 45, 47, 49, 52, 72, 73, 88, 89, 92, 95, 115, 120–123

IoT Internet of Things. 1–4, 6–9, 12, 13, 16–21, 23, 25, 28, 30, 31, 43, 45–53, 55, 65, 66, 70, 71, 73, 74, 85, 86, 88–90, 108, 109, 120–122, 124

IP Internet Protocol. 23, 24

ISO International Organization for Standardization. 25

JAR Java ARchive. 59, 82

JSON JavaScript Object Notation. 77

M2M Machine-to-Machine. 3, 16, 21

MCDA Multi-Criteria Decision Analysis. 54

MOA Message Oriented Approach. 30

MOM Message-Oriented Middleware. 19, 20, 25, 27

MQTT Message Queue Telemetry Transport. 3, 4, 8, 9, 13, 19–21, 24–26, 29–31, 49, 50, 55–57, 59, 61, 65–70, 121

NON Non-Confirmable. 23

P2P Peer-to-Peer. 8, 45, 51

PAP Policy Administration Point. 41

PDP Policy Decision Point. 40, 41, 53, 58, 61, 64, 75, 103–105, 114, 119

PEP Policy Enforcement Point. 9, 40, 41, 46, 53, 56–59, 63, 64, 66, 69, 72–78, 80–85, 101–105, 122

PII Personal Identifiable Information. 2

PIP Policy Information Point. 40, 41, 56–58, 63, 64, 67, 72, 74, 75, 77, 98, 102, 105

QoS Quality of Service. 20, 21, 23–26, 31, 62

RAdAC Risk-Adaptable Access Control. 32

RAM Random Access Memory. 31, 51, 60

RBAC Role-based Access Control. 32, 33, 36, 46, 86, 95, 124

REST REpresentational State Transfer. 3

ROM Read Only Memory. 51

RPC Remote Procedure Call. 28

SASL Simple Authentication and Security Layer. 25, 27, 28

SM Session Manager. 40, 41, 57, 58, 63, 75, 102–105

SoS System of Systems. 5, 9, 12, 14, 88–94, 120, 122, 123

SSL Secure Socket Layer. 4, 20, 29, 31

TBAC Task-Based Access Control. 32, 36, 46, 47

TCP Transmission Control Protocol. 20–22, 24, 25, 27–31

TLS Transport Layer Security. 4, 20, 21, 25, 28, 29, 31, 50

TOPSIS Technique for Order Preference by Similarity to Ideal Solution. 123

U-XACML XACML with Continuous Usage Control Features. 6, 41–43, 71, 72, 76, 77

UCIoT Usage Control in the Internet of Things. 51

UCON Usage Control. 5–14, 16, 19, 33, 36–43, 45–47, 49, 51, 52, 55, 56, 60, 62–67, 69–73, 76, 79, 83, 85, 86, 88–90, 92, 95–101, 104–106, 108, 120–124

UCS Usage Control System. 9, 39–41, 45, 46, 56–59, 62–64, 66–70, 73–77, 79–84, 90–94, 100, 101, 121–123

UDP User Datagram Protocol. 20–23, 27, 31

URI Uniform Resource Identifier. 22, 27

URL Uniform Resource Locators. 27

W3C World Wide Web Consortium. 26

WAMP Web Application Messaging Protocol. 19, 20, 28, 29

WAR Web application ARchive. 82

XACML eXtensible Access Control Markup Language. 5, 6, 10, 18, 32, 34, 35, 41, 43, 52, 53, 72, 73, 76, 85, 122

XML Extensible Markup Language. 6, 27, 28, 31, 33, 35

XMPP Extensible Messaging and Presence Protocol. 19, 27, 28, 31

Bibliography

[1] Abraham, A.: Hybrid intelligent systems: evolving intelligence in hierarchical layers. In: Do Smart Adaptive Systems Exist?, pp. 159–179. Springer (2005)

[2] Ahl, V., Allen, T.F.: Hierarchy theory: a vision, vocabulary, and epistemology. Columbia University Press (1996)

[3] Al-Fuqaha, A., Guizani, M., Mohammadi, M., Aledhari, M., Ayyash, M.: Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications. IEEE Communications Surveys Tutorials **17**(4), 2347–2376 (Fourthquarter 2015). https://doi.org/10.1109/COMST.2015.2444095

[4] ARTICTE, P.N.: Raster Procedures for Multi-Criteria/Multi-0biective Decisions. Photogrammetric Engineering & Remote Sensing **61**(5), 539–547 (1995)

[5] Asir S Vedamuthu and David Orchard and Frederick Hirsch and Maryann Hondo and Prasad Yendluri and Toufic Boubez and Ümit Yalçinalp: Web Services Policy 1.5 - Primer. techreport, World Wide Web Consortium - W3C (Sep 2007)

[6] Atzori, L., Iera, A., Morabito, G.: The Internet of Things: A survey. Computer Networks **54**(15), 2787 – 2805 (2010). https://doi.org/https://doi.org/10.1016/j.comnet.2010.05.010, http://www.sciencedirect.com/science/article/pii/S1389128610001568

[7] Azimi, I., Anzanpour, A., Rahmani, A.M., Pahikkala, T., Levorato, M., Liljeberg, P., Dutt, N.: HiCH: Hierarchical Fog-Assisted Computing Architecture for Healthcare IoT. ACM Trans. Embed. Comput. Syst. **16**(5s), 174:1–

174:20 (Sep 2017). https://doi.org/10.1145/3126501, http://doi.acm.org/10.1145/3126501

[8] Backes, M., Pfitzmann, B., Schunter, M.: A Toolkit for Managing Enterprise Privacy Policies. In: Snekkenes, E., Gollmann, D. (eds.) Computer Security – ESORICS 2003. pp. 162–180. Springer Berlin Heidelberg, Berlin, Heidelberg (2003)

[9] Bastos, D., Shackleton, M., El-Moussa, F.: Internet of Things: A survey of technologies and security risks in smart home and city environments. In: Living in the Internet of Things: Cybersecurity of the IoT - 2018. pp. 1–7 (March 2018). https://doi.org/10.1049/cp.2018.0030

[10] Bastos, D., Shackleton, M., El-Moussa, F.: Internet of Things: A survey of technologies and security risks in smart home and city environments. In: Living in the Internet of Things: Cybersecurity of the IoT - 2018. pp. 1–7 (March 2018). https://doi.org/10.1049/cp.2018.0030

[11] Becker, M.Y., Fournet, C., Gordon, A.D.: SecPAL: Design and semantics of a decentralized authorization language. Journal of Computer Security **18**(4), 619–665 (2010)

[12] Bera, A.: 80 IoT Statistics (Infographic) (2019), https://safeatlast.co/blog/iot-statistics/

[13] Bettini, C., Jajodia, S., Wang, X.S., Wijesekera, D.: Provisions and Obligations in Policy Rule Management. Journal of Network and Systems Management **11**(3), 351–372 (Sep 2003). https://doi.org/10.1023/A:1025711105609, https://doi.org/10.1023/A:1025711105609

[14] Bormann, C., Ersue, M., Keranen, A.: Terminology for Constrained-Node Networks. RFC 7228, RFC Editor (May 2014), https://tools.ietf.org/html/rfc7228

[15] Bouyssou, D.: Outranking methods. Encyclopedia of optimization pp. 1919–1925 (2001)

[16] Capossele, A., Cervo, V., De Cicco, G., Petrioli, C.: Security as a CoAP resource: an optimized DTLS implementation for the IoT. In: 2015 IEEE international conference on communications (ICC). pp. 549–554. IEEE (2015)

[17] Carniani, E., D'Arenzo, D., Lazouski, A., Martinelli, F., Mori, P.: Usage Control on Cloud Systems. Future Gener. Comput. Syst. **63**(C), 37–55 (Oct 2016). https://doi.org/10.1016/j.future.2016.04.010, http://dx.doi.org/10.1016/j.future.2016.04.010

[18] Chadwick, D., Lischka, M.: Obligation Standardization. In: W3C Workshop on Access Control Application Scenarios. pp. 1–5 (2009), https://www.w3.org/2009/policy-ws/papers/Chadwick.pdf

[19] Chen, D., Varshney, P.K.: QoS Support in Wireless Sensor Networks: A Survey. In: International conference on wireless networks. vol. 233, pp. 1–7 (2004)

[20] Chun, S.M., Park, J.T.: Mobile CoAP for IoT mobility management. In: 2015 12th Annual IEEE Consumer Communications and Networking Conference (CCNC). pp. 283–289 (Jan 2015). https://doi.org/10.1109/CCNC.2015.7157990

[21] Cirillo, F., Wu, F.J., Solmaz, G., Kovacs, E.: Embracing the Future Internet of Things. Sensors **19**(2), 351 (2019)

[22] Colitti, W., Steenhaut, K., Caro, N.D., Buta, B., Dobrota, V.: Evaluation of constrained application protocol for wireless sensor networks. In: 2011 18th IEEE Workshop on Local Metropolitan Area Networks (LANMAN). pp. 1–6 (Oct 2011). https://doi.org/10.1109/LANMAN.2011.6076934

[23] Colitti, W., Steenhaut, K., Caro, N.D., Buta, B., Dobrota, V.: Evaluation of constrained application protocol for wireless sensor networks. In: 2011 18th IEEE Workshop on Local Metropolitan Area Networks (LANMAN). pp. 1–6 (Oct 2011). https://doi.org/10.1109/LANMAN.2011.6076934

[24] Collina, M., Corazza, G.E., Vanelli-Coralli, A.: Introducing the QEST broker: Scaling the IoT by bridging MQTT and REST. In: 2012 IEEE 23rd International Symposium on Personal, Indoor and Mobile Radio Communications - (PIMRC). pp. 36–41 (Sept 2012). https://doi.org/10.1109/PIMRC.2012.6362813

[25] Colombo, A.W., Bangemann, T., Karnouskos, S., Delsing, J., Stluka, P., Harrison, R., Jammes, F., Lastra, J.L., et al.: Industrial cloud-based cyber-physical systems. The IMC-AESOP Approach (2014)

[26] Colombo, M., Lazouski, A., Martinelli, F., Mori, P.: A proposal on enhancing XACML with continuous usage control features. In: Grids, P2P and Services Computing, pp. 133–146. Springer US (2010)

[27] Costantino, G., La Marra, A., Martinelli, F., Mori, P., Saracino, A.: Privacy Preserving Distributed Attribute Computation for Usage Control in the Internet of Things. In: 2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/ 12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE). pp. 1844–1851 (Aug 2018). https://doi.org/10.1109/TrustCom/BigDataSE.2018.00279

[28] crew, I.O.: Hypertext Transfer Protocol (HTTP) (2018), https://www.iotone.com/term/hypertext-transfer-protocol-http/t557

[29] Crossbar.io: Web application messaging protocol (wamp) (2012), https://wamp-proto.org/

[30] Damianou, N., Dulay, N., Lupu, E., Sloman, M.: The Ponder Policy Specification Language. In: Sloman, M., Lupu, E.C., Lobo, J. (eds.) Policies for Distributed Systems and Networks. pp. 18–38. Springer Berlin Heidelberg, Berlin, Heidelberg (2001)

[31] Demchenko, Y., Koeroo, O., de Laat, C., Sagehaug, H.: Extending XACML Authorisation Model to Support Policy Obligations Handling in Distributed Application. In: Proceedings of the 6th International Workshop on Middleware for Grid Computing. pp. 5:1–5:6. MGC '08, ACM, New York, NY, USA (2008). https://doi.org/10.1145/1462704.1462709, http://doi.acm.org/10.1145/1462704.1462709

[32] Department, S.R.: Internet of Things (IoT) connected devices installed base worldwide from 2015 to 2025 (2016), https://www.statista.com/statistics/471264/iot-number-of-connected-devices-worldwide

[33] Ersue, M., Romascanu, D., Schoenwaelder, J., Herberg, U.: Management of Networks with Constrained Devices: Problem Statement and Requirements. RFC 7547, RFC Editor (May 2015), https://tools.ietf.org/html/rfc7547

[34] Etzion, O., Fournier, F., Arcushin, S.: Tutorial on the Internet of Everything. In: Proceedings of the 8th ACM International Conference on Dis-

tributed Event-Based Systems. pp. 236–237. DEBS '14, ACM, New York, NY, USA (2014). https://doi.org/10.1145/2611286.2611308, http://doi.acm.org/10.1145/2611286.2611308

[35] Faiella, M., Martinelli, F., Mori, P., Saracino, A., Sheikhalishahi, M.: Collaborative Attribute Retrieval in Environment with Faulty Attribute Managers. In: 2016 11th International Conference on Availability, Reliability and Security (ARES). pp. 296–303 (Aug 2016). https://doi.org/10.1109/ARES.2016.51

[36] Farroha, B., Farroha, D.: Challenges of "operationalizing" dynamic system access control: Transitioning from ABAC to RAdAC. In: 2012 IEEE International Systems Conference SysCon 2012. pp. 1–7 (March 2012). https://doi.org/10.1109/SysCon.2012.6189525

[37] Fette, I., Melnikov, A.: The WebSocket Protocol. 6455 (Dec 2011). https://doi.org/110.17487/RFC6455, https://tools.ietf.org/html/rfc6455

[38] Foundation, N.S.: Cyber-Physical Systems (CPS) (2012), https://www.nsf.gov/pubs/2011/nsf11516/nsf11516.pdf

[39] Foundation, X.S.: Xep-0060: Publish-subscribe (October 2019), https://xmpp.org/extensions/xep-0060.html

[40] Fysarakis, K., Askoxylakis, I., Soultatos, O., Papaefstathiou, I., Manifavas, C., Katos, V.: Which IoT protocol? comparing standardized approaches over a common m2m application. In: Global Communications Conference (GLOBECOM), 2016 IEEE. pp. 1–7. IEEE (2016)

[41] Geissbauer, R., Vedso, J., Schrauf, S.: Industry 4.0: Building the digital enterprise. https://pwc.to/2Hr5BuU, accessed: 2019 - 02 - 21

[42] Gerdes, S., Bergmann, O., Bormann, C., Selander, G., Seitz, L.: Datagram Transport Layer Security (DTLS) Profile for Authentication and Authorization for Constrained Environments (ACE). Internet-Draft draft-ietf-ace-dtls-authorize-07, Internet Engineering Task Force (Mar 2019), https://datatracker.ietf.org/doc/html/draft-ietf-ace-dtls-authorize-07, work in Progress

[43] Gershenfeld, N., Krikorian, R., Cohen, D.: The Internet of Things. Scientific American **291**(4), 76–81 (2004), http://www.jstor.org/stable/26060727

[44] Giusto, D., Iera, A., Morabito, G., Atzori, L.: The Internet of Things: 20th Tyrrhenian Workshop on Digital Communications. Springer Publishing Company, Incorporated (2014)

[45] Gkioulos, V., Rizos, A., Michailidou, C., Martinelli, F., Mori, P.: Enhancing Usage Control for Performance: A Proposal for Systems of Systems (Research Poster). In: 2018 International Conference on High Performance Computing Simulation (HPCS). pp. 1061–1062 (July 2018). https://doi.org/10.1109/HPCS.2018.00169

[46] Gkioulos, V., Rizos, A., Michailidou, C., Mori, P., Saracino, A.: Enhancing Usage Control for Performance: An Architecture for Systems of Systems. In: Katsikas, S.K., Cuppens, F., Cuppens, N., Lambrinoudakis, C., Antón, A., Gritzalis, S., Mylopoulos, J., Kalloniatis, C. (eds.) Computer Security. pp. 69–84. Springer International Publishing, Cham (2019)

[47] Gkioulos, V., Wolthusen, S.: Enabling dynamic security policy evaluation for service-oriented architectures in tactical networks. Norsk informasjonssikkerhetskonferanse (NISK) **8**(1), 109–120 (Dec 2015)

[48] Gkioulos, V., Wolthusen, S.D.: Reconciliation of ontologically defined security policies for tactical service oriented architectures. In: Doss, R., Piramuthu, S., Zhou, W. (eds.) Future Network Systems and Security. pp. 47–61. Springer International Publishing, Cham (2016), Efficient Security Policy Reconciliation in Tactical Service Oriented Architectures

[49] Gkioulos, V., Wolthusen, S.D.: A Security Policy Infrastructure for Tactical Service Oriented Architectures. In: Cuppens-Boulahia, N., Lambrinoudakis, C., Cuppens, F., Katsikas, S. (eds.) Security of Industrial Control Systems and Cyber-Physical Systems. pp. 37–51. Springer International Publishing, Cham (2017)

[50] Gkioulos, V., Wolthusen, S.D.: Constraint Analysis for Security Policy Partitioning Over Tactical Service Oriented Architectures. In: Grzenda, M., Awad, A.I., Furtak, J., Legierski, J. (eds.) Advances in Network Systems. pp. 149–166. Springer International Publishing, Cham (2017)

[51] Gkioulos, V., Wolthusen, S.D., Flizikowski, A., Stachowicz, A., Nogalski, D., Gleba, K., Sliwa, J.: Interoperability of security and quality of Service Policies Over Tactical SOA. In: 2016 IEEE Symposium Series on Computational Intelligence (SSCI). pp. 1–7 (Dec 2016). https://doi.org/10.1109/SSCI.2016.7850077

[52] Granjal, J., Monteiro, E., Silva, J.S.: Security for the internet of things: a survey of existing protocols and open research issues. IEEE Communications Surveys & Tutorials **17**(3), 1294–1312 (2015)

[53] Greene, R., Devillers, R., Luther, J.E., Eddy, B.G.: GIS-based multiple-criteria decision analysis. Geography Compass **5**(6), 412–432 (2011)

[54] Gubbi, J., Buyya, R., Marusic, S., Palaniswami, M.: Internet of Things (IoT): A vision, architectural elements, and future directions. Future Generation Computer Systems **29**(7), 1645 – 1660 (2013). https://doi.org/https://doi.org/10.1016/j.future.2013.01.010, http://www.sciencedirect.com/science/article/pii/S0167739X13000241, including Special sections: Cyber-enabled Distributed Computing for Ubiquitous Cloud and Network Services & Cloud Computing and Scientific Applications âĂŤ Big Data, Scalable Analytics, and Beyond

[55] Guo, J., Chen, I., Tsai, J.J.P., Al-Hamadi, H.: A hierarchical cloud architecture for integrated mobility, service, and trust management of service-oriented IoT systems. In: 2016 Sixth International Conference on Innovative Computing Technology (INTECH). pp. 72–77 (Aug 2016). https://doi.org/10.1109/INTECH.2016.7845021

[56] Harrison, M.A., Ruzzo, W.L., Ullman, J.D.: Protection in Operating Systems. Commun. ACM **19**(8), 461–471 (Aug 1976). https://doi.org/10.1145/360303.360333, http://doi.acm.org/10.1145/360303.360333

[57] Hart, J.K., Martinez, K.: Environmental Sensor Networks: A revolution in the earth system science? Earth-Science Reviews **78**(3), 177 – 191 (2006). https://doi.org/https://doi.org/10.1016/j.earscirev.2006.05.001, http://www.sciencedirect.com/science/article/pii/S0012825206000511

[58] Hartke, K.: Observing Resources in the Constrained Application Protocol (CoAP). RFC 7641 (Sep 2015). https://doi.org/10.17487/RFC7641, https://rfc-editor.org/rfc/rfc7641.txt

[59] Hilty, M., Basin, D., Pretschner, A.: On Obligations. In: di Vimercati, S.d.C., Syverson, P., Gollmann, D. (eds.) Computer Security – ESORICS 2005. pp. 98–117. Springer Berlin Heidelberg, Berlin, Heidelberg (2005)

[60] Hu, V.C., Ferraiolo, D., Kuhn, R., Friedman, A.R., Lang, A.J., Cogdell, M.M., Schnitzer, A., Sandlin, K., Miller, R., Scarfone, K.: Guide to attribute based access control (ABAC) definition and considerations. National Institute of Standards and Technology (NIST) Special Publication **800**(162) (2013)

[61] Hussain, F.: Internet of Everything, pp. 1–11. Springer International Publishing, Cham (2017). https://doi.org/10.1007/978-3-319-55405-1_1, https://doi.org/10.1007/978-3-319-55405-1_1

[62] Ianella, R.: Open Digital Rights Language (ODRL), https://www.w3.org/ns/odrl/2/ODRL20.html

[63] Jara, A.J., Ladid, L., Gómez-Skarmeta, A.F.: The Internet of Everything through IPv6: An Analysis of Challenges, Solutions and Opportunities. JoWua **4**(3), 97–118 (2013)

[64] Kagal, L., Finin, T., Joshi, A.: A policy language for a pervasive computing environment. In: Proceedings POLICY 2003. IEEE 4th International Workshop on Policies for Distributed Systems and Networks. pp. 63–74 (June 2003). https://doi.org/10.1109/POLICY.2003.1206958

[65] Karagiannis, V., Chatzimisios, P., VÃązquez-Gallego, F., Alonso-ZÃąrate, J.: A Survey on Application Layer Protocols for the Internet of Things. Transaction on IoT and Cloud Computing **1**(1) (Jan 2015). https://doi.org/10.5281/zenodo.51613, https://doi.org/10.5281/zenodo.51613

[66] Karopoulos, G., Mori, P., Martinelli, F.: Usage Control in SIP-based Multimedia Delivery. Comput. Secur. **39**, 406–418 (Nov 2013). https://doi.org/10.1016/j.cose.2013.09.005, http://dx.doi.org/10.1016/j.cose.2013.09.005

[67] Kasem-Madani, S., Meier, M.: Security and Privacy Policy Languages: A Survey, Categorization and Gap Identification. CoRR **abs/1512.00201** (2015)

[68] Keoh, S.L., Kumar, S.S., Tschofenig, H.: Securing the Internet of Things: A Standardization Perspective. IEEE Internet of Things Journal **1**(3), 265–275 (June 2014). https://doi.org/10.1109/JIOT.2014.2323395

[69] Khaitan, S.K., McCalley, J.D.: Design Techniques and Applications of Cyberphysical Systems: A Survey. IEEE Systems Journal **9**(2), 350–365 (June 2015). https://doi.org/10.1109/JSYST.2014.2322503

[70] Krautsevich, L., Lazouski, A., Martinelli, F., Mori, P., Yautsiukhin, A.: Usage Control, Risk and Trust, pp. 1–12. Springer Berlin Heidelberg, Berlin, Heidelberg (2010). https://doi.org/10.1007/978-3-642-15152-1_1, https://doi.org/10.1007/978-3-642-15152-1_1

[71] Kyng, M., Nielsen, E.T., Kristensen, M.: Challenges in Designing Interactive Systems for Emergency Response. In: Proceedings of the 6th Conference on Designing Interactive Systems. pp. 301–310. DIS ’06, ACM, New York, NY, USA (2006). https://doi.org/10.1145/1142405.1142450, http://doi.acm.org/10.1145/1142405.1142450

[72] La Marra, A., Martinelli, F., Mori, P., Rizos, A., Saracino, A.: Improving MQTT by Inclusion of Usage Control. In: Wang, G., Atiquzzaman, M., Yan, Z., Choo, K.K.R. (eds.) Security, Privacy, and Anonymity in Computation, Communication, and Storage. pp. 545–560. Springer International Publishing, Cham (2017)

[73] La Marra, A., Martinelli, F., Mori, P., Rizos, A., Saracino, A.: Introducing Usage Control in MQTT. In: Katsikas, S.K., Cuppens, F., Cuppens, N., Lambrinoudakis, C., Kalloniatis, C., Mylopoulos, J., Antón, A., Gritzalis, S. (eds.) Computer Security. pp. 35–43. Springer International Publishing, Cham (2018)

[74] La Marra, A., Martinelli, F., Mori, P., Rizos, A., Saracino, A.: Using IFTTT to Express and Enforce UCON Obligations. In: Heng, S.H., Lopez, J. (eds.) Information Security Practice and Experience. pp. 213–231. Springer International Publishing, Cham (2019)

[75] Lazouski, A.: Access and Usage Control in Grid (2011)

[76] Lazouski, A., Martinelli, F., Mori, P.: Survey: Usage Control in Computer Security: A Survey. Comput. Sci. Rev. **4**(2), 81–99 (May 2010). https://doi.org/10.1016/j.cosrev.2010.02.002, http://dx.doi.org/10.1016/j.cosrev.2010.02.002

[77] Lazouski, A., Martinelli, F., Mori, P., Saracino, A.: Stateful Data Usage Control for Android Mobile Devices. International Journal of Information Security pp. 1–25 (2016). https://doi.org/10.1007/s10207-016-0336-y, http://dx.doi.org/10.1007/s10207-016-0336-y

[78] Le, A., Keller, L., Fragouli, C., Markopoulou, A.: MicroPlay: A Networking Framework for Local Multiplayer Games. In: Proceedings of the First ACM International Workshop on Mobile Gaming. pp. 13–18. MobileGames '12, ACM, New York, NY, USA (2012). https://doi.org/10.1145/2342480.2342485, http://doi.acm.org/10.1145/2342480.2342485

[79] Lesjak, C., Hein, D., Hofmann, M., Maritsch, M., Aldrian, A., Priller, P., Ebner, T., Ruprechter, T., Pregartner, G.: Securing smart maintenance services: Hardware-security and TLS for MQTT. In: 2015 IEEE 13th International Conference on Industrial Informatics (INDIN). pp. 1243–1250 (July 2015). https://doi.org/10.1109/INDIN.2015.7281913

[80] Li, L., Li, Y., Lin, J., Zhang, T.: Enhanced AllJoyn Network with Centralized Management, pp. 183–188. Springer Berlin Heidelberg, Berlin, Heidelberg (2015). https://doi.org/10.1007/978-3-662-47401-3_24, http://dx.doi.org/10.1007/978-3-662-47401-3_24

[81] Lin, H., Bergmann, N.W.: IoT Privacy and Security Challenges for Smart Home Environments. Information **7**(3) (2016). https://doi.org/10.3390/info7030044, http://www.mdpi.com/2078-2489/7/3/44

[82] Locke, D.: Mq telemetry transport (mqtt) v3. 1 protocol specification. IBM developerWorks Technical Library (2010)

[83] Luzuriaga, J.E., Cano, J.C., Calafate, C., Manzoni, P., Perez, M., Boronat, P.: Handling mobility in IoT applications using the MQTT protocol. In:

2015 Internet Technologies and Applications (ITA). pp. 245–250 (Sept 2015). https://doi.org/10.1109/ITechA.2015.7317403

[84] Malczewski, J.: GIS and multicriteria decision analysis. John Wiley & Sons (1999)

[85] Marra, A.L., Martinelli, F., Mori, P., Saracino, A.: Implementing Usage Control in Internet of Things: A Smart Home Use Case. In: 2017 IEEE Trustcom/BigDataSE/ICESS. pp. 1056–1063 (Aug 2017). https://doi.org/10.1109/Trustcom/BigDataSE/ICESS.2017.352

[86] Martinelli, F., Michailidou, C., Mori, P., Saracino, A.: Too Long, did not Enforce: A Qualitative Hierarchical Risk-Aware Data Usage Control Model for Complex Policies in Distributed Environments. In: Proceedings of the 4th ACM Workshop on Cyber-Physical System Security, CPSS@AsiaCCS 2018, Incheon, Republic of Korea, June 04-08, 2018. pp. 27–37 (2018). https://doi.org/doi:10.1145/3198458.3198463, http://doi.acm.org/10.1145/3198458.3198463

[87] Martinelli, F., Mori, P.: On Usage Control for GRID Systems. Future Gener. Comput. Syst. **26**(7), 1032–1042 (Jul 2010). https://doi.org/10.1016/j.future.2009.12.005, http://dx.doi.org/10.1016/j.future.2009.12.005

[88] McIntire, C.D., Colby, J.A.: A Hierarchical Model of Lotic Ecosystems. Ecological Monographs **48**(2), 167–190 (1978). https://doi.org/10.2307/2937298, https://esajournals.onlinelibrary.wiley.com/doi/abs/10.2307/2937298

[89] McIntire, C.D., Colby, J.A.: A hierarchical model of lotic ecosystems. Ecological Monographs **48**(2), 167–190 (1978)

[90] Melnikov, A., Fette, I.: The WebSocket Protocol. RFC 6455 (dec 2011). https://doi.org/10.17487/RFC6455, https://rfc-editor.org/rfc/rfc6455.txt

[91] van der Meulen, R.: Gartner Says 8.4 Billion Connected “Things” Will Be in Use in 2017, Up 31 Percent From 2016 (2017), https://www.gartner.com/newsroom/id/3598917

[92] Miraz, M.H., Ali, M., Excell, P.S., Picking, R.: A review on Internet of Things (IoT), Internet of Everything (IoE) and Internet of Nano Things (IoNT). In: 2015 Internet Technologies and Applications (ITA). pp. 219–224 (Sep 2015). https://doi.org/10.1109/ITechA.2015.7317398

[93] Moore, B., Ellesson, E., Strassner, J., Westerinen, A.: RFC 3060: Policy Core Information Model – Version 1 Specification (Feb 2001), http://www.ietf.org/rfc/rfc3060.txt

[94] Nadkarni, A., Enck, W., Jha, S., Staddon, J.: Policy by Example: An Approach for Security Policy Specification. arXiv preprint arXiv:1707.03967 (2017)

[95] Namiot, D., Sneps-Sneppe, M.: Proximity as a service. In: 2012 2nd Baltic Congress on Future Internet Communications. pp. 199–205 (April 2012). https://doi.org/10.1109/BCFIC.2012.6217947

[96] NIST: MQTT and the NIST Cybersecurity Framework Version 1.0. http://docs.oasis-open.org/mqtt/mqtt-nist-cybersecurity/v1.0/cn01/mqtt-nist-cybersecurity-v1.0-cn01.pdf (2014), [Online; accessed 22-January-2017]

[97] OASIS: Oasis advanced message queuing protocol (amqp) version 1.0 (October 2012), http://docs.oasis-open.org/amqp/core/v1.0/os/amqp-core-overview-v1.0-os.html

[98] OASIS Standard: eXtensible Access Control Markup Language (XACML) Version 3.0. http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-os-en.html. (Jan 2013)

[99] OASIS Standard: eXtensible Access Control Markup Language (XACML) Version 3.0. http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-os-en.html. (Jan 2013)

[100] O'Connor, A.C., Loomis, R.J.: 2010 economic analysis of role-based access control. NIST, Gaithersburg, MD **20899** (2010)

[101] O'Grady, P.J., Kim, Y., Young, R.E.: A hierarchical approach to concurrent engineering systems. International Journal of Computer Integrated Manufacturing **7**(3), 152–162 (1994).

https://doi.org/10.1080/09511929408944605, https://doi.org/10.1080/09511929408944605

[102] O'Neill, M.: The Internet of Things: do more devices mean more risks? Computer Fraud & Security **2014**(1), 16 – 17 (2014). https://doi.org/https://doi.org/10.1016/S1361-3723(14)70008-9, http://www.sciencedirect.com/science/article/pii/S1361372314700089

[103] Ovadia, S.: Automate the Internet With "If This Then That" (IFTTT). Behavioral & Social Sciences Librarian **33**(4), 208–211 (2014). https://doi.org/10.1080/01639269.2014.964593, https://doi.org/10.1080/01639269.2014.964593

[104] Palumbo, F., Ullberg, J., Štimec, A., Furfari, F., Karlsson, L., Coradeschi, S.: Sensor network infrastructure for a home care monitoring system. Sensors **14**(3), 3833–3860 (2014)

[105] Park, J., Sandhu, R.: Towards Usage Control Models: Beyond Traditional Access Control. In: Proceedings of the Seventh ACM Symposium on Access Control Models and Technologies. pp. 57–64. SACMAT '02, ACM, New York, NY, USA (2002). https://doi.org/10.1145/507711.507722, http://doi.acm.org/10.1145/507711.507722

[106] Park, J., Sandhu, R.: The UCONABC Usage Control Model. ACM Trans. Inf. Syst. Secur. **7**(1), 128–174 (Feb 2004). https://doi.org/10.1145/984334.984339, http://doi.acm.org/10.1145/984334.984339

[107] Pattee, H.H.: Hierarchy Theory: The Challenge of Complex Systems (1973)

[108] Pimentel, V., Nickerson, B.G.: Communicating and Displaying Real-Time Data with WebSocket. IEEE Internet Computing **16**(4), 45–53 (July 2012). https://doi.org/10.1109/MIC.2012.64

[109] Pretschner, A., Hilty, M., Basin, D.: Distributed Usage Control. Commun. ACM **49**(9), 39–44 (Sep 2006). https://doi.org/10.1145/1151030.1151053, http://doi.acm.org/10.1145/1151030.1151053

[110] Rad, C.R., Hancu, O., Takacs, I.A., Olteanu, G.: Smart Monitoring of Potato Crop: A Cyber-Physical System Architecture Model in the Field of Precision Agriculture. Agriculture and Agricultural Science Procedia **6**, 73 – 79

(2015). https://doi.org/http://dx.doi.org/10.1016/j.aaspro.2015.08.041, http://www.sciencedirect.com/science/article/pii/S2210784315001746, conference Agriculture for Life, Life for Agriculture

[111] Raza, S., Shafagh, H., Hewage, K., Hummen, R., Voigt, T.: Lithe: Lightweight Secure CoAP for the Internet of Things. IEEE Sensors Journal **13**(10), 3711–3720 (Oct 2013). https://doi.org/10.1109/JSEN.2013.2277656

[112] Raza, S., Duquennoy, S., HÃűglund, J., Roedig, U., Voigt, T.: Secure communication for the Internet of Things: a comparison of link-layer security and IPsec for 6LoWPAN. Security and Communication Networks **7**(12), 2654–2668 (2014). https://doi.org/10.1002/sec.406, http://dx.doi.org/10.1002/sec.406

[113] Rescorla, E., Modadugu, N.: Datagram Transport Layer Security Version 1.2. RFC 6347 (Jan 2012). https://doi.org/10.17487/RFC6347, https://rfc-editor.org/rfc/rfc6347.txt

[114] Rescorla, E., Tschofenig, H., Modadugu, N.: The Datagram Transport Layer Security (DTLS) Protocol Version 1.3. Internet-Draft draft-ietf-tls-dtls13-31, Internet Engineering Task Force (Mar 2019), https://datatracker.ietf.org/doc/html/draft-ietf-tls-dtls13-31, work in Progress

[115] RFID, I.D.N.E.., Nanosystems, I.G.M..: Internet of Things in 2020, A Roadmap for the Future (2009)

[116] Rizos, A., Bastos, D., Saracino, A., Martinelli, F.: Distributed UCON in CoAP and MQTT Protocols. In: Computer Security. Springer International Publishing, Cham (2019), to Appear

[117] Roca, D., Milito, R., Nemirovsky, M., Valero, M.: Tackling IoT Ultra Large Scale Systems: Fog Computing in Support of Hierarchical Emergent Behaviors, pp. 33–48. Springer International Publishing, Cham (2018). https://doi.org/10.1007/978-3-319-57639-8_3, https://doi.org/10.1007/978-3-319-57639-8_3

[118] Rouse, M.: Interner of Things: A definition (2016), https://internetofthingsagenda.techtarget.com/definition/Internet-of-Things-IoT

[119] Saaty, R.: The analytic hierarchy process - what it is and how it is used. Mathematical Modelling **9**(3), 161 – 176 (1987). https://doi.org/http://dx.doi.org/10.1016/0270-0255(87)90473-8, http://www.sciencedirect.com/science/article/pii/0270025587904738

[120] Saaty, R.: The analytic hierarchy process - what it is and how it is used. Mathematical Modelling **9**(3), 161 – 176 (1987). https://doi.org/http://dx.doi.org/10.1016/0270-0255(87)90473-8, http://www.sciencedirect.com/science/article/pii/0270025587904738

[121] Saaty, T.L.: The analytic hierarchy and analytic network processes for the measurement of intangible criteria and for decision-making. In: Multiple criteria decision analysis, pp. 363–419. Springer (2016)

[122] Saint-Andre, P.: Extensible Messaging and Presence Protocol (XMPP): Core. RFC 6120 (Mar 2011). https://doi.org/10.17487/RFC6120, https://rfc-editor.org/rfc/rfc6120.txt

[123] Salthe, S.N.: A Hierarchical Framework for Levels of Reality: Understanding Through Representation. Axiomathes **19**(1), 87–99 (Mar 2009). https://doi.org/10.1007/s10516-008-9056-x, https://doi.org/10.1007/s10516-008-9056-x

[124] Salthe, S.N.: Evolving hierarchical systems. Columbia University Press (2010)

[125] Samarati, P., de Vimercati, S.C.: Access Control: Policies, Models, and Mechanisms, pp. 137–196. Springer Berlin Heidelberg, Berlin, Heidelberg (2001). https://doi.org/10.1007/3-540-45608-2_3, https://doi.org/10.1007/3-540-45608-2_3

[126] Sandhu, R.S., Samarati, P.: Access control: principle and practice. IEEE Communications Magazine **32**(9), 40–48 (Sept 1994). https://doi.org/10.1109/35.312842

[127] Sandhu, R.S., Coyne, E.J., Feinstein, H.L., Youman, C.E.: Role-Based Access Control Models. Computer **29**(2), 38–47 (Feb 1996). https://doi.org/10.1109/2.485845, http://dx.doi.org/10.1109/2.485845

[128] Security, H.: Number of connected devices reached 22 billion, where is the revenue? (2019), https://www.helpnetsecurity.com/2019/05/23/connected-devices-growth

[129] Shalaginov, A.: Soft Computing and Hybrid Intelligence for Decision Support in Forensics Science. In: 2016 IEEE Conference on Intelligence and Security Informatics (ISI). pp. 304–306 (Sep 2016). https://doi.org/10.1109/ISI.2016.7745495

[130] Shalaginov, A.: Advancing Neuro-Fuzzy Algorithm for Automated Classification in Largescale Forensic and Cybercrime Investigations: Adaptive Machine Learning for Big Data Forensic. Ph.D. thesis, Norwegian University of Science and Technology (2018)

[131] Shelby, Z., Hartke, K., Bormann, C.: The Constrained Application Protocol (CoAP). RFC 7252 (Jun 2014). https://doi.org/10.17487/RFC7252, https://rfc-editor.org/rfc/rfc7252.txt

[132] Shirey, R.: RFC 4949: Internet Security Glossary – Version 2 (Aug 2007), https://tools.ietf.org/rfc/rfc4949.txt

[133] Simon, H.A.: The Organization of Complex Systems, pp. 245–261. Springer Netherlands, Dordrecht (1977). https://doi.org/10.1007/978-94-010-9521-1_14, https://doi.org/10.1007/978-94-010-9521-1_14

[134] Simon, H.A.: The Architecture of Complexity, pp. 457–476. Springer US, Boston, MA (1991). https://doi.org/10.1007/978-1-4899-0718-9_31, https://doi.org/10.1007/978-1-4899-0718-9_31

[135] Singh, M., Rajan, M.A., Shivraj, V.L., Balamuralidhar, P.: Secure MQTT for Internet of Things (IoT). In: 2015 Fifth International Conference on Communication Systems and Network Technologies. pp. 746–751 (April 2015). https://doi.org/10.1109/CSNT.2015.16

[136] Smith, N., Sage, A.: An introduction to hierarchical systems theory. Computers & Electrical Engineering **1**(1), 55 – 71 (1973). https://doi.org/https://doi.org/10.1016/0045-7906(73)90027-X, http://www.sciencedirect.com/science/article/pii/004579067390027X

[137] Smith, S.W.: Trusted Computing Platforms, pp. 1–239. No. 1, Springer US (2005). https://doi.org/10.1007/b103637, https://www.springer.com/gp/book/9780387239163

[138] Srivatsa, M., Liu, L.: Securing Publish-subscribe Overlay Services with EventGuard. In: Proceedings of the 12th ACM Conference on Computer and Communications Security. pp. 289–298. CCS '05, ACM, New York, NY, USA (2005). https://doi.org/10.1145/1102120.1102158, http://doi.acm.org/10.1145/1102120.1102158

[139] for Standardization, I.O.: Iso/iec 19464:2014 - advanced message queuing protocol (amqp) v1.0 specification (2014), https://www.iso.org/obp/ui/#iso:std:iso-iec:19464:ed-1:v1:en

[140] Surbatovich, M., Aljuraidan, J., Bauer, L., Das, A., Jia, L.: Some Recipes Can Do More Than Spoil Your Appetite: Analyzing the Security and Privacy Risks of IFTTT Recipes. In: Proceedings of the 26th International Conference on World Wide Web. pp. 1501–1510. WWW '17, International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, Switzerland (2017). https://doi.org/10.1145/3038912.3052709, https://doi.org/10.1145/3038912.3052709

[141] Talaminos-Barroso, A., Estudillo-Valderrama, M.A., Roa, L.M., Reina-Tosina, J., Ortega-Ruiz, F.: A Machine-to-Machine protocol benchmark for eHealth applications - Use case: Respiratory rehabilitation. Computer Methods and Programs in Biomedicine **129**, 1 – 11 (2016). https://doi.org/http://dx.doi.org/10.1016/j.cmpb.2016.03.004, http://www.sciencedirect.com/science/article/pii/S0169260715302959

[142] Tan, Y., Goddard, S., Pérez, L.C.: A Prototype Architecture for Cyber-physical Systems. SIGBED Rev. **5**(1), 26:1–26:2 (Jan 2008). https://doi.org/10.1145/1366283.1366309, http://doi.acm.org/10.1145/1366283.1366309

[143] Thangavel, D., Ma, X., Valera, A., Tan, H.X., Tan, C.K.Y.: Performance evaluation of MQTT and CoAP via a common middleware. In: Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP), 2014 IEEE Ninth International Conference on. pp. 1–6. IEEE (2014)

[144] Thomas, R.K., Sandhu, R.S.: Task-based authorization controls (TBAC): a family of models for active and enterprise-oriented authorization management, pp. 166–181. Springer US, Boston, MA (1998). https://doi.org/10.1007/978-0-387-35285-5_10, https://doi.org/10.1007/978-0-387-35285-5_10

[145] Tiloca, M., Selander, G., Palombini, F., Park, J.: Group OSCORE - Secure Group Communication for CoAP. Internet-Draft draft-ietf-core-oscore-groupcomm-04, Internet Engineering Task Force (Mar 2019), https://datatracker.ietf.org/doc/html/draft-ietf-core-oscore-groupcomm-04, work in Progress

[146] Tragos, E.Z., Angelakis, V., Fragkiadakis, A., Gundlegard, D., Nechifor, C.S., Oikonomou, G., PÃűhls, H.C., Gavras, A.: Enabling reliable and secure IoT-based smart city applications. In: 2014 IEEE International Conference on Pervasive Computing and Communication Workshops (PERCOM WORKSHOPS). pp. 111–116 (March 2014). https://doi.org/10.1109/PerComW.2014.6815175

[147] TrendLabs: Trend Micro Security Predictions for 2018 (2017), http://bit.ly/30PQ2og

[148] Ukil, A., Bandyopadhyay, S., Bhattacharyya, A., Pal, A., Bose, T.: Lightweight security scheme for IoT applications using CoAP. International Journal of Pervasive Computing and Communications **10**(4), 372–392 (2014)

[149] Uszok, A., Bradshaw, J., Jeffers, R., Suri, N., Hayes, P., Breedy, M., Bunch, L., Johnson, M., Kulkarni, S., Lott, J.: KAoS policy and domain services: toward a description-logic approach to policy representation, deconfliction, and enforcement. In: Proceedings POLICY 2003. IEEE 4th International Workshop on Policies for Distributed Systems and Networks. pp. 93–96 (June 2003). https://doi.org/10.1109/POLICY.2003.1206963

[150] Valentine, J.W., May, C.L.: Hierarchies in biology and paleontology. Paleobiology **22**(1), 23âĂŞ33 (1996). https://doi.org/10.1017/S0094837300015992

[151] Villari, M., Celesti, A., Fazio, M., Puliafito, A.: AllJoyn Lambda: An architecture for the management of smart environments in IoT. In: 2014 International Conference on Smart Computing Workshops. pp. 9–14 (Nov 2014). https://doi.org/10.1109/SMARTCOMP-W.2014.7046676

[152] De Capitani di Vimercati, S., Samarati, P., Jajodia, S.: Policies, Models, and Languages for Access Control. In: Proceedings of the 4th International Conference on Databases in Networked Information Systems. pp. 225–237. DNIS'05, Springer-Verlag, Berlin, Heidelberg (2005). https://doi.org/10.1007/978-3-540-31970-2_18, http://dx.doi.org/10.1007/978-3-540-31970-2_18

[153] Vorapojpisut, S.: A Lightweight Framework of Home Automation Systems Based on the IFTTT Model. JSW **10**(12), 1343–1350 (2015)

[154] Wang, Y., Wei, L., Jin, Q., Ma, J.: Alljoyn Based Direct Proximity Service Development: Overview and Prototype. In: 2014 IEEE 17th International Conference on Computational Science and Engineering. pp. 634–641 (Dec 2014). https://doi.org/10.1109/CSE.2014.138

[155] Westerinen, A., Schnizlein, J., Strassner, J., Scherling, M., Quinn, B., Herzog, S., Huynh, A., Carlson, M., Perry, J., Waldbusser, S.: Terminology for Policy-Based Management. RFC 3198, RFC Editor (November 2001), https://tools.ietf.org/html/rfc3198

[156] Wilson, D.: Forms of hierarchy: A selected bibliography. General Systems **14**, 3–+ (1969)

[157] Woolf, N.: DDoS attack that disrupted internet was largest of its kind in history, experts say (2016), https://www.theguardian.com/technology/2016/oct/26/ddos-attack-dyn-mirai-botnet

[158] Wu, J.: Hierarchy and Scaling: Extrapolating Information along a Scaling Ladder. Canadian Journal of Remote Sensing **25**(4), 367–380 (1999). https://doi.org/10.1080/07038992.1999.10874736, https://doi.org/10.1080/07038992.1999.10874736

[159] Wu, J., Loucks, O.L.: From Balance of Nature to Hierarchical Patch Dynamics: A Paradigm Shift in Ecology. The Quarterly Review of Biology

70(4), 439–466 (1995). https://doi.org/10.1086/419172, https://doi.org/10.1086/419172

[160] Yang, L., Di Martino, B., Zhang, Q.: Internet of Everything. Mobile Information Systems **2017** (2017)

[161] Yoon, K.P., Hwang, C.L.: Multiple attribute decision making: an introduction, vol. 104. Sage publications (1995)

[162] Yuan, E., Tong, J.: Attributed based access control (ABAC) for Web services. In: IEEE International Conference on Web Services (ICWS'05). p. 569 (July 2005). https://doi.org/10.1109/ICWS.2005.25

[163] Zanella, A., Bui, N., Castellani, A., Vangelista, L., Zorzi, M.: Internet of Things for Smart Cities. IEEE Internet of Things Journal **1**(1), 22–32 (Feb 2014). https://doi.org/10.1109/JIOT.2014.2306328

[164] Zhang, G., Parashar, M.: Context-aware dynamic access control for pervasive applications. In: Proceedings of the Communication Networks and Distributed Systems Modeling and Simulation Conference. pp. 21–30 (2004)

[165] Zhang, L.x., Zou, J.s.: Research of ABAC Mechanism Based on the Improved Encryption Algorithm Under Cloud Environment, pp. 463–469. Springer Netherlands, Dordrecht (2015). https://doi.org/10.1007/978-94-017-9618-7_46, http://dx.doi.org/10.1007/978-94-017-9618-7_46

[166] Zhang, X., Parisi-Presicce, F., Sandhu, R., Park, J.: Formal Model and Policy Specification of Usage Control. ACM Trans. Inf. Syst. Secur. **8**(4), 351–387 (Nov 2005). https://doi.org/10.1145/1108906.1108908, http://doi.acm.org/10.1145/1108906.1108908

[167] Zheng, K., Lv, T., Li, Y., Lu, Y.: The analysis and implementation of AllJoyn based thin client communication system with heartbeat function. In: International Conference on Cyberspace Technology (CCT 2014). pp. 1–4 (Nov 2014). https://doi.org/10.1049/cp.2014.1293