Internet of Things Gateway Access Control

Stefanos Plastras

November 2020


ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

Athens University of Economics and Business

School of Information Sciences and Technology
Department of Informatics
Athens, Greece

Master Thesis
in
Computer Science

Internet of Things Gateway Access Control

Stefanos Plastras

Committee: Professor George C. Polyzos (Supervisor)

Professor Vasilios Siris

Professor George Xylomenos

November 2020

Stefanos Plastras

Internet of Things Gateway Access Control

November 2020

Supervisor: Prof. George C. Polyzos

Athens University of Economics and Business

School of Information Sciences and Technology

Department of Informatics

Mobile Multimedia Laboratory

Athens, Greece

Abstract

The Internet of Things (IoT) is a dynamic environment of smart Things that can be connected to the Internet. Of considerable interest is the environment of a smart house, in which heterogeneous Things are connected, which gathers data from the real world and performs actions of interest to end-users. In this scenario, important security issues arise, such as full access to all of the Things of the house by users without distinguishing roles among them.

It becomes clear that there is an urgent need to address how access control policies, and security services more generally, should be defined in the environment of a smart home in order to ensure the authentication and authorization of users in terms of their access to Things.

open Home Automation Bus (openHAB), one of the most widely used software platforms for home automation, can be used to control the operation of smart Things, implement their management and the roles of different users. Authorization based access control can be implemented through *OAuth 2.0* and *JSON Web Token* (JWT).

In this thesis, a presentation and study of openHAB as a smart home automation environment is provided, as well as its integration with modern security standards such as OAuth 2.0 and JWT. We also present our implementation of openHAB in a general IoT Gateway, which manages Things and enforces access control on them in order to provide an appropriate security level.

Furthermore, this thesis studies and documents to what point openHAB can offer authentication and authorization services, as well as the current state of smart home automation solutions in terms of security. Finally, results are provided for the provision and application of present-day safety mechanisms in the openHAB environment as well as ideas for future research.


ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

Περίληψη

Το Διαδίκτυο των Πραγμάτων (Internet of Things—IoT) αποτελεί ένα δυναμικό περιβάλλον από έξυπνες συσκευές ή αντικείμενα που μπορούν να συνδεθούν στο Διαδίκτυο. Σημαντικό ενδιαφέρον παρουσιάζει το περιβάλλον ενός έξυπνου σπιτιού, στο οποίο συνδέονται αντικείμενα τα οποία συγκεντρώνουν δεδομένα από τον πραγματικό κόσμο και εκτελούν διάφορες διαδικασίες εξ ονόματος των τελικών χρηστών. Σε αυτό το σενάριο, προκύπτουν σημαντικά θέματα ασφάλειας όπως η πλήρης πρόσβαση στο σύνολο των αντικειμένων του σπιτιού από έναν χρήστη χωρίς τη διάκριση ρόλων μεταξύ των χρηστών.

Γίνεται ξεκάθαρο, ότι εμφανίζεται επιτακτική ανάγκη να διευθετηθεί το πως θα καθορισθούν πολιτικές και υπηρεσίες ασφάλειας στο περιβάλλον ενός έξυπνου σπιτιού ώστε να διασφαλισθεί η αυθεντικοποίηση και εξουσιοδότηση των χρηστών σε ότι αφορά την πρόσβαση σε αντικείμενα.

Σε αυτή τη κατεύθυνση, θα αξιοποιηθεί, το *open Home Automation Bus* (openHAB), ένα από τα πιο διαδεδομένα εργαλεία ανοιχτού λογισμικού για αυτοματοποίηση έξυπνου σπιτιού, το οποίο θα ελέγχει τη λειτουργία έξυπνων αντικειμένων, τη διαχείριση τους και τον ρόλο διαφορετικών χρηστών μέσω εφαρμογής τεχνολογιών ασφάλειας και ελέγχου πρόσβασης, όπως *OAuth 2.0* και *JSON Web Token* (JWT).

Στην παρούσα Διπλωματική Εργασία γίνεται παρουσίαση και ανάλυση του openHAB ως περιβάλλον αυτοματοποίησης έξυπνου σπιτιού περιλαμβανομένων υπηρεσιών εξουσιοδότησης OAuth 2.0 και ανταλλαγής μηνυμάτων JWT. Επίσης, παρουσιάζεται η υλοποίηση του openHAB σε ένα γενικό IoT Gateway, το οποίο διαχειρίζεται τα αντικείμενα και την πρόσβαση σε αυτά με σκοπό την παροχή ασφάλειας και ελέγχου πρόσβασης.

Ακόμη, η εργασία αυτή μελετά και καταγράφει σε ποιο βαθμό το openHAB μπορεί να προσφέρει υπηρεσίες αυθεντικοποίησης και εξουσιοδότησης, αλλά και τη τρέχουσα κατάσταση ήδη υπαρχόντων λύσεων αυτοματοποίησης έξυπνου σπιτιού υπό το πρίσμα της ασφάλειας. Τελικά, χρήσιμα αποτελέσματα εξάγονται και αξιολογούνται για την

παροχή και εφαρμογή των μηχανισμών ασφαλείας στο περιβάλλον openHAB και παρουσιάζονται ιδέες για μελλοντική έρευνα.

Acknowledgement

First of all, I would like to thank my supervisor, Professor George C. Polyzos for his continuous and valuable support for both the preparation of this thesis and in my studies in general. Furthermore, I would like to express my sincere appreciation and gratitude to Dr. Nikos Fotiou, PostDoctoral Researcher at the Mobile and Multimedia Lab (MMlab), for technical guidance and knowledge he provided me that allowed me to successfully complete this work.

In addition, I would like to thank the PhD candidates of the MMlab, Spyros Chadoulos and Iakovos Pittaras for their technical advice. Special thanks go to my friend and fellow student Vasilis Kostantinou, for the interesting and beneficial discussions we had.

Last but not least, I would like to thank my family and especially my parents for their encouragement, support and help they provided me all this time.


ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

Contents

1 Introduction

Today's society has witnessed a huge technological development, that of the Internet of Things (IoT). IoT is a dynamic environment that tries to reconcile the physical with the virtual world into a single one. In this environment, heterogeneous Things of different specification, technology and intelligence can interact, generate and exchange data with each other, resulting in the expansion of new applications and services beyond the known Internet [SQO19]. The term IoT was first studied in 1999 by Kevin Ashton in the effort to integrate RFID technology into the Internet and began as a simple scenario before reaching to create new perspectives in the modern technological world [Sob20]. However, one feasible definition for this technology is: "A dynamic global network infrastructure with self-configuring capabilities based on standard and interoperable communication protocols where physical and virtual 'Things' have identities, physical attributes and virtual personalities and use intelligent interfaces, and are seamlessly integrated into the information network" [SQO19]. It is easily perceived that IoT is now receiving quite a lot of interest because of its ability to disperse billions of real world Things together. Therefore, the role of IoT can be applied to a wide range of categories in people's lives from smart cities, smart health, smart grids and smart agriculture to smart homes.

One of the IoT application domains is the smart home, an area that has received great interest from users due to the fact that automation of many daily operations in a home can save time and energy. Moreover, the complexity of the devices and the different functions that the various appliances serve, need to be coordinated under a smart home framework [WTI19]. Currently, to achieve the interconnection of all devices and the easier control and management of them, a special device called IoT Gateway is used. Usually this device integrates different connectivity technologies and protocols such as BLE, WiFi, Ethernet, ZigBee and provides a set of user-friendly interfaces to configure, control and manage the smart home, IoT Gateway and devices. An important role is played by smart home automation frameworks where they manage Things, the data they generate and interact with the user by offering user-oriented interfaces and services. Consequently, many smart home automation frameworks rely on this architecture and put their effort in a IoT Gateway allowing it to execute all the necessary tasks. Therefore, it is the responsibility of the framework to provide a set of security policies in a smart home which users have to consider [Chi+19]. For example, in a set of devices who is going to have unlimited access, who is going to have restricted access and who user or service has the role to delegate the necessary permissions for these accesses? As a consequence, severe security issues arise as access to smart home devices and services is determined through the framework

in general, which leads to a serious dependence of its capabilities and security policies, thus leading to great challenges that need to be solved.

Currently, each framework must offer basic home security services, covering the entire range of users who are going to interact with it having distinct roles. At that moment, there are many who fail to cover security services [El-+19], thus failing to provide appropriate policies. Firstly, in authentication level, the appropriate policies that will determine whether a user actually claims to be who he has stated that he is are not supported. Then, in authorization level, no policies are defined on the basis of which users can be categorized (grouped) and depending on the category to assign them the appropriate access rights to the home (devices, services) [MH19]. It becomes clear that, the current problem lies in the absence of access control policies from the core of many frameworks and as a result it makes it clear that security techniques that currently meet these requirements should be studied and integrated into them with the aim of ultimately providing frameworks that will define appropriate access control policies utilizing interoperable and state of the art technologies.

To deal with the aforementioned challenges and limitations that come with the Smart Home frameworks, we leverage technologies that are widely adopted by academia and industry and provide upgraded authentication and authorization services to a multitude of Internet applications. That is the authentication and authorization service OAuth 2.0 [Int12], an open standard for access delegation, commonly used as a way for Internet users to grant websites or applications access to their information on other websites but without giving them the password. Today, it is already used in many protocols and services like Google, Facebook and Amazon [Wik20b]. It can therefore support the requirements of a smart home framework in these areas. In addition, Json Web Token (JWT) [Int15], a data exchange technology between client and server, can help us to reliably and flexibly transfer data based on simplicity, speed and security as all information is digitally signed. JWT as an Internet standard processes data in json format and in OAuth 2.0 is being used in a wide range of today Internet applications [Wik20a]. Finally, these standards are open and generalized as any operator, technology and application can utilize as they address amongst other the issue of interoperability.

In this thesis, we study open Home Automation Bus (openHAB), an open source smart home automation framework based on Eclipse Smart Home platform and written in Java [ope20c]. It can leverage many different technologies and integrate many manufacturers devices in order to provide a global service to it's users. In this point, user management, authorization and authentication are highly important concepts for such a framework due to openHAB inability to address security in these aspects. We propose a design that incorporates the above- mentioned technologies into openHAB framework providing access control mechanism and a generic smart home security framework that each automation technology could utilize. To achieve that we propose an OAuth 2.0-based architecture that allows

users to have classified access to the home in a flexible manner using access tokens. Our architecture has many advantages compared to other known smart home security solutions, such as interoperability, scalability and generalization.

1.1 Motivation and Problem Statement

It is quite widespread that security gaps in IoT have significant implications at smart home level. Some of the potential attacks may be data loss, unauthorized access, or even causing unwanted actions to the entire home (e.g., unwanted access on thermal devices could burn the entire home). The fact is, in the context of openHAB, that is unclear what impact a potential security breach could have at any level.

Currently, the set of applications for IoT is quite recent and this implies that most frameworks for smart homes do most of the work mainly on the level of functionality rather than security. But at the moment, there is not a global tool for assessing security in IoT applications. As the technologies that frameworks integrate become more and more complex, so will the range of vulnerabilities and attacks expand.

Therefore, not fully understanding what openHAB can cover in terms of security, questions may be raised about its acceptance by the community of smart houses but also concerns can be raised about the existing runtimes. So, all these motivates us to review the existing security models and frameworks for smart homes including openHAB application. Up to this point, the research questions of this thesis needed to be addressed are:

- **Is there a security model that covers smart home frameworks in their entirety and up to what point it has been studied, researched, and implemented?**

- **How modern security models could be combined on smart home frameworks?**

openHAB core, presents an unclear landscape in terms of security, and privacy as well as it is not well defined what it can offer. There is no official update that integrates access control policies or implements authorization models [ope20c]. It is of great interest and challenge to have a record or study of what security models it uses and what vulnerabilities may appear in the future. Still, it is difficult to have an indication of the active evolution of security due to openHAB being an open source software [ope20b].

1.2 Thesis Structure

Chapter 2

In this chapter, we mention the necessary background information and technologies used in our work. This chapter is divided into three subsections. The first one stands for the IoT (see 2.1), the second for openHAB Framework (see 2.2) and the third for access control mechanisms (see 2.3).

Chapter 3

In chapter 3, related work is presented. Initially, we introduce frameworks similar to openHAB smart home framework (see 3.1), then we examine openHAB security under the research scope (see 3.2), and finally we present current smart home access control implementations (see 3.3).

Chapter 4

In chapter 4, we first introduce the security requirements of our proposed design (see 4.1). Then, we present our OAuth 2.0-based access control architecture (see 4.2). Afterwards, we analyze the implementation of the system, the components it consists of, the way we built it, and the tools we used (see 4.3).

Chapter 5

In chapter 5, we proceed to the system evaluation. We use specific metrics (access level, execution times, etc.) and we try to understand what parts of the system are critical and why, and what parts do not play major role. Finally, we examine future improvements that could be applied in the system to improve the overall performance and security services of it. In 5.1, we present some experiments measuring execution times on our system. In 5.2, a simple security scenario is presented without our proposed design as well as the same scenario with our solution applied in order to see the benefits of our system.

Chapter 6

Finally, this chapter summarizes our work.

2 Background

In this chapter, we study the theoretical background information and technologies that we use to implement the system. Background consists of three parts. In the first part is the Internet of Things (IoT) background, which we study Thing as an entity in IoT system, characteristics and requirements in smart home environments and the role of the gateway device. In the second part, we focus on openHAB, which is an open source smart home automation framework, its features and its security. Then, in the last part we study various access control mechanisms that are already being applied in Internet technologies.

2.1 Internet of Things

IoT is a new architecture of networking Things, buildings, large and small machines and people, in such a way that they communicate and interact with each other, usually with some level of automation, collecting and sending data through a common network. Technologies for connecting to IoT can be various such as Wi-Fi, GSM, 3G, 4G-LTE, 5G, Bluetooth, fixed networks etc. Initially the idea was started with the purpose of interconnecting industrial equipment, but then this range was expanded, resulting today in the interconnected devices to include everyday Things (for example home appliances).

Today, IoT is getting bigger and bigger [BNS19] as there are already more connected Things than people in the world. The more and more increasing volume of data managed by networks is estimated to reach 35 Exabytes in 2020. Also, the number of IoT devices, and the data derived from them, is expected to increase exponentially, as the number of machines will exceed mobile ones [Kou20]. Apart from smart homes, cities, healthcare and agriculure, it has resulted in the massive evolution of Industrial Internet of Things (IIoT) or else the fourth industrial revolution. The concept is the same for IoT consumers but uses a combination of sensors, networks, big data and artifical intelligence (AI) to optimize industrial process.

The IoT makes an effort to make our environment, homes, offices, vehicles smarter, more measurable. There are IoT technologies such as Amazon's Echo and Google Home which make it easy to control, manage and act on every aspect of environment whether is a smart home or an office.

2.1.1 Thing

Things can be defined as an entity, an idea, a quantity or even as an abstract being in the world. They are usually at the lowest level in the IoT stack as they are tasked with collecting or generating data. In this context, we have the Smart Things which actually constitute a set of devices or services that are interconnected to be accessible via a computer network or even via the Internet [Lan+20]. These are autonomous physical or digital objects endowed with networking, sensing, processing and intelligence capabilities. Physical Things exist in the physical world and are capable of being sensed, actuated and connected. Examples are the electrical equipment such a a digital thermal sensor. Instead, digital or virtual Things exist in the information world and are capable of being stored, processed and accessed. Examples include HTTP services, or APIs. Smart Things are characterized by the ability to perform simultaneous applications supporting different levels of mobility and personalization as provide customization capability under the control of individual users.

But, as the IoT environments and especially business models become increasingly complex, Smart Things are expected to be enriched with further capabilities to bridge the gap between the physical and virtual worlds by offering new services such as decentralized information processing, decision making, independent execution of complete business processes [Lan+20].

2.1.2 Smart Home Environment

The smart home has been described as a single entity consisting of devices, software, interfaces, connectivity capabilities and which interacts with either external entities or users providing home management functions. The term smart home can have similar names such as automated homes, unified home systems, electronic homes, connected homes. In traditional homes, operations on objects are done manually plus the user has no information about the state of the object or its energy consumption. With the appearance of the house, these limitations are exceeded since the user interacts with the object through an interface directly without direct access to the device itself as he has digital access to the state of the object and has knowledge of information concerning it such as energy consumption [FOS20]. This idea originated in the 1930s.

A technical definition on what is a smart home, can be defined as: *"A smart home can be described by a house which is equipped with smart objects, a home network makes it possible to transport information between objects and a residential gateway to connect the smart home to the outside Internet world. Smart objects make it possible to interact with inhabitants or to observe them"* [Ric+06].

- **Architecture**
  Smart home gathers data from a group of devices and sends them to server back-end for some kind of process. This process creates a pipeline model in which there is a data source, a smart home local gateway capable of receiving all these data volume and sending them to a remote cloud server for data process. This concept proposes a smart home general cloud-based architecture, as shown in figure 2.1. But the architecture can be rearranged depending on the level at which the data processing is carried out. The data processing models are located at cloud level and in home level, at gateway and the various smart home devices. The internal network of

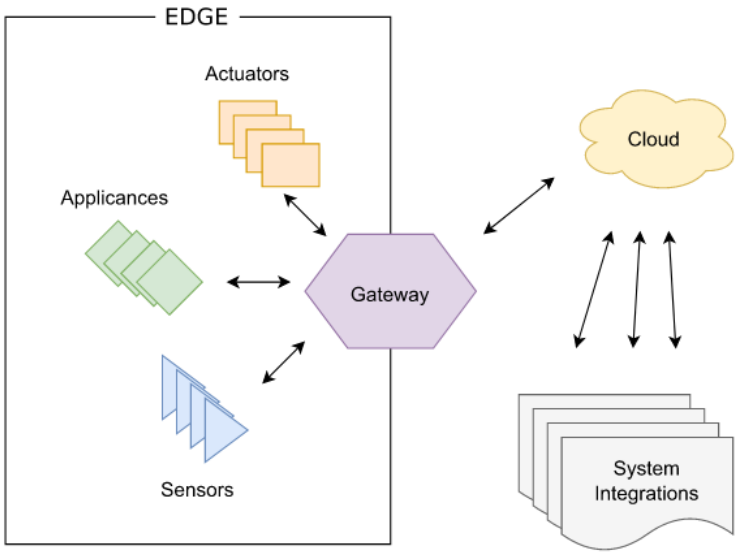


Fig. 2.1: Cloud-based smart home architecture [MCM18]

the home consists of a multitude of devices, sensors, household appliances. These devices communicate with a gateway located at the edge of the network, providing a connection between the internal network and the Internet. Gateway's role is to support the integration of heterogeneous devices into the home space as well as their connectivity for interoperability with all devices [AO19]. But, gateway can contribute to data processing partially before the cloud but also completely bypassing the cloud. As a result, three schemes derive from the general architecture, edge, cloud and fog computing.

1. Cloud computing
   Cloud computing technology essentially gathers all the data generated by the end devices in a home and with appropriate algorithms and infrastructures such as memory, network and processors, processes this data and produces useful results for the end user. Usually resources in a cloud environment are requested on demand through a pool of manageable resources. This is the last stage in architecture in which the data is processed.

2. Fog computing
   Fog computing is the technique in which the cloud computing model expands and part of the data processing now occurs at the level of the smart home and specifically in gateway device. This is a recent technology studied by Cisco [CIS15] in 2014. But the data generated by a smart home continues to grow, and so more data flows to the cloud resulting in longer response time and greater network bandwidth being spent. The Fog model overcomes these issues and transfers some of the processing from cloud to gateway by saving on time and network resources. The gateway is closer to the data, which is why this technique manages to cope with the above issues. Another important element is that the Fog model solves the single point of failure (SPOF), since if the cloud is damaged, the data processing will continue to exist in the home gateway.

3. Edge computing
   The Edge computing model is another example of data computing that is similar to the Fog computing model. Now the data processing leaves the gateway and is transferred to every smart device that has computing resources and which has the responsibility to decide whether to keep the data to be processed or whether to send the data more centrally, to the Cloud.

- **Application Domains**
  The smart home provides a wide category of applications to users as research has advanced and discovered new requirements in this space [FOS20]. The applications in general, offer comfort, well-being, energy efficient and security services. Three main application areas are energy efficiency and management, entertainment and security.

  Usually energy consumption is spent in homes as a terminal in the flow of electricity. It is understood that the requirements for electricity are increasing as the houses are growing. Thus, the main goals are to reduce the cost of energy for homes without degrading the quality of life and comfort. The functions of smart home energy management are the control of household appliances, real-time control of the energy consumption of the house, the provision of suitable interfaces for monitoring. One proposed technology to achieve these goals is smart energy grids [Dil20].

  The purpose of entertainment apps is to provide a feeling of joy and relaxation for home users by giving them the ability to manage media players digitally in an automatic way. The benefits of using such apps are to make everyday life easier, access to multimedia home content from any place [FOS20].

  The multitude of applications and technologies present in the smart home introduce some complexity, which creates gaps in its security. By security is meant the de-

tection of malicious acts, unauthorized access, data interception. There should be security techniques covering concepts such as authentication, user authorization, data encryption and Privacy Protection [FOS20].

- **Advantages**
  As an important trend in people's daily lives, it offers undeniable advantages.

  Managing all devices from a single point is a great advantage. It is possible to manage through a single interface from which any can configure, control, enable, disable, customize each device. Thus, each new user does not need to learn new technologies, but simply have access to this interface.

  Another important benefit is cost and energy savings. Smart House has modules and software such that it learns which devices are not in active use and thus reduces energy consumption. Moreover, with the provision of statistics on energy consumption, it is easy to adjust the home settings to a more cost-saving way.

  Flexibility is a term offered by a smart home and enables the addition of new devices, often of different specifications and technologies. It doesn't matter if the device is old or state-of-the-art, as the home implements integration function seamlessly.

2.1.3 Gateway

Gateway is considered as a connection device or service between Smart Things and applications in an IoT environment. Gateways come in several forms and can be physical devices such as mobiles phones, specialised IoT hardware, generic devices (eg, rasberry pi platform) or a piece of software running in a service or a process modeling a gateway role. A basic distinction between gateways can be either a Basic or a Smart Gateway. In the first type, gateway is an entity that has specific purpose of gathering data and processing them locally. Smart Gateways, expand this model, and they can send aggregated data from local to cloud storage offering some intelligence.

- **Functionalities**
  An IoT Gateway must support different functionalities that exist in almost every environment such as smart homes, cities.

  1. Edge computing
     Edge computing is an expansion of cloud computing. Tasks that are normally executed in the cloud are brought to the network edge reducing latency [TM19]. Gateway devices are usually more powerful than Thing devices and can execute demanding tasks that devices do not have the resources needed.

2. Dataflow control
   Gateways may have many communication interfaces, supporting concurrent communication with devices and cloud services. For example, one interface could be WiFi for connection with devices and the other a LTE for connection with a cloud server. Also, in scenarios which gateway relays information between sensors and sensor networks, it is crucial to support data aggregation in order to save energy by filtering data before sending [KV19].

3. Proxy and caching
   Proxy is a service that acts as an intermediary for client requests seeking resources from a resource server. Proxy is useful in cases clients should not access physical devices directly. Consequently, reduction of response time, network bandwidth and energy consumption may be drawbacks but certainly there are important benefits to use gateway as a proxy server. Proxy can support caching, saving this way energy in the gateway [KV19]. Furthermore, security features can be applied and authentication could be achieved as well as data encryption with protocols such as HTTPS or TLS.

4. Resource discovery
   Resource discovery is quite important service in IoT applications as there are multitude of different types of Things and devices. This process can be done in different layers of the network depending on the network size and type. But, Things should somehow be able to expose their resource, so that a client through a gateway with resource discovery will be able to know about them.

- **Management**
  As IoT Things, gateways need to be managed. There are various reasons for this, security updates, new functional fixes, device settings. Thus, gateway is a device that has resources such as processor(s), memory, network bandwidth and as a result these resources need to be monitored. MQTT is a protocol that brings interesting perspective to gateway management using a publish-subscribe model [KCR15]. IoT Gateway can be used as a broker node to forward control messages to different IoT devices.

- **Architecture**

1. Software architecture
   Operating system (OS) is the first level that support a gateway [TM19]. An operating system that many gateways use is Linux. Then, upon an operating system, gateways can run application code on a run-time environment. Moreover, a gateway needs connectivity features, both wireless and wired to communicate with devices and servers. At this stage, gateways are ready to gather data

from devices and forward processed data to cloud or data centers. Remote management provides a channel to establish communication with gateway to configure, update, change settings or applications. And security manager, offers security features on data and device management and connections. The software IoT Gateway architecture, as shown in 2.2.

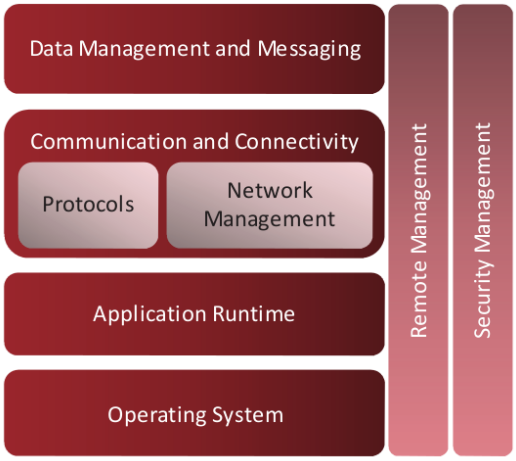


Fig. 2.2: IoT Gateway architecture [KV19]

2. Communication Stack
   Gateway communication requirements cover a range from physical to application layer. That means there exist a wide range of technologies and protocols available for this purpose [KV19]. In the application layer, HTTP protocol can be mapped to RESTful APIs. Constrained Application Protocol (CoAP) is an Internet application protocol and is a binary version of HTTP and uses REST structure. Websocket is another communication technology with the same addresing procedure as HTTP. It is recommended for real time applications. Message Queuing Telemetry Transport (MQTT) is a protocol that is based on publish-subscribe model over TCP/IP stack. Advanced Message Queuing Protocol (AMQP) refers to topic-based publish-subscribe messaging and routing. And Extensible Messaging and Presence Protocol (XMPP), an XML based protocol better suitable to structured data and devices.

   In transport layer, Transmission Control Protocol (TCP) and User Datagram Protocol (UDP) are used. UDP is preferable in a constrained environment due to smaller overhead it introduces in connection.

   In network layer, IPv4 and IPv6 are the main protocols for communication. Another protocol is IPv6 over low power wireless personal area networks (6LoWPAN), a special IPv6 protocol for wireless environments. Still, IPv6 Routing Protocol (RPL) can be used for routing purposes in IPv6 networks.

In physical layer, there are many communication technologies. Some of them are Bluetooth, Bluetooth Low Energy (BLE), Wi-Fi, ZigBee, Z-wave and cellular connectives such as 3G, 4G. The gateway communication stack, as shown in 2.3

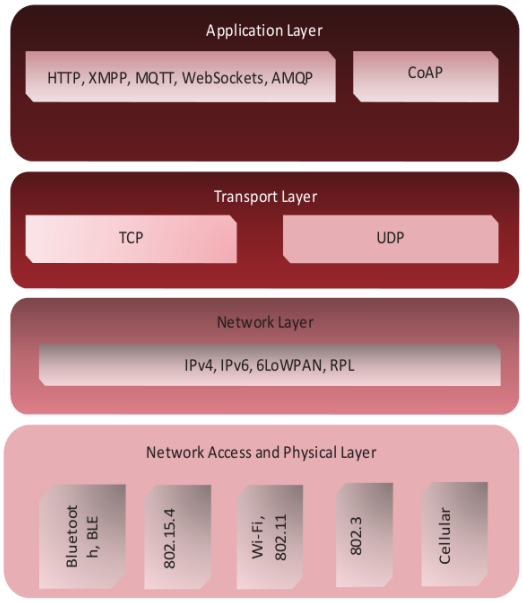


Fig. 2.3: IoT Gateway communication stack [KV19]

- **Design Requirements**
  Every IoT Gateway must provide basic services and functionality in IoT environments. Consequently, there are several challenges to design a gateway [KV19].

  1. Communication
     Communication requirements must combine different scenarios. There are environments that are bound by low power devices restrictions but also environments that instead require a high level of speed. So we need gateway to provide connectivity services to both these devices and high-end environments like cloud. Still the IoT space consists of a mass of device connections, and there should be support to obtain an address and connect to the gateway [AO19]. There should be compatibility with IPv4, IPv6 protocols but supported whenever switching between them is required.

  2. Security
     Security is an area that appears in every aspect of an IoT system. The level of security that a gateway must ensure is that of the protection and integrity of the data transferred to and from the gateway and the information that resides in the device itself. The methods of counteraction in this case are the encryption techniques applied to the data. Furthermore, gateway communica-

tion with other entities, such as servers, clients must be authenticated so that communication between them is secure and valid [KV19].

3. Intelligence
   In a typical scenario, the data collected from a multitude of sensors and devices is gathered in the gateway and it forwards it to a cloud server to start processing. A serious disadvantage is the introduction of time delay until the data reaches the cloud and is processed, a parameter that depends on the individual network. The introduction of intelligence, i.e. software capable of analyzing and processing data and making decisions on demand, will reduce the volume of information flowing to the cloud, thus saving time and network resources.

4. Scalability
   It is a challenge to add devices on demand to an IoT environment and manage them as they increase. We need a hierarchy in a set of gateways and a process of coordinating them.

2.2 openHAB Framework

openHAB [ope20a] is a smart home framework that focuses on interoperability among all kinds of devices from different manufactures (vendors). It handles inter-device interaction through logical modules called bindings. For example, a smart plug from Xiaomi may not be able to interact with other devices out of box, but it may be able to do so if the appropriate binding is developed for the openHAB ecosystem. openHAB, as it contains the word open, is an open-source software [ope20b] that serves as a gateway that brings together a diverse range of devices through the use of bindings. A binding makes a link to a Thing that may be of either physical or logical (virtual) nature. For instance, a smart plug switch is a physical Thing, and its state of being turned on or off is part of the data it exposes. However, a weather service from the Internet that provides weather information such as temperature, humidity, and precipitation probability is a logical Thing. Thus, a binding may include such a weather service as a Thing in the logical sense.
A feature of openHAB is that it can remain functional and provide services without an Internet connection [ope20a]. Generally, Things are connected to openHAB whenever it is appropriate and not to a cloud service. This indicates openHAB attempt as a platform to solve vendor interoperability issues. However, this turns into a limitation, in cases where independent openHAB runtime environments are required to communicate with each other to co-operate when data is gathered separately.

As for openHAB framework versions, openHAB currently supports openHAB v1 [ope18b], which is still receiving support but gradually fewer and fewer users and contributors

support it, and the latest version v2.5.9 [ope18a], which was released on 21 of September 2020.

The core functionality of openHAB project is made of Eclipse SmartHome framework [ECL], a framework that allows building smart home solutions that have a strong focus on heterogeneous environments, i.e. solutions that deal with the integration of different protocols or standards. Its purpose is to provide a uniform access to devices and information and to facilitate different kinds of interactions with them. This framework consists out of a set of Open Services Gateway initiative (OSGi) bundles that can be deployed on an OSGi runtime and which defines OSGi services as extension points. OSGi technology [OSG] is composed of a set of specifications, a reference implementation for each specification, and a set of compliance tests for each specification that together define a dynamic module system for Java. OSGi provides a vendor-independent, standards-based approach to modularizing Java software applications and infrastructure.

2.2.1 openHAB Architecture

The overall architecture of openHAB as shown in the figure 2.4, it is developed in Java and its core is the Eclipse SmartHome framework. By using Apache Karaf and Eclipse Equinox it instantiates an OSGi runtime environment. Jetty is deployed as the HTTP server where resources are exposed through the HTTP protocol. openHAB is highly extensible through the use of Add-ons. Add-ons and extensions may be used to extend the existing user interface, or to provide the communication bridge with physical devices (bindings). Add-ons from the previous version may be used through a special module that performs the necessary adjustments.
As the core for the openHAB software stack, Eclipse SmartHome (ESH) provides a flexible and modularized framework for smart home and ambient assisted living solutions with a focus on heterogenous environments. The goal of ESH is to offer a solution for the fragmented market in smart home solutions by offering a medium where vendor-incompatible devices can be operated transparently. As in the first version of openHAB, the idea of a binding is reused in ESH. Bindings implement a Thing-specific protocol e.g. ZigBee, and by loading the appropriate binding, the connection between the devices or services e.g., a TV set or weather service, and the framework, can be established. Furthermore, the bindings are connected to an event bus, and this allows to do inter-component communication. Through the event bus, it is possible to send commands to the device or service, or else to receive status updates from them.
The architecture of ESH as shown in Figure 2.5. In the figure, at least four Things are connected to the ESH framework through its respective bindings. There are two ways in which activity prompts changes in the Things connected: through a user interface (UI) and automation logic. The UI in ESH is made up of adaptable sitemaps comprised of dynamic web pages, and this medium allows for interaction with the Things through a web

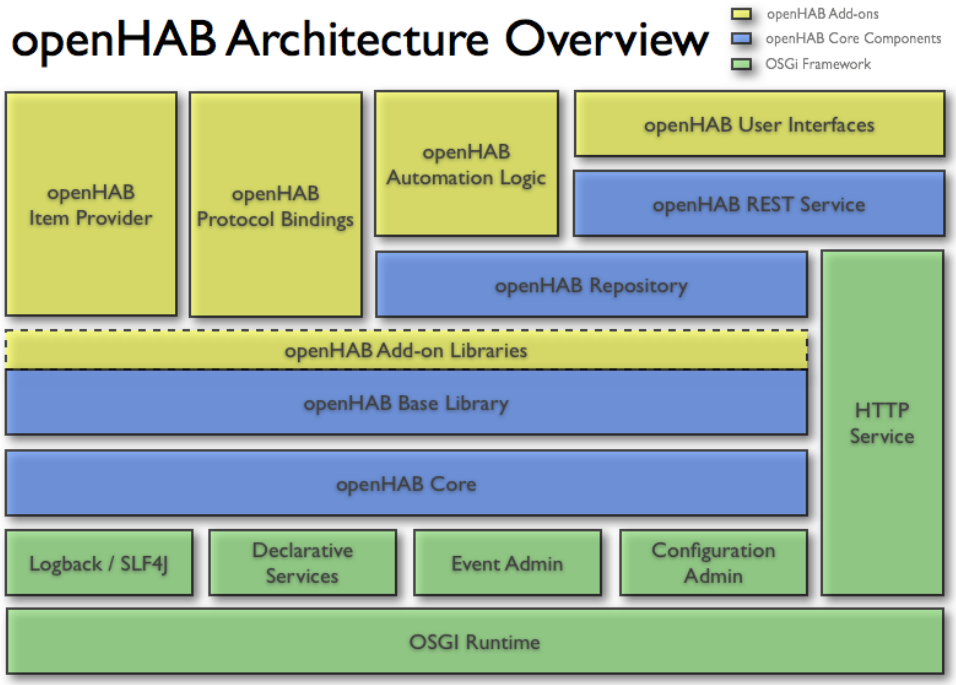


Fig. 2.4: openHAB Architecture [ope18b]

browser. The automation logic is established in terms of rules, which are specified through configuration files. Additionally, there is also a REST API which is exposed through certain URIs, not shown in this figure. For all cases, commands may be directed to the event bus, which are then forwarded to its respective binding. Likewise, status updates may be shown in the UI, or served to the automation logic to perform more operations on top of these results. Finally, it is possible to make use of a custom logging module and console, mostly for development purposes.

2.2.2 openHAB Security

In general, openHAB presents different scenarios about communication with Things. There is the internal part which is interaction between openHAB and Thing, and external part which is interaction between openHAB, Thing and a remote service. But, for every scenario, security requirements are different.
Internal communication between openHAB and Thing(s) is heavily depended on the network security. That means that the home will be under a wireless network and therefore encryption protocols should be applied such as Advanced Encryption Standard (AES). An eavesdropper would be able to get the data only if he could break AES security.
External communication indicates communication with an external point such as cloud, exposing vulnerability within the Internet. An eavesdropper could attack data at the point data have transferred from openHAB to cloud. HTTPS protocol could be applied to prevent data leak, securing data through Internet.

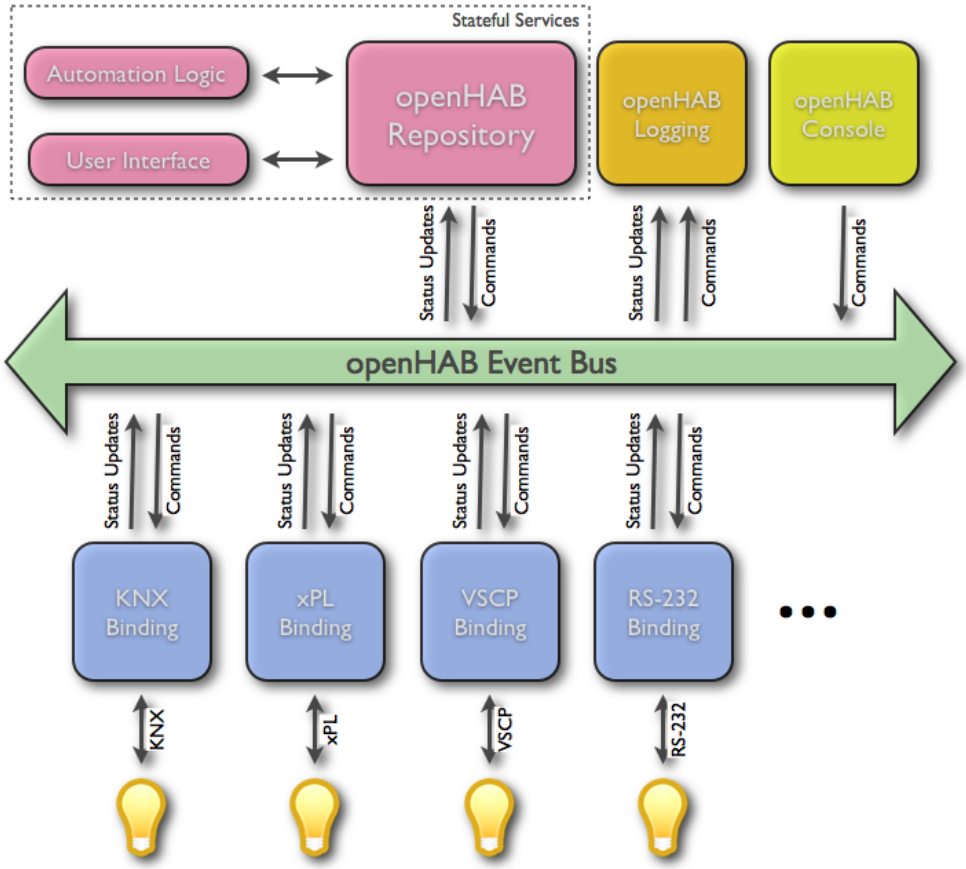


Fig. 2.5: Eclipse SmartHome Architecture [ope18b]

Every smart home framework should be accessible from in or out of home (via Internet) as this is basic functionality. Exposing openHAB to the outside is risky enough due to denial-of-service attacks, unrestricted Uniform Resource Locators (URLs) and session hijacking and many other attacks that could cause malfunction to openHAB service. However, to preserve privacy and security it is recommended to keep openHAB Gateway closed and inaccessible from Internet. May, this security option limits use cases for the smart home but satisfy security requirements. Consequently, remote access to openHAB is challenge that requires security options ensuring certain criteria. Official openHAB organization and documentation defines three main options for ensuring secure remote access [ope20c]. In this documentation are also described the weaknesses that openHAB is subject to and points that cannot support in term of security. Secure remote access options:

1. Virtual Private Network (VPN) Connection
   There are many different VPN technologies that one could utilize. A VPN network provides data encryption and authentication services.

2. myopenHAB Cloud Service
   myopenHAB Cloud Service [ope20e] runs on cloud infrastructure creating a tunnel connection and forwarding all requests through this tunnel. openHAB will see these

incoming requests as originating from the local loopback interface. It is a free service but it cannot offer any Service Level Agreement (SLA)s regarding availability.

3. Running openHAB Behind a Reverse Proxy
   A reverse proxy simply directs client requests to the appropriate server. Authentication and encryption services are being offered.

Up to this point, openHAB does not (yet) support restricting access through HTTP(S) for certain users - there is no authentication in place, nor is there a limitation of functionality or information that different users can access.

2.3 Access Control

Information security can be defined as the ability to preserve the integrity, confidentiality, authenticity and availability of a system's resources and data in all phases, storage, processing, transmission. Therefore, every Information System and digital environment should incorporate a security model in order to remain functional and provide the required services to users. But security at any level can be compromised if the appropriate threat is taken advantage of. It is therefore crucial to have a security model with proper policies in place to prevent possible security breaches in any Information environment.

An important part of security is protection against unauthorized access, invalid modification of information or resources, and at the same time ensuring access of authorized entities. The process of enforcing measures to ensure access to resources under a set of security policies is known as access control [Oua+17]. Access control solves the issue of who has access to what, when, and in which conditions. An access control mechanism is divided in three parts, security policy, authentication and authorization model.

Security policy High-level rules that define which resources and data should be regulated and to what degree.

Authorization model Abstract representation of the security policy in terms of the rules and resources present in a computer system.

Authorization mechanism Software and hardware implementation of the functions that enforce the security policy through the abstraction provided by the authorization model.

The access to resources and data for a particular security policy is specified through the establishment of an access control model. However, the model itself does not execute the policy, and for that, enforcement is needed. Enforcement takes the form of the im-

plementation of technical security mechanisms, such as credentials, digital signatures, encryption, access control lists, firewalls. An access control mechanism, should permit the use of flexible verification methods.

2.3.1 Authentication

Before the authorization stage, the identity of the user requesting access to a resource or data should be verified. After authentication, the authorization model can determine whether the user really needs to access or not according to the specified policies. Ways to authenticate the user fall into three categories [Vel18], depending on what information they know, who they are, or what they own.

- **Password authentication**
  The user provides a name or an ID along with a password and the system uses them to authenticate (verify) if a matching user exists.

- **Biometric authentication**
  Based on an user's unique physical characteristics, such as fingerprints, facial characteristics, eye characteristics.

- **Token authentication**
  A physical or virtual (digital) object that the user posseses for authentication. For example, smart cards or digital tokens.

Authentication also happened in Web and Internet in a wider scope [El-+19]. Different and heterogeneous entities, software and processes try to communicate and exchange data in a secure way. Protocols such as HTTP, HTTPS are used between client and server [Vel18]. Specifically, an HTTP request includes Authorization header containing authorization information representing user credentials. Basic, bearer are formats allowed for Authorization header. Still, JWT is similar to bearer scheme.

- **Basic Authentication**
  Basic access authentication is a method for an HTTP user agent (e.g. a web browser) to provide a user name and password when making a request. Is the simplest technique for enforcing access controls to web resources because it does not require cookies, session identifiers, or login pages.

- **Bearer Token**
  A security token with the property that any party in possession of the token (a "bearer") can use the token in any way that any other party in possession of it can.

- **JSON Web Token**
  JavaScript Object Notation (JSON) is a standard text-based format for representing structured data based on JavaScript object syntax. It is commonly used for transmitting data in web applications (e.g., sending some data from the server to the client, so it can be displayed on a web page, or vice versa). [Wik20a]. A jwt is divided in three parts: header, payload, and signature. The payload includes claims about user identity. To preserve the authenticity and integrity of the payload, a digital signature from the token issuer is attached. The header simply includes the type of the token and the algorithm used for the signature.

Currently, many Internet standards and technologies are integrated and as a result, need for an open technology for authentication and authorization has raised. OAuth 2.0 is an open standard for access delegation, commonly used as a way for Internet users to grant websites or applications access to their information on other websites but without giving them the passwords. This mechanism is used by companies such as Amazon, Google, Facebook, Microsoft and Twitter to permit the users to share information about their accounts with third party applications or websites [Wik20b].
Generally, OAuth 2.0 provides clients a "secure delegated access" to server resources on behalf of a resource owner. It specifies a process for resource owners to authorize third-party access to their server resources without sharing their credentials. Designed specifically to work with Hypertext Transfer Protocol (HTTP), OAuth 2.0 essentially allows access tokens to be issued to third-party clients by an authorization server, with the approval of the resource owner. The third party then uses the access token to access the protected resources hosted by the resource server [OK19].

2.3.2 Authorization Models

Authorization is a model that implements a security policy and interprets it well defined rules executed by a computer. Authorization models categories are:

- **Discretionary Access Control (DAC)**
  Discretionary access control (DAC) [Oua+17]is a type of access control restricting access to objects based on the identity of subjects and/or groups to which they belong. The controls are discretionary in the sense that a subject with a certain access permission is capable of passing that permission (perhaps indirectly) on to any other subject.

- **Mandatory Access Control (MAC)**
  Mandatory access control (MAC) is an access control that restricts the ability resource owners have to grant or deny access to resources in a system. Rules are defined by

the system administrator, strictly enforced by the operating system (OS) and cannot be modified by users.

- **Role-based Access Control (RBAC)**
  Role-based access control (RBAC) [Oua+17] is a concept of assigning permissions to users based on their role within an organization. It provides a clean, manageable approach to access management that is more tolerant to error than assigning permissions to users individually.

- **Attribute-based Access Control (ABAC)**
  Attribute-based access control restricts access to server service depending on the attributes and credentials that the requesting client discloses to the server. The policy regulating access to services is therefore defined over attributes and credentials provided by clients [Oua+17].

3 Related Work

openHAB is one of many existing smart home solutions. In this section, we present the current state of smart home automation frameworks that are available to market and we focus on security mechanisms they offer as well as the weaknesses they have. Thus, we examine the current academic research on openHAB in terms of security. Finally, we examine current smart home access control implementations based on academic research and present an overall state.

3.1 Smart Home Frameworks

- **Google Nest**
  Google Nest [Nes20]is a brand of Google LLC used to market smart home products including smart speakers, smart displays, streaming devices, thermostats, smoke detectors, routers and security systems including smart doorbells, cameras and smart locks. The Nest brand name was originally owned by Nest Labs in 2010 when later Google acquired Nest Labs in 2014. Nest comes with Hub Max, a gateway-like device which works with thousands of smart home devices, like lights, TVs, and locks, providing easy control on them all from one place.

  The Nest API uses the OAuth 2.0 protocol for authentication and authorization [est20]. Before a product can access private data using the Nest API, it must obtain an access token that grants access to that API. Nest handles the user authentication, session selection, and user consent. Authorization realized through permissions groups. Each entity in a Nest account has a single "owner." It is possible for a family account to have different owners for different entities. Owners have full control of all devices, settings, subscriptions, and Works with Nest connections in the structure. Owners can invite other people to share control of the Nest products in their home. Each person with access has a separate Nest account, and can control Nest devices in the owner's home. They can do almost everything that the owner can. Also, HTTPS is supported for REST API.

- **SmartThings**
  SmartThings is an American smart home company founded in 2011. It focuses on the development of eponymous automation software and an associated array of

client applications and cloud platforms for smart homes and the consumer Internet of Things. Since August 2014 SmartThings has been a subsidiary of Samsung Electronics [Sma20a]. SmartThings ecosystem includes devices connection to SmartThings through a third-party cloud, directly to the SmartThings Cloud, or with a SmartThings-compatible hub. Automations are WebHook or AWS Lambda functions that allow a user to control their SmartThings ecosystem without manual intervention. The SmartThings app is used to configure and control Automations and IoT devices in the SmartThings catalog and provides REST APIs that enable the integration, control and monitor IoT devices and services on SmartThings Cloud.

All SmartThings resources are protected with OAuth 2.0 Bearer Tokens [Sma20b]. Permissions that user select will define the OAuth 2.0 scopes for the token. Specifically:

-Personal access token scopes are associated with the specific permissions selected when creating them.

-SmartApp token scopes are derived from the permissions requested by the SmartApp and granted by the end-user during installation.

- **AWS IoT - Connected Home**
  IoT solutions in AWS Marketplace, along with AWS IoT services, enable organizations to quickly leverage ready-to-use applications tailored to their business needs, ensure connectivity, and provide device management to speed up the insights that fuels efficiency, productivity, and growth. AWS IoT offers many use cases from Industrial applications for predictive quality to commercial applications for traffic monitoring. But, AWS IoT provides also Connected Home [aws20a], a smart home solution for home automation, home security, monitoring, and home networking.

  Each connected device or client must have a credential to interact with AWS IoT. All traffic to and from AWS IoT is sent securely over Transport Layer Security (TLS). AWS cloud security mechanisms protect data as it moves between AWS IoT and other AWS services. Authentication is been based on X509 certificates. AWS Identity and Access Management (IAM) is an AWS service that helps an administrator securely control access to AWS resources. IAM administrators control who can be authenticated (signed in) and authorized (have permissions) to use AWS IoT resources [aws20b].

- **HomeKit**
  HomeKit is a software framework by Apple that lets users configure, communicate with, and control smart-home appliances using Apple devices [app20a], released in September 2014. By designing rooms, items, and actions in the HomeKit service, users can enable automatic actions in the house through a simple voice command to

Siri or through the Home app. HomeKIt provides discovery HomeKit-compatible automation accessories as adds them to a persistent, cross-device home configuration database.

HomeKit uses a secure pairing mechanism to authenticate with an iDevice, e.g., iPhone, iPad, etc. It employs a propietary HomeKit Accessory Protocol (HAP) to enable third-party accessories to communicate with home or Apple devices. The HomeKit Accessory Protocol supports both IP and Bluetooth LE as transport protocols. The pairing process depends on the transport protocol. There is also hardware security in In. HomeKit introduced an authentication co-processor that only members enrolled in their MFi program can put into their accessories. A commercial HomeKit accessory must include an authentication coprocessor, and additionally include a Wi-Fi Alliance certificate or Bluetooth SIG certification, depending on the type of transport for that particular device [app20d]. In terms of authorization, user can manage remote access and edit permissions for people that he invite to control his home [app20b].

- **HomeAssistant**
  The Home Assistant project started in September 2013. In November 2013, the core functionality was first published on GitHub. As of May 2020, it has over 1930 developers who have contributed to its core. Is a free and open-source home automation software designed to be the central control system in a smart home or smart house. Written in Python its main focus is on local-control and privacy [app20c].

  One major advantage of Home Assistant is that it's not dependent on cloud services. Remote access can be achieved either by using Home Assistant cloud, or enabling VPN or SSH tunnel encrypting traffic. It supports authentication login-based, and Multi-Factor authentication based on the solution of a second challenge after password input. Also, it supports external OAuth 2.0 authentication. In terms of authorization, permissions limit the Things a user has access to or can control. Permissions are attached to groups, of which a user can be a member and the combined permissions of all groups a user is a member of decides what a user can and cannot see or control. Policies are dictionaries that implemented internally but this is an experimental feature that is not enabled or enforced yet [hom20b]. Finally, there exist a current limitation, that other user accounts will have the same access as the owner account. However, in the future, non-owner accounts will be able to have restrictions applied [hom20a].

- **ThingsBoard**
  ThingsBoard Inc. is a US corporation founded in 2016. Is an open-source IoT platform for data collection, processing, visualization, and device management [Thi20d]. It

enables device connectivity via industry standard IoT protocols - MQTT, CoAP and HTTP and supports both cloud and on-premises deployments. ThingsBoard combines scalability, fault-tolerance and performance. It has two editions, Community and Professional (CE, PE). Each one has it's own security features but Professional Edition has a significant advantage over Community as it incorporates authorization capabilities based on roles.

ThingsBoard in both editions provide Single Sign On functionality for customers and automatically create tenants, customers or subcustomers using external user management OAuth 2.0 protocol [Thi20c]. Device credentials are used in order to connect to the ThingsBoard server by applications that are running on the device. ThingsBoard is designed to support two different device credentials, access tokens and X.509 certificates [Thi20b]. In terms of authorization Community Edition supports straight-forward security model with three main roles: System administrator, Tenant administrator and Customer user. System administrator is able to manage tenants, while tenant administrator manages devices, dashboards, customers and other entities that belong to particular tenant. Customer user is able to view dashboards and control devices that are assigned to specific customer. This kind of edition functionality is sufficient for a lot of simple use cases, especially building real-time end-user dashboards. Instead, Professional Edition brings much more flexibility in terms of user, customer and role management. It is designed to cover use cases for businesses and enterprises with multiple user groups that have different permissions but may interact with the same devices and assets. It uses Advanced Role-Based Access Control (RBAC) for IoT devices and applications [Thi20a].

In table 3.1, are presented information about smart home frameworks. The vendor of each solution is an important factor because each of the vendors, has it's own user community and standards. The larger the community is (more users), the greater security evolves and requirements are arise as more and more users will use, search the framework. Some of the vendors cover a wide range of IoT solutions beyond smart home and that means, they provide further services to support this range such as cloud, computing infrastructure and data bases. Last but not least, is the cost that every framework comes with. HomeAssistant for example comes at no cost while other technologies require the purchase of a specific gateway or hub device to manage the smart home and that adds an extra cost.

In table 3.2, a security overview is presented for each of the frameworks focusing on the most crucial security aspects. Authorization, authentication, integration with other standards and technologies, interoperability, cloud support, and gateway support are the main characteristics we focus on.

	Vendor	Year	Community [Users]	Use Case	Software	Cost
Nest	Google	2010	105018	Smart Home	Proprietary	>$90
SmartThings	Samsung	2011	87142	Smart Home	Proprietary	>$70
Connected Home	Amazon	2015		Smart Home	Proprietary	Low
HomeKit	Apple	2014	No communtity support	Smart Home	Proprietary	$179
HomeAssistant	Home Assistant Core Team and Community	2013	73165	Smart Home	open-source [1930 developers]	Free
ThingsBoard	ThingsBoard Inc.	2016	>1747	Smart Metering, Energy, Farming - Fleet Tracking - Smart Home	open-source [193 developers]	>$10/month

Tab. 3.1: Smart home frameworks general comparison

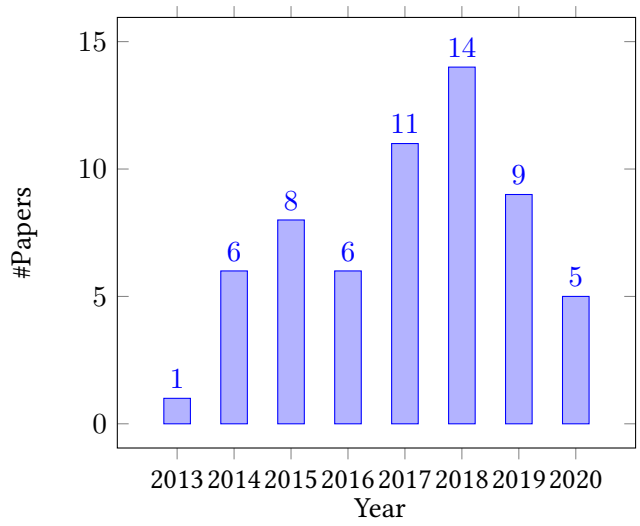
	Authentication	Authorization	Integration	Interoperability	Cloud	Gateway
Nest	OAuth 2.0	OAuth 2.0, Permissions		REST API	GoogleCloud-Platform	Google Nest Hub
SmartThings	OAuth 2.0	OAuth 2.0		SmartThings API, SmartApp API	SmartThings Cloud	hub / cloud
Connected Home	X.509	Identity Access Management (IAM) - Access Control List (ACL)	OAuth 2.0	AWS IoT API	AWS Cloud	gateway / cloud
HomeKit	hardware / software-based Authentication Coprocessor	Permissions			iCloud	hub
HomeAssistant	Login / Multi-factor	Permission Group	OAuth 2.0	REST, Websocket API	Home Assistant Cloud	gateway / cloud
ThingsBoard	OAuth 2.0 / Access Token / X.509	Limited Access Control (CE) / RBAC (PE)	Sigfox / AWS IoT (PE)	REST, Websocket API	ThingsBoard Cloud (PE)	gateway

Tab. 3.2: Smart home frameworks security comparison

3.2 openHAB Security Research

While there are several smart home solutions on the market, openHAB plays an important role as it has its own large users community and it is a solution that implements integration

among many different manufacturers in smart devices field. Therefore, openHAB has piqued the interest of the research and scientific community and as a result it has been studied and implemented in different environments. But it has also been studied and analyzed under the security perspective. An analysis of openHAB research is being made since framework release in market with emphasis on academic, scientific articles dealing with security aspect.
In the next chart bar chart, we notice the scientific articles on openHAB in search from 2013 to 2020 and with the word OpenHAB in the title of each article as we focus on articles that primarily deal with the openHAB platform. The search was based on the Google Scholar platform [Goo].



As can be seen from the above diagram, 60 scientific articles have been studied in the given time interval. However, we focus on the articles that have as their main objective the study and analysis of the security of openHAB in terms of Access Control. There is a subset of these articles concering openHAB bindings security but this is out of our study scope. Accordingly, the articles that deal with openHAB Access Control are 2.

- **Building an Open Source Access Control System for Fablabs based on odoo and openHAB**
  This study [MS20] studies a feasible open source access control solution based on openHAB framework. The issue that occurs is the access control of machines for visitors in small workshops since everyone can use a machine for their work. As for the management of all machines openHAB plays the role of coordinator since it is vendor-agnostic.
  The solution proposed is the design of a REST API with which each visitor will interact, authenticate and get access to the requested machine he demands for his work. The proposed system, in addition to API, has a database that maintains the booking between the visitor and the machine he has booked. The Access Control

between the visitor and the machine is modeled as a record on the data base under the limitations that for the time the booking lasts no one else can access the same resource (machine) but the same visitor must request for release of the resource.

- **Scaling Home Automation to Public Buildings: A Distributed Multiuser Setup for OpenHAB 2**
  Authors presented [Hei+17] the issue of automation lack in large public buildings. Home automation solutions work well in small places, but limited range of wireless transceivers is not sufficient for buildings. Therefore, the proposed system is a set of openHAB gateways acting as slaves in different areas in a building and an openHAB gateway acting as master interconnecting them with MQTT protocol. In this scenario, the master openHAB gateway is responsible to manage, add, remove items and Things through MQTT broker and forwards the commands to openHAB slaves. This solution implemented through openHAB software modification. And upon this, authors added user authentication and access control based on group membership of a user.
  However, this study does not implement access control in the context of a single home with a central openHAB gateway where Things is in the same place. Still, the proposed solution is based on openHAB software modification and it cannot be used by other systems, designs.

3.3 Current Smart Home Access Control Implementations

Today IoT has evolved in a complex environment, with a massive number of devices interconnected in it and constantly new services emerging. In this context, new security requirements are being raised as all these devices raise issues of privacy and secure management, access. In fact, additional and different requirements pose challenges for the use of various security protocols. Specifically, the need arises for dynamic and fine-grained access control mechanism, where users/resources are constrained [Oua+17]. However, research community is currently studying and implementing access control techniques over IoT and consequently there are works on this field.

1. **Smart Contract-Based Access Control**
   Blockchain technology provide a simple infrastructure for two devices to directly transfer a piece of property such as money or data between one another with a secured and reliable time-stamped contractual handshake. To enable message exchanges, IoT devices will leverage smart contracts which then model the agreement between the two parties. This feature enables the autonomous functioning of smart devices without the need for centralized authority.

Authors implemented an IoT access control scheme [Zha+19] proposing a smart contract-based access control framework which consists of multiple access control contracts (ACCs), one judge contract (JC) and one register contract (RC) to achieve distributed and trustworthy access control for IoT systems.
In the framework, each ACC maintains a policy list (what permission with which action on which resource) and provides one access control method for a subject-object pair, which implements both static access right validation based on predefined access control policies and dynamic access right validation by checking the behavior of the subject. The ACCs also provide functions for adding, updating and deleting access control policies. Once called by a subject for access control, the ACC will be run and verified by most participants in the system, ensuring the trustworthiness of the access control.

2. **Blockchain-Based Access Control**
   Another important implementation of access control is mentioned in the work of Oscar Novo, *"Blockchain Meets IoT: An Architecture forScalable Access Management in IoT"* [Nov18] in which the proposed architecture describes a new decentralized access management system. Access control information is stored and distributed using blockchain technology. Also, architecture defines a new node called management hub that requests access control information from the blockchain on behalf of the IoT devices. In addition to that, the solution involves a single smart contract that defines all the operations allowed in the access control system. Finally, scalability issue of managing access to billions of constrained devices is addressed.

3. **Decentralized Identifier (DID) - Verifiable Credentials (VCs)**
   One major technology Decentralized identifiers (DIDs) are a new type of identifier that enables verifiable, decentralized digital identity. A DID identifies any subject (e.g., a person, organization, Thing, data model, abstract entity, etc.) that the controller of the DID decides that it identifies. Along with DIDs, another technology of the future is VC. Every Verifiable Credential is created by an issuer and allows the information validation. In the work of D. Lagutin, Y. Kortesniemi, N. Fotiou [LKF18], is presented an OAuth 2.0-based method to delegate the processing and access policy management to the Authorisation Server thus allowing also systems with constrained IoT devices to benefit from DIDs and VCs.

4 System Design and Implementation

In this section, we introduce our OAuth 2.0-based architecture system using openHAB framework as a smart home solution. First of all, we analyze the security requirements that our system satisfies such as permissions delegation, authentication, resource access control based on OAuth 2.0 and interoperability. Also, we present the architecture of the system, explaining the reasons for this design. Then, we provide a detailed work-flow of our proof of concept implementation of the system, containing the important parts in an algorithmic representation of its gradual development in pseudo-code. For each specific part, we present all the appropriate details. Finally, we experiment with the system in a basic scenario verifying that our model has practical potential.

4.1 Requirements

Our system is based on OAuth 2.0 technology and the use of JSON Web Tokens to provide security services in the context of smart home. It is clear that a set of security requirements for our access control framework based on an IoT environment should be ensured. These requirements are:

- Flexibility: Our proposed access control framework should be able to authorize resource requests at a flexible way. The proposed design offers straightforward access control with OAuth 2.0 using access tokens for high level resources and for low-level resources based on the roles of a user.

- Scalability: Our design should be scalable to manage the growning number of IoT devices in the smart home as well as the new users requesting resource access. Our token-based design provides scalability as authorization service can register users on demand and issues access tokens.

- Integrity: Firstly, resources in an IoT Gateway should only be controlled by legitimate users. In our design, a resource's access authority is validated by access token and roles. Thus, token integrity can be ensured using Hash-Based Message Authentication Code HMAC in JSON web token.

- Interoperability: OAuth 2.0 supports API for token issuance and resource register.

- Flexible interfaces: The proposed design to support HTTP operations such as token issuance and resource access, provides a web interface.

- User Authentication: OAuth 2.0 supports user authentication through client credentials grant type, each time a user requests an access token.

- Privacy: openHAB Gateway in our design should be deployed behind a reverse HTTP proxy. In this way, we can protect resource server from exposing open port in the Internet as identity remain hidden and privacy is enhanced.

4.2 Design

An overview of our proposed design is shown in figure 4.1. In this figure, we can distinct the entities, client, OAuth 2.0 Authorization Server, Proxy Server, openHAB Gateway and IoT devices. The questions we try to answer in this context are, which are the entities in our design, which of them interact with each other and why they interact. In our scenario, gateway and proxy server are located at smart home. Client has two roles, smart home owner (or admin) who has full resource access, and guest, a user who requests temporary resource access entering home.

The workflow of our system is described below. The first scenario is the communication between user and authorization server. User as a home owner is able to register Thing (IoT device) endpoints indicating actions (eg, turn TV ON), or APIs (eg, /rest/TV/) in authorization server. Apart from this, he is able to create new user (guest) and link the user account with smart home Things endpoint after a user request. Still, a user may request from authorization server, an access token issuance representing his authorization level and what Things endpoint he is able to access. But, before that, he has to be authenticated in the authorization server using client credentials, as owner created for him. In the second scenario, user communicates with proxy server making an http request indicating some action on a Thing, or a Thing creation, modification or deletion depending on his authorization level. Proxy should validate the incoming access token, checking it's private key and parsing resources tag from json web token. After successful validation, proxy should forward the http request to openHAB Gateway, and gateway should execute the request and response the result back to proxy, where proxy should forward response back to user.

Proxy integrates configuration (file), in which we can set and define Things endpoints, cryptographic key for validation purposes and domain and port of openHAB Gateway. In

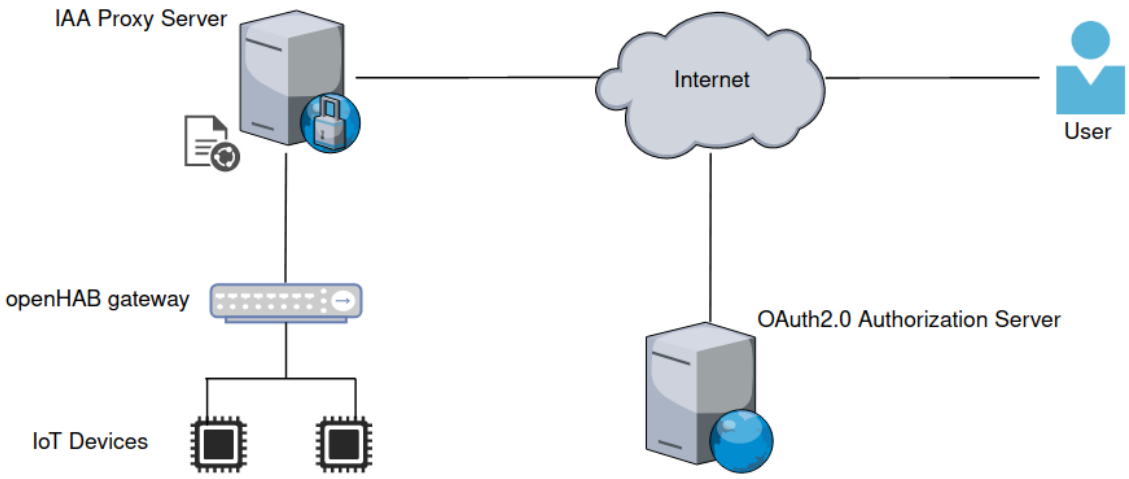


Fig. 4.1: Our OAuth 2.0-based openHAB smart home architecture

terms of security and privacy, proxy protects gateway from DDoS attacks since gateway IP is secret in the Internet. Consequently, even if attackers could break down the proxy service, resources behind openHAB would remain secure and functional. Thus, openHAB Gateway is responsible to communicate and exchange HTTP messages only with proxy as this way, gateway cannot be accessed by other machine or computer protecting it's resources.

Things (IoT devices) can be any device, from a sensor to an actuator since openHAB supports a wide area of devices and manufactures but also general digital devices can be used.

openHAB Gateway is responsible for communicating with Things. It offers many connectivity techniques such as Bluetooth, Wi-Fi, Ethernet.

4.3 Implementation

In this section, we describe the hardware we used to deploy our applications and services, how we deploy each of the components as well as the scenario that validates the functionality of our proposed system.

4.3.1 IAA Proxy Server

Proxy server in our design is an open-source implementation of identity-authentication-authorization HTTP proxy written in Python [nik20]. In that repository, there are details about how to install and deploy the proxy server and edit the configuration file depending on the needs of the scenario. We used a Raspberry Pi 3 model B+ hardware [ras20a] to

deploy proxy, as raspberry offers low cost hardware and capable of running simple tasks. As for the operating system on the raspberry pi, Raspbian was selected [ras20b], in release 10 and code-name buster. To deploy proxy in raspberry pi, we should install some python modules like Werkzeug, pynacl, pyjwt as described in [nik20]. Raspbian OS comes with python language preinstalled. Finally, we should add Thing URLs and gateway ip, port to configuration file as [nik20] described. Proxy as a service runs at 9000 port.

4.3.2 openHAB Gateway

To install openHAB software, we used a second raspberry pi 3 model B+ hardware running Raspbian 10 buster as operating system. The openHAB version we installed is 2.5.9 Build and steps required to install openHAB are described in the official site [ope20a]. As a service, openHAB runs at 8080 port. openHAB Gateway is responsible for communicating only with proxy. To satisfy this requirement we implement a hybrid solution based on application and system (network) level. In the application level, we leverage the environment variable openHAB offers, *OPENHAB_HTTP_ADDRESS*, which by default allows a connection from any location (0.0.0.0). In the system level, we apply firewall rule by enabling *ufw* [ubu17] tool for Linux systems. Enabling firewall, we enter as parameter proxy IP and define rule to allow only http traffic from this IP. As a result, openHAB Gateway is restricted to communicate only with proxy denying connections from any other hosts.

The IoT devices that we used for testing are a TP-LINK smart plug, with two attributes and a LG smart webOS TV with three attributes. The attributes of the first device are the Power On/Off attribute and the Switch Led ON/OFF. The second IoT device has a Power On/Off, a mute ON/OFF and a volume control attribute.

openHAB provides REST API, therefore proxy supports HTTP POST, GET, PUT and DELETE requests in order to provide functionality to users.

4.3.3 OAuth 2.0-based Authorization Server

As for the authorization service, we used an OAuth 2.0-based Authorization Server [con20]. Our Authorization Server provides Web interfaces to register users, resource endpoints and links between them. Also, it uses RS256 cryptographic algorithm for creating access tokens.

Below, in the table 4.1, we can see the hardware used for our services installation.

	Hardware	Memory (RAM)	Memory (Storage)	Processor	OS
openHAB Gateway	Raspberry Pi 3 Model B+	1GB	32GB	Broadcom 64-bit SoC @ 1.4GHz	Raspbian 10 buster
IAA Proxy Server	Raspberry Pi 3 Model B+	1GB	32GB	Broadcom 64-bit SoC @ 1.4GHz	Raspbian 10 buster
Python Flask WebApp (UI)	Raspberry Pi 3 Model B+	1GB	16GB	Broadcom 64-bit SoC @ 1.4GHz	Raspbian 10 buster

Tab. 4.1: Hardware employed for our design

4.3.4 Python Flask WebApp (UI)

In order to give to the user the ability to interact with the OAuth 2.0-based Authorization Server and proxy, we implemented an extra feature. We developed a Web application, using python flask, which is a library providing modules for building Web functionality and interfaces [Fla20]. The purpose of the Web application is to provide to the users two interfaces, one for interacting with our Authorization Server requesting access token, and the other for interacting with the proxy requesting HTTP actions on Things.
The hardware we used to deploy the Web application is a raspberry pi 3 model b+, running Raspbian 10 buster, and we considered for our scenario to deploy Web application inside smart home. Consequently, the Web application communicates with the proxy.

First of all, Web application should support HTTP POST, GET, PUT and DELETE methods, since openHAB provides REST API. Web application exports url */smart_home_proxy* which can be accessed by users. In case a user wants to create a new Thing into openHAB, he enters the json description, HTTP method, his access token for authorization, and the HTTP URL for this action (eg, Thing creation) that openHAB will execute. Thhese four methods are shown in the form of pseudo-code in algorithms 1, 2, 3, and 4, respectively.

As a service, python flask Web app runs at 7000 port.

Algorithm 1 GET request

1: **procedure** GET
2: $token \leftarrow json\_web\_token$ ▷ token user has from OAuth 2.0
3: $method \leftarrow' GET'$
4: $url \leftarrow thing\_url$ ▷ eg, Thing URL indicating power on
5: $headers \leftarrow \{'Accept' : ' application/json','Authorization' : 'Bearer' + token+''\}$
6: $response \leftarrow http.get('proxy : 9000' + url, headers)$
7: **return** $response$

Algorithm 2 POST request

```
1: procedure POST
2:     body ← json_description                ▷ thing json description, eg, Thing creation
3:     token ← json_web_token                 ▷ token user has from OAuth 2.0
4:     method ←' POST'
5:     url ← thing_url                        ▷ eg, Thing URL indicating power on
6:     headers ← {'Accept' :' application/json','Authorization' :' Bearer' +
   token+''}
7:     response ← http.post('proxy : 9000' + url, headers, body)
8:     return response
```

Algorithm 3 PUT request

```
1: procedure PUT
2:     body ← json_description                ▷ Thing json description, eg, Thing attribute
   modification
3:     token ← json_web_token                 ▷ token user has from OAuth 2.0
4:     method ←' PUT'
5:     url ← thing_url                        ▷ eg, Thing URL indicating power on
6:     headers ← {'Accept' :' application/json','Authorization' :' Bearer' +
   token+''}
7:     response ← http.put('proxy : 9000' + url, headers, body)
8:     return response
```

Algorithm 4 DELETE request

```
1: procedure DELETE
2:     token ← json_web_token                 ▷ token user has from OAuth 2.0
3:     method ←' DELETE'
4:     url ← thing_url                        ▷ eg, Thing URL indicating thing id
5:     headers ← {Accept : application/json, Authorization : Bearer + token}
6:     response ← http.delete('proxy : 9000' + url, headers)
7:     return response
```

On the other hand, Web application provides interface for requesting an access token from Authorization Server. To support this functionality, Web application provides an HTTP POST method that user can execute providing client credentials as grant type and gaining his access token with his available resource. Web application exports */smart_home_oauth* url which can be accessed from user. This method is shown in the algorithm 5.

Algorithm 5 Token Request

1: **procedure** TOKEN REQUEST
2: $clientId \leftarrow user\_client\_id$
3: $clientSecret \leftarrow user\_client\_secret$
4: $data \leftarrow \{grant\_type : client\_credentials, client\_id : clientId, client\_secret : clientSecret\}$
5: $headers \leftarrow \{Content - Type : application/x - www - form - urlencoded\}$
6: $response \leftarrow http.post('https : //as.controlthings.gr/oauth2/token/', headers, data)$
7: **return** $response.json()$

4.3.5 Simple Resource Access Scenario

To experiment with and evaluate our architecture, we implement a simple resource access scenario. In this scenario, we created a user, registered the TP-LINK smart plug attribute endpoint, and linked the user with this endpoint. The first part is to request his token from OAuth 2.0-based Authorization Server and the second is to communicate with proxy requesting to power on the led of the Thing. We consider as an assumption, that the user (guest) is in the smart home. These two actions should be done through our Web application. The workflow for this scenario is:

1. openHAB TP-LINK smart plug led URL
   With the addition of the TP-LINK Smart Plug, openHAB creates URLs to identify the properties of the smart plug. So a user who has the appropriate authorization, can call the URL and cause the appropriate action on that object. The URL that is able to open the led is */basicui/CMD?tplinksmarthome_hs100_CCF21B_led=ON*.

2. User - Endpoint Register Link
   In this step, the owner of the home must log in to Authorization Server (via web browser) and create a user account for the guest. This will create the client id and the corresponding secret that will be given to the guest user. Then, owner should register into Authorization Server the URL (endpoint) that opens the led of the smart plug. By adding the device to openHAB, the device attributes become accessible via HTTP URL. Finally, the owner should associate, that is, give authorization to the user who created to call this URL. Client id and secret of created user are *LiIONmIG0+K0cf08lagBig==* and *WGnVksDpXasQ09b1Nx9UJg==* respectively.

3. WebApp - Token Request
   Using algorithm 5, we call the URL */smart_home_oauth* and then enter the client id and secret. As a response from Authorization Server, a JWT is returned in which there is information about the authorization of the user (what resources he can access). If we decode the JWT, we can see the authorized resources as part of the payload part:
   {"resources": [
   "www.example.com/LightON",
   "/basicui/CMD?tplinksmarthome_hs100_CCF21B_led=ON",
   "/basicui/CMD?tplinksmarthome_hs100_CCF21B_led=OFF",
   "/rest/things",
   "/rest/items/tplinksmarthome_hs100_CCF21B_led"
   ]}
   It can be seen that, inside the token there is the claim that tp-link led on url is accessible from this user.

4. WebApp - Request Led ON
   As last step, user (guest) can request to open the smart plug led (eg, ON state). Using algorithm 1, user call */smart_home_proxy* and then he must give inputs of access token, and Thing action URL. As a result, Web application should redirect HTTP GET request to proxy.

5. Proxy - Authorization validation
   Before proxy redirects the HTTP request to openHAB Gateway, it should validate the access token from the user. A prerequisite is that the URL, which the user wants to call, must exist as a resource in the configuration file of the proxy. Proxy also has to check the token with the public key of Authorization Server and verify it. Finally, within the resources tag of the token there must be this URL, in order to prove that user has been authorized.

6. Gateway response
   After successful validation, proxy forwards the HTTP request to openHAB Gateway, which executes the HTTP request causing the desirable action on the smart tp-link plug, namely opening the led. As a result, openHAB will response a 200 status message to proxy. Proxy will redirect this message to Web application and the user will be informed that the action was done.

5 Evaluation

In this chapter, we will evaluate our system based on performance and security aspects. In the first part, we will analyze the response times, so as to figure out whether the system is fast or slow, which has an impact on the end user for the quality of the service provided. In the second part, two scenarios will be presented, one without the solution we proposed and one with the application of our solution. Thus, the merits of our design will be ascertained.

5.1 Performance evaluation

In this section, we present the performance evaluation of the proposed system. We used the scenario we showed in the subsection 4.3.5, Simple Resource Access Scenario. In this scenario, we have the GET method, by which the user calls to open the led of a smart plug. After that, user is receiving his token from the Authorization Server. Therefore, we will measure the execution time of these two actions in order to gain knowledge of the performance. To measure the time taken, we add appropriate time measurement code in the web application, utilizing the time library [Pyt20] of the python language. Below, there is the timer implementation, as shown in algorithm 6.

Algorithm 6 Timer

```
1: procedure TIMER
2:     start ← time()
3:     response ← http.get(..)
4:     end ← time()
5:     time_elapsed ← end − start
6:     return time_elapsed
```

The times extracted are shown in the table 5.1. The results are presented in milliseconds. The parameters considered are the response time to a user's request for token access to Authorization Server and the response time to a user's GET request to the proxy (and thus to openHAB Gateway). To have more reliable data, we performed the measurements five times and calculate the average number of them. In the table 5.2, average execution times are shown.

As for response times, between the user's communication with Authorization Server, we notice small fluctuations. This communication passes through the Internet and therefore

System GET method	Access Token request
282	826.6775
247.0974	3917.4396
238.7850	735.2492
252.9530	333.1589
241.3613	1029.9925

Tab. 5.1: Execution time in msecs (milliseconds)

System GET method	Access Token request
252.43934	1368.50354

Tab. 5.2: Average execution time in msecs (milliseconds)

network traffic at the level of the Internet is quite changing every time due to its complexity. It is therefore not affected by our system, but by external factors.
As for the GET request, it is clear that the times might have been slightly smaller if the proxy server was not presented. The proxy obviously inserts a small time delay since HTTP request has to go through there. Furthermore, token validation must be done, a process that has its own short duration. But we accept this small amount of time, since the proxy ensures important features in the security of our system.

5.2 Security evaluation

Each system in order to be considered secure, must meet certain security requirements such as the integrity of the data it manages, the confidentiality between the participating entities and the availability in order to provide its entire functionality seamlessly. Through the two scenarios, we prove that our system meets the requirements for smart home access control, addressing the problems that arise in the simple scenario without using the components of our design. The first scenario concerns a use case, in which a user takes advantage of openHAB official security recommendation to use a simple proxy in order to communicate with smart home and cause a Thing change (or add, modify, remove a Thing). While in the second scenario, we apply the solution that we propose in this work, so as to observe the difference and point out the benefits.

As far as end-to-end security is concerned, the link needed to be secured is the communication between a Thing and the Access Point (AP) (network router). There are some Things of various manufacturers that incorporate security mechanisms from their manufacturer. One such example is the Z-WAVE device class [ope20d]. However, there are cases, when devices that do not have the necessary computing resources, need to communicate wireless (e.g., over Wi-Fi) with the AP. An example is the use of a simple sensor, such as a humidity sensor, temperature sensor, etc. In a scenario like that, a first level of security is its con-

nection to a hardware platform, such as Arduino or Raspberry Pi, where information is transferred via cable (GPIO pins). A second level of security is the secure communication between the platform and the AP. There are APs that support security protocols such as Wi-Fi Protected Access 2 (WPA2) [Wik20c]. This protocol encrypts all traffic at the network level with a secret key which takes different values from time to time using the AES encryption algorithm. In this way, secure communication between Thing and AP is achieved. Furthermore, the communication between the openHAB Gateway and the AP is also secured, because there are connected with an Ethernet cable, which provides reliable and fast data transfer.

But even if our system offers a remarkable level of security, it should not be omitted that the system is a part of an IoT environment. Therefore, IoT devices as well as the openHAB Gateway in the hardware and operating system level should be studied and security provision should taken into account.

5.2.1 Simple security scenario

In first scenario, user leverages the recommended security level that openHAB proposes [ope] and that is the integration of a reverse proxy server into smart home. But, in this scheme, we reject proxy server deployment on the same machine that openHAB runs, the gateway. This may lead to security issues because DDoS attack could have consequences both on proxy and openHAB services since running on the same physical computer machine. Reverse proxy could be NGINX [NGI20], or Apache Proxy [APA20] implementations. These proxies offers a basic authentication level, using a configuration-password file storing user information. Furthermore, proxies, can provide HTTPs encryption between openHAB and proxy ensuring data integrity.

As a result, in this scenario, there are two users who attempt to perform an action on Smart Things (IoT devices). Both of them, have full access on every device since they are able to call any HTTP URL indicating action on a Thing. Consequently, there is no access control mechanism restricting access on Things. We assume that user A, tries to open tp-link smart plug led (LEN=ON) and User B turn on smart webOS LG TV (Power=ON). But, we consider that User B should not have access to turn on the TV. This scenario is shown in figure 5.1.

The URLs that cause the corresponding actions and each user call are shown in table 5.3. Obviously, in order for the URLs to be executed by openHAB, the address of the proxy (eg, 'http://proxyIP/URL') should be added. Before users call HTTP request, they can be authenticated using password-based login, a function provided by proxy. After that, each user makes an HTTP request with the corresponding URL, adds the proxy IP and the request is executed by the openHAB application. Consequently, user A opens smart plug

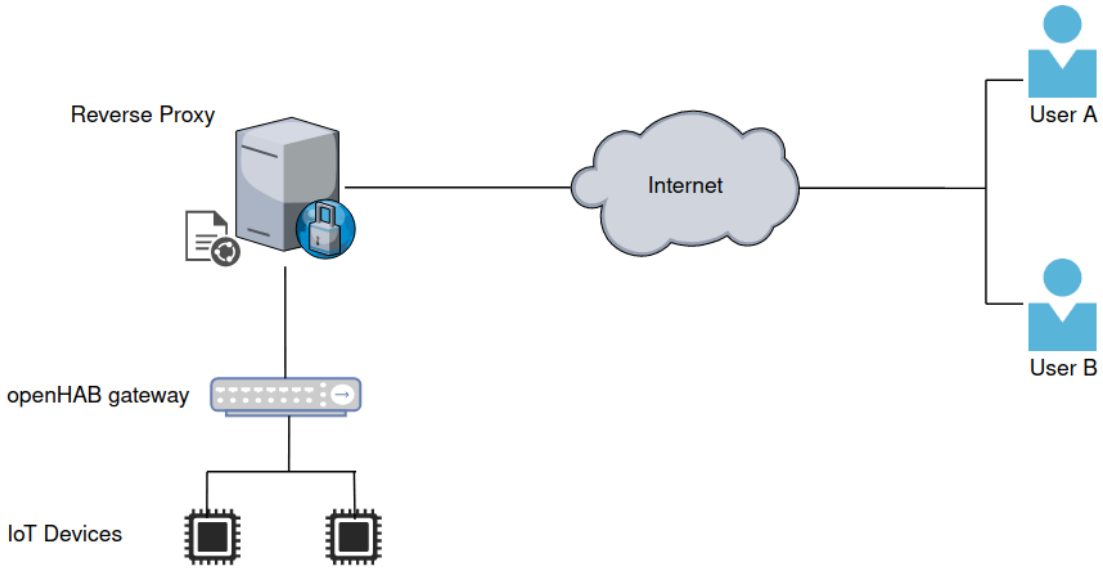


Fig. 5.1: Simple security scenario concerning smart home access control

led (ON), and user B opens the smart TV (ON). The result is, that user B was not supposed to have access on smart TV, but he accessed it. This is a consequence due to lack of access control into this design. Therefore, security could be compromised.

	Thing URL
User A	/basicui/CMD?tplinksmarthome_hs100_CCF21B_led=ON
User B	/basicui/CMD?lgwebos:WebOSTV94a13f5f-8570-b11b-bd6f-c1709041e590:power=ON

Tab. 5.3: Thing URLs each user calls

5.2.2 Proposed OAuth 2.0-based scenario

In the second scenario, the two users will attempt to gain access to the two IoT devices (smart plug led, TV power) and trigger some actions. We take advantage of the design that we propose as it has been described in 4.2. Now, users will have to apply for authorization before gaining access to the devices. That means, home owner will have to create an account for user A and associate it with the authorization of access to the smart plug. While the second user should not have access to the TV. In this case, if user B attempts to power ON the TV, he will be prevented by the system in its attempt to gain access.

In case of user A, he makes the HTTP request, requesting smart plug led turn on. First, he must gain his access token from Authorization Server. Then, he should make HTTP request along with token to IAA Proxy. Proxy is responsible for validating the token and deciding if the user has the authorization level to access the URL. This user is indeed authorized to access the requesting resource. Consequently, the HTTP request is forwarded to openHAB and executed (e.g., turning ON smart plug led).

However, for user B the case is different. User B should not have access on smart TV (we do not want that as scenario). So, user creates the HTTP request, requesting power on the smart TV and he sends it to IAA Proxy. But, proxy will not process the request since no access token is received. Therefore, it will reject the request, denying access. Still, user B could construct a similar or fake access token. In this case, proxy would validate token as invalid, because there is no the requested resource embedded as part of token. Again, proxy would have to deny the access.

We notice that without our design, all users fall into a category of users, which is characterized by the same level of access which is full access to all Things. Which is obvious, since there is no access control mechanism, except the proxy server offering encryption and authentication but not addressing the access control problem.

The design that we propose and implement comes to extend this simple scenario and address the issue of access control and the entity that implements it and determines the access policies.


ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

6 Conclusions and Future Work

In IoT, more and more devices are interconnected, creating an even wider environment. Thus, IoT raises new challenges, such as support for heterogeneity, mass connectivity, and interoperability, which can be achieved by utilizing existing solutions, promising new technologies, or combinations of them.

In IoT environments, security always plays a crucial role because many of today's technologies fail to meet this requirement to a desirable degree. Devices that are part of an IoT system usually do not have the specifications to implement security methods, so a mechanism is needed off them to secure them. Access control is an important security requirement in environments with many devices and resources, especially when many and diverse users request access to them. In the context of a smart home, technology and research show that a robust security model is provided by the use of OAuth 2.0 standard, which determines which entities have access to what resources for how long and with what rights. To prove the level of access allowed, JSON Web Token technology is utilized, a transparent, reliable, and fast data structure.

During the preparation of this thesis, two research questions were raised. Both of them were addressed by fulfilling the goals of the thesis. First of all, by reviewing the related literature, we concluded that there is not one security model that can address every security problem that there is, and it can fit in any framework and system, but each framework incorporates its own solution. For example, OAuth 2.0 standard is suitable only for authorization and access control. Furthermore, to investigate how modern security models can be combined in order to provide more secure and flexible smart home frameworks, we design an architecture that use well-known technologies such as JWTs and OAuth 2.0 standard so as to achieve access control and secure the smart home system, openHAB. At the home gateway level we have chosen the openHAB framework, an open source tool with vendor-agnostic support. However, openHAB fails to provide the smart home environment with the required level of security. Therefore, the solution we propose introduces the IAA proxy server that communicates exclusively with the openHAB gateway and enhances its privacy by isolating it from the Internet. It also introduces the OAuth 2.0-based authorization server in which users request their access token indicating their resource access level.

The performance and security evaluation of our system relies on two scenarios. In terms of performance, the proposed system demonstrates good results. The average time to perform an action on a Thing (e.g., switch on the led of a smart plug) is 252.4 milliseconds

and the time for requesting an access token is 1368.5 milliseconds respectively, values considered to be negligible for our system. On the other hand, in terms of security properties, our system supports flexibility due to OAuth 2.0-based authorization server high level resources access tokens, and scalability as OAuth 2.0-based authorization server can register multiple users on demand. Moreover, integrity is ensured as HMAC ensures JWT data integrity, and interoperability is provided by exposing OAuth 2.0-based authorization server API for token request and user resource registration. Finally, the system provides user authentication as OAuth 2.0-based authorization server offers client authentication based on client credentials, and privacy as IAA proxy server isolates openHAB gateway. Security evaluation also shows us that unauthorized users cannot access resources (Things) that they are not supposed to have access.

As future work, we can consider the blockchain technology, which can be used as a transparent mechanism, as raises new prospects for dealing with security gaps such as privacy, security, and integrity issues within a smart home. It provides access control application capabilities and is a technology that does not depend on different standards used in smart home environments. Therefore it can be adopted and integrated quickly and reliably into the environment of an IoT smart home.

Another promising future technology is the Verifiable credentials (VCs). VCs represent claims about credentials and are assigned to a holder. Thus, in controlling access to IoT environments, they promise enhanced privacy and the ability to retain information about what a holder is authorized to do.

Bibliography

[AO19] T. Adesina and O. Osasona. "A Novel Cognitive IoT Gateway Framework: Towards a Holistic Approach to IoT Interoperability". In: *2019 IEEE 5th World Forum on Internet of Things (WF-IoT)*. 2019, pp. 53–58.

[APA20] APACHE. *Apache Module mod$_{proxy}$*. 2020. URL: https://www.openhab.org/docs/installation/security.html#running-openhab-behind-a-reverse-proxy (visited on Sept. 25, 2020).

[app20a] apple developer. *HomeKit*. 2020. URL: https://developer.apple.com/homekit/ (visited on Sept. 25, 2020).

[app20b] apple developer. *Share control of your home*. 2020. URL: https://support.apple.com/en-us/HT208709 (visited on Sept. 25, 2020).

[app20c] apple developer. *Share control of your home*. 2020. URL: https://www.home-assistant.io/ (visited on Sept. 25, 2020).

[app20d] apple developer. *Using the HomeKit Accessory Protocol Specification (Non-Commercial Version)*. 2020. URL: https://developer.apple.com/support/homekit-accessory-protocol/ (visited on Sept. 25, 2020).

[aws20a] aws. *AWS IoT - Connected Home*. 2020. URL: https://aws.amazon.com/iot/solutions/connected-home/?nc=sn&loc=3&dn=3 (visited on Sept. 25, 2020).

[aws20b] aws. *Security in AWS IoT*. 2020. URL: https://docs.aws.amazon.com/iot/latest/developerguide/security.html (visited on Sept. 25, 2020).

[BNS19] S. Balaji, Karan Nathani, and R. Santhakumar. "IoT Technology, Applications and Challenges: A Contemporary Survey". In: *Wireless Personal Communications* 108.1 (Sept. 2019), pp. 363–388.

[Chi+19] B. Chifor, S. Arseni, I. Matei, and I. Bica. "Security-Oriented Framework for Internet of Things Smart-Home Applications". In: *2019 22nd International Conference on Control Systems and Computer Science (CSCS)*. 2019, pp. 146–153.

[CIS15] CISCO. *Fog Computing and the Internet of Things: Extend the Cloud to Where the Things Are*. 2015. URL: https://www.cisco.com/c/dam/en_us/solutions/trends/iot/docs/computing-overview.pdf (visited on Sept. 25, 2020).

[con20] controlthings. *Control Center*. 2020. URL: https://as.controlthings.gr/Identity/Account/Login?ReturnUrl=%5C%2F (visited on Sept. 25, 2020).

[Dil20] G. Dileep. "A survey on smart grid technologies and applications". In: *Renewable Energy* 146 (2020), pp. 2589–2625.

[ECL] ECLIPSE FOUNDATION. *Eclipse SmartHome*. URL: https://projects.eclipse.org/projects/iot.smarthome (visited on Sept. 25, 2020).

[El-+19] Mohammed El-hajj, Ahmad Fadlallah, Maroun Chamoun, and Ahmed Serhrouchni. "A Survey of Internet of Things (IoT) Authentication Schemes". In: *Sensors* 19.5 (Mar. 2019), p. 1141.

[est20] estDevelopers. *OAuth 2.0 Authentication and Authorization*. 2020. URL: https://developers.nest.com/guides/api/how-to-auth (visited on Sept. 25, 2020).

[Fla20] Flask. *Flask Web Development*. 2020. URL: https://flask.palletsprojects.com/en/1.1.x/ (visited on Sept. 25, 2020).

[FOS20] J. A. Fadhil, O. A. Omar, and Q. I. Sarhan. "A Survey on the Applications of Smart Home Systems". In: *2020 International Conference on Computer Science and Software Engineering (CSASE)*. 2020, pp. 168–173.

[Goo] Google Scholar. *openHAB search*. URL: https://scholar.google.com/scholar?q=allintitl%5C%3A+openhab&hl=en&as_sdt=0%5C%2C5&as_ylo=2013&as_yhi=2020 (visited on Sept. 25, 2020).

[Hei+17] F. Heimgaertner, S. Hettich, O. Kohlbacher, and M. Menth. "Scaling home automation to public buildings: A distributed multiuser setup for OpenHAB 2". In: *2017 Global Internet of Things Summit (GIoTS)*. 2017, pp. 1–6.

[hom20a] home assistant. *Authentication*. 2020. URL: https://www.home-assistant.io/docs/authentication/ (visited on Sept. 25, 2020).

[hom20b] home assistant developers. *Permissions*. 2020. URL: https://developers.home-assistant.io/docs/auth_permissions (visited on Sept. 25, 2020).

[Int12] Internet Engineering Task Force (IETF). *The OAuth 2.0 Authorization Framework*. 2012. URL: https://tools.ietf.org/html/rfc6749 (visited on Sept. 25, 2020).

[Int15] Internet Engineering Task Force (IETF). *JSON Web Token (JWT)*. 2015. URL: https://tools.ietf.org/html/rfc7519 (visited on Sept. 25, 2020).

[KCR15] S. Kim, H. Choi, and W. Rhee. “IoT home gateway for auto-configuration and management of MQTT devices”. In: *2015 IEEE Conference on Wireless Sensors (ICWiSe)*. 2015, pp. 12–17.

[Kou20] Xristina Koulouri. “Μηχανισμός ΚατανομήςΠόρων σε Massive MIMO Δίκτυα Πέμπτης Γενιάς–Διατύπωση του προβλήματος του σακιδίου”. MA thesis. Greece: University of Patras, 2020.

[KV19] J. Krishnan and S. K. Vasudevan. “An Extensive Survey on IoT Smart Gateways, Software Architecture, Related Protocols and Challenges”. In: *2019 International Conference on Vision Towards Emerging Trends in Communication and Networking (ViTECoN)*. 2019, pp. 1–6.

[Lan+20] David J. Langley, Jenny van Doorn, Irene C.L. Ng, et al. “The Internet of Everything: Smart things and their impact on business models”. In: *Journal of Business Research* (2020).

[LKF18] D. Lagutin, Yki Kortesniemi, and Nikos Fotiou. “Enabling Decentralised Identifiers and Verifiable Credentials for Constrained Internet-of-Things Devices using OAuth-based Delegation”. In: 2018.

[MCM18] Dragos Mocrii, Yuxiang Chen, and Petr Musilek. “IoT-based smart homes: A review of system architecture, software, communications, privacy and security”. In: *Internet of Things* 1-2 (2018), pp. 81–98.

[MH19] Mardiana binti Mohamad Noor and Wan Haslina Hassan. “Current research on Internet of Things (IoT) security: A survey”. In: *Computer Networks* 148 (2019), pp. 283–294.

[MS20] Fabian Meyer and Michael Schäfer. “Building an Open Source Access Control System for Fablabs based on odoo and openHAB.” In: *SENSORNETS*. 2020, pp. 85–92.

[Nes20] Nest. *nest*. 2020. URL: `https://nest.com/` (visited on Sept. 25, 2020).

[NGI20] NGINX. *NGINX Reverse Proxy*. 2020. URL: `https://www.nginx.com/` (visited on Sept. 25, 2020).

[nik20] nikosft. *mmlab-aueb/identity-authentication-authorization*. 2020. URL: `https://github.com/mmlab-aueb/identity-authentication-authorization` (visited on Sept. 25, 2020).

[Nov18] O. Novo. “Blockchain Meets IoT: An Architecture for Scalable Access Management in IoT”. In: *IEEE Internet of Things Journal* 5.2 (2018), pp. 1184–1195.

[OK19] S. Oh and Y. Kim. “Interoperable OAuth 2.0 Framework”. In: *2019 International Conference on Platform Technology and Service (PlatCon)*. 2019, pp. 1–5.

[ope] openHAB.

[ope18a] openHAB. *openhab-distro*. 2018. URL: `https://github.com/openhab/openhab-distro/releases/tag/2.5.9` (visited on Sept. 25, 2020).

[ope18b] openHAB. *openhab1-addons*. 2018. URL: https://github.com/openhab/openhab1-addons/wiki (visited on Sept. 25, 2020).

[ope20a] openHAB. *openHAB*. 2020. URL: https://www.openhab.org/ (visited on Sept. 25, 2020).

[ope20b] openHAB. *openhab/openhab-docs*. 2020. URL: https://github.com/openhab/openhab-docs (visited on Sept. 25, 2020).

[ope20c] openHAB. *Securing access to openHAB*. 2020. URL: https://www.openhab.org/docs/installation/security.html (visited on Sept. 25, 2020).

[ope20d] openHAB. *Things Supported by the Z-Wave Binding*. 2020. URL: https://www.openhab.org/addons/bindings/zwave/doc/things.html (visited on Sept. 25, 2020).

[ope20e] openHAB Foudation. *Welcome to myopenHAB*. 2020. URL: https://www.myopenhab.org/ (visited on Sept. 25, 2020).

[OSG] OSGi Alliance. *OSGi, The Dynamic Module System for Java*. URL: https://www.osgi.org/developer/what-is-osgi/ (visited on Sept. 25, 2020).

[Oua+17] Aafaf Ouaddah, Hajar Mousannif, Anas Abou Elkalam, and Abdellah Ait Ouahman. "Access control in the Internet of Things: Big challenges and new opportunities". In: *Computer Networks* 112 (2017), pp. 237–262.

[Pyt20] Python. *time — Time access and conversions*. 2020. URL: https://docs.python.org/3/library/time.html (visited on Sept. 25, 2020).

[ras20a] rasberrypi. *Raspberry Pi 3 Model B+*. 2020. URL: https://www.raspberrypi.org/products/raspberry-pi-3-model-b-plus/ (visited on Sept. 25, 2020).

[ras20b] rasbian. *Welcome to Raspbian*. 2020. URL: https://www.raspbian.org/ (visited on Sept. 25, 2020).

[Ric+06] V. Ricquebourg, D. Menga, D. Durand, et al. "The Smart Home Concept : our immediate future". In: *2006 1ST IEEE International Conference on E-Learning in Industrial Electronics*. 2006, pp. 23–28.

[Sma20a] SmartThings. *Samsung SmartThings Add a little smartness to your things*. 2020. URL: https://www.smartthings.com/ (visited on Sept. 25, 2020).

[Sma20b] SmartThings Developers. *Authorization and Permissions*. 2020. URL: https://smartthings.developer.samsung.com/docs/auth-and-permissions.html#Best-practices (visited on Sept. 25, 2020).

[Sob20] C. C. Sobin. "A Survey on Architecture, Protocols and Challenges in IoT". In: *Wireless Personal Communications* 112.3 (June 2020), pp. 1383–1429.

[SQO19] S. S. Sabry, N. A. Qarabash, and H. S. Obaid. "The Road to the Internet of Things: a Survey". In: *2019 9th Annual Information Technology, Electromechanical Engineering and Microelectronics Conference (IEMECON)*. 2019, pp. 290–296.

[Thi20a] ThingsBoard. *Advanced Role-Based Access Control (RBAC) for IoT devices and applications*. 2020. URL: https://thingsboard.io/docs/user-guide/rbac/ (visited on Sept. 25, 2020).

[Thi20b] ThingsBoard. *Device authentication options*. 2020. URL: https://thingsboard.io/docs/user-guide/device-credentials/ (visited on Sept. 25, 2020).

[Thi20c] ThingsBoard. *OAuth 2.0 Support*. 2020. URL: https://thingsboard.io/docs/user-guide/oauth-2-support/ (visited on Sept. 25, 2020).

[Thi20d] ThingsBoard. *ThingsBoard Open-source IoT Platform*. 2020. URL: https://thingsboard.io/ (visited on Sept. 25, 2020).

[TM19] Antero Taivalsaari and Tommi Mikkonen. "Gateways to Heaven: Observations and Predictions on the Software Architecture of IoT Gateways". In: *Proceedings of the 17th International Conference on Advances in Mobile Computing Multimedia*. MoMM2019. Munich, Germany: Association for Computing Machinery, 2019, pp. 219–225.

[ubu17] ubuntu documentation. *UFW*. 2017. URL: https://help.ubuntu.com/community/UFW (visited on Sept. 25, 2020).

[Vel18] J. Velázquez. "Securing openHAB Smart Home through User Authentication and Authorization". In: 2018.

[Wik20a] Wikipedia. *JSON Web Token*. 2020. URL: https://en.wikipedia.org/wiki/JSON_Web_Token (visited on Sept. 25, 2020).

[Wik20b] Wikipedia. *OAuth*. 2020. URL: https://en.wikipedia.org/wiki/OAuth (visited on Sept. 25, 2020).

[Wik20c] Wikipedia. *Wi-Fi Protected Access*. 2020. URL: https://en.wikipedia.org/wiki/Wi-Fi_Protected_Access (visited on Sept. 25, 2020).

[WTI19] V. Williams, S. Terence J., and J. Immaculate. "Survey on Internet of Things based Smart Home". In: *2019 International Conference on Intelligent Sustainable Systems (ICISS)*. 2019, pp. 460–464.

[Zha+19] Y. Zhang, S. Kasahara, Y. Shen, X. Jiang, and J. Wan. "Smart Contract-Based Access Control for the Internet of Things". In: *IEEE Internet of Things Journal* 6.2 (2019), pp. 1594–1605.


ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

List of Acronyms

AES Advanced Encryption Standard

AI Artificial Intelligence

AP Access Point

AMQP Advanced Message Queuing Protocol

API Application Programming Interface

BLE Bluetooth Low Energy

BT Bluetooth

CoAP Constrained Application Protocol

DoS Denial of Service

GSM Global System for Mobile Communications

HTTP Hypertext Transfer Protocol

HTTPS Hypertext Transfer Protocol Secure

IIoT Industrial Internet of Things

IoT Internet of Things

IP Internet Protocol

IPv4 Internet Protocol version 4

IPv6 Internet Protocol version 6

JSON JavaScript Object Notation

JWT Json Web Token

LTE Long Term Evolution

MQTT Message Queuing Telemetry Transport

openHAB open Home Automation Bus

OS Operating System

OSGi Open Services Gateway initiative

REST Representational State Transfer

RFID Radio-frequency Identification

RPL Routing Protocol for Low-Power and Lossy Networks

SLA Service Level Agreement

SPOF Single Point of Failure

TCP Transmission Control Protocol

TLS Transport Layer Security

UDP User Datagram Protocol

URL Uniform Resource Locator

VPN Virtual Private Network

Wi-Fi Wireless Fidelity

WPA2 Wi-Fi Protected Access 2

XML Extensible Markup Language

XMPP Extensible Messaging and Presence Protocol

3G Third Generation

4G Fourth Generation

6LoWPAN IPv6 over Low-Power Wireless Personal Area Networks


ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

List of Figures


ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

List of Tables


ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

List of Algorithms


ATHENS UNIVERSITY OF ECONOMICS & BUSINESS
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ