Identity and consent in the internet of persons, things and services

Ivan Marco Lobe Kome

UNIVERSITE BRETAGNE LOIRE / MATHSTIC

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

THÈSE DE DOCTORAT DE

L'ÉCOLE NATIONALE SUPERIEURE MINES-TELECOM ATLANTIQUE BRETAGNE PAYS DE LA LOIRE - IMT ATLANTIQUE
COMUE UNIVERSITÉ BRETAGNE LOIRE

ÉCOLE DOCTORALE N° 601
Mathématiques et Sciences et Technologies de l'Information et de la Communication
Spécialité : *Informatique*

Par

Marco LOBE KOME

Identity and Consent in the Internet of Persons, Things and Services

Thèse présentée et soutenue à Rennes, le 11 février 2019
Unité de recherche : LabSTICC
Thèse N° : 2019IMTA0131

Rapporteurs avant soutenance :

Abdelmadjid	BOUABDALLAH	Professeur, Université Technologique de Compiègne
Pascal	URIEN	Professeur, Laboratoire Heudiasyc

Composition du Jury :

Président :

Guy	PUJOLLE	Professeur, Campus Pierre et Marie Curie, Sorbonne Université

Examinateurs :

Nadia	EL MRABET	Maître de conférences, Mines de Saint-Etienne
David	ESPES	Maître de conférences, Université de Bretagne Occidentale
Vincent	FREY	Ingénieur Chercheur, Orange Labs Rennes

Dir. de thèse :

Frédéric	CUPPENS	Professeur, IMT Atlantique
Nora	CUPPENS	Directrice de recherche, IMT Atlantique, (Co-directrice)

Invité(s) :

Adam	OUOROU	Directeur de recherche, Orange Labs Paris-Châtillon
Joel	EVAIN	Manager, Orange Labs Rennes

Dedicated to my mother, my sister.
Special dedication to my father, I miss you.

Abstract

The constant efforts of miniaturization of computing machines is transforming our relationships with machines and their role in society. The number of tiny computers remotely controlled is skyrocketing and those connected things are now more and more asked to do things on human behalf. The trend consists in making room for these specific machines into the Internet, in other words, building communication protocols adapted to their limited resources. This trend is commonly known as the Internet of Things (IoT) which consist of appliances and mechanisms different from those meant to be used exclusively by humans, the Internet of Persons. This conceptual separation being adopted, how would a Person exchange information with Things ? Sorts of brokers can help bridging that gap. The networking of those brokers led to the concept of Internet of Services. Persons and Things are connected through Services. This global networking is called the Internet of Persons Things and Services.

Connected Things are very special computers, they are meant to execute specific tasks and with the most autonomy possible. As a result, the communication interface with humans is poor. Other mechanisms must be created in order to seamlessly bind these conceptual areas: Internet of Things, Persons and Services. Our work is on the edge of these 3 Internet areas and our contributions are twofold.

In the first hand, we focus on new use cases involving Internet-connected devices and Persons. Unlike the current trend which consists in adapting Internet of Persons' security mechanisms to the Internet of Things and developing solutions aiming at making Persons ubiquitous by sharing their identity, our work consists in defining a device identity and a filiation relationship between a Person and its Internet-connected devices. To this end, first, we proposed a protocol allowing a Person to perform automatic purchases by delegating permissions to an Internet-connected device. The main issue to tackle being the secure biding of devices' and persons' identities. We formally proved that the protocol preserves the Integrity, Anonymity and Confiden-

tiality security properties. This first contribution defines a bridge between the Internet of Things and the Internet of Person through the delegation of authorization. As a result, it helps people regaining control over their data because unlike the current trend where Persons abide by Services' policies, the protocol is defined upon an architecture allowing Persons to define a refined access control policy. Second, to implement use cases like the previous one, we need an efficient Internet protocol layer. We proposed an Internet protocol over User Datagram Protocol (UDP) built around the 3 main IoT concerns: Advertisement, Synchronous and Asynchronous Requesting and Notification. This protocol is the second version of the Constrained Application Protocol (CoAP), strengthen with other IoT protocols assets.

On the other hand, we address the problem of the secrecy of data on constrained Internet-connected devices. First, we defined a security layer allowing two or more devices to establish cryptographic shared keys and cyphering communications without the traditional heavy and costly Public Key Infrastructure (PKI). As the number of Internet connected-device is expected to reach billions within the decade to come according Gartner [Gartner 2017], the challenge is to secure IoT communications with the widely used Diffie-Hellman algorithm, in a certificate-less architecture. Our solution exploits delegation of authentication solutions to authenticate parties and near field networking abilities of connected devices to build shared keys over private channels. Second, we proposed a solution to prevent from data exfiltration attacks on more powerful connected devices: Android smartphones. The challenge is to detect malicious programs nested on the user's computer and exfiltrating personal data on smartphones. The solution consists in adding a security layer on Android which job is to filter programs on the computer, aiming at accessing to the file system of the smartphone. With our solution, undetected malicious programs on the computer are detected. We formally proved that the proposed solution guarantees Secrecy and Integrity of data as well as Authentication of involved parties.

Finally, in this quest for a better integration of Internet connected-devices into the Internet of Persons, our work contributes to the definition of protocols on application and network layers, with IoT concerns and security at heart.

Résumé

La course à la miniaturisation des appareils informatiques est en train de transformer notre relation avec ces derniers, ainsi que leurs rôles dans notre société. Le nombre d'ordinateurs miniatures controlés à distance augmente considérablement et ces objets connectés - comme ils sont communément appelés - sont de plus en plus sollicités pour effectuer des tâches à la place de l'Homme. La tendance actuelle consiste à créer une place dans Internet pour ces objets connectés, autrement dit, à construire des protocoles adaptés à leurs ressources limitées. Cette tendance est connue comme l'*Internet des Objets* - ou l'acronyme anglais IoT - qui est différent des protocoles destinés à une utilisation exclusivement par des humains dit *Internet des Personnes* ou IoP en anglais. Avec l'adoption de cette séparation conceptuelle, comment est-ce qu'une personne échangerait ses informations avec des objets sans sacrifier la sécurité ? Pour aider à réduire cet écart, on a besoin d'un intermédiaire et la mise en réseau de ces intermédiaires amène à construire le concept d'*Internet des Services* ou IoS en anglais. Les personnes et les objets sont connectés à travers les services. Le réseau dans son ensemble, incluant les personnes, les objets et les services est donc l'**Internet des Personnes, des Objets et des Services**. Les objets connectés sont des ordinateurs spéciaux capables d'exécuter des tâches spécifiques avec la plus grande autonomie. Par soucis d'économie d'énergie et d'optimisation de taille, l'interface de communication avec l'humain est moins riche. D'autres mécanismes devraient être créés afin de relier les concepts d'Internet des Objets, des Personnes et des Services. Notre travail se situe à l'intersection de ces trois domaines et notre contribution est double.

Dans un premier temps, notre attention est dirigée vers les nouveaux cas d'utilisation impliquant les objets connectés et les personnes. Loin de la tendance actuelle qui consiste à calquer les mécanismes de sécurité de l'Internet des Personnes à l'Internet des Objets, par exemple, considérer qu'une personne et un objet devraient avoir la même identité numérique; notre travail définit l'identité de l'objet et la relation de filiation entre la personne et les objets connectés à Internet. Dans ce

but, nous proposons d'abord un protocole permettant à une personne d'effectuer des achats automatiques en déléguant les permissions à un objet connecté. Le principal problème à maitriser est l'association sécurisée de l'identité des objets et des personnes. Nous avons prouvé formellement que le protocole préserve les propriétés de sécurité d'Intégrité, d'Anonymat et de Confidentialité. Cette première contribution crée un pont entre l'IoT et l'IoS à travers la délégation de l'autorisation. Par conséquent, les personnes pourront reprendre le contrôle sur leurs données car la solution proposée est basée sur une architecture permettant aux personnes de définir une politique de contrôle d'accès plus fine.
Deuxièmement, pour mettre en place de nouveaux cas d'utilisation comme celui précédemment énoncé, nous avons besoin d'une couche protocolaire qui s'y prête. Nous proposons un protocole sur la couche applicative s'appuyant sur le User Datagram Protocol (UDP) et construit autour des trois principales préoccupations de l'IoT : la **Découverte** (Advertisement), les **Requêtes Synchrones et Asynchrones** et les **Notifications**. Nous proposons, d'améliorer le Constrained Application Protocol (CoAP) en le renforçant avec les atouts des autres protocoles très répandus de l' IoT comme MQTT et mDNS.

Nous abordons par ailleurs, le problème de la sécurité des données gérées dans les objets connectés dits contraints. D'abord, nous avons défini une couche de sécurité permettant à deux objets ou plus d'établir des clés cryptographiques partagées et de chiffrer les communications en s'affranchissant du poids et du coût supplémentaire de l'infrastructure à clés publiques (PKI). Selon Gartner [Gartner 2017], dans une décennie environ, le nombre d'objets connectés atteindra le milliard, ainsi le challenge est de sécuriser les communications de l'IoT, dans une architecture sans certificat, grâce au très répandu algorithme de Diffie-Hellman. Notre solution exploite la délégation des solutions d'authentification pour authentifier les parties, puis les capacités des objets connectés á avoir plusieurs cartes réseaux pour créer des clés partagées sur des canaux privés. Ensuite, nous proposons une solution qui déjoue les attaques d'exfiltration des données sur des objets connectés plus puissants comme les smartphones Android. La difficulté est de détecter les programmes malveillants implantés sur l'ordinateur de l'utilisateur et empêcher l'exfiltration des données personnelles de son smartphone. La solution consiste à ajouter une couche de sécurité sur l'appareil Android. La tâche qui filtre scrupuleusement les programmes voulant accéder au système de fichiers du smartphone. Nous avons prouvé que notre solution garantit le Secret et l'Intégrité des données, ainsi que l'Authentification des parties impliquées.

Finalement, dans la quête d'une meilleure intégration des objets connectés à Internet, nous avons contribué à la définition de protocoles autant sur la couche applicative que sur la couche réseau du modèle OSI, avec pour préoccupations principales les contraintes de l'IoT et la sécurité.

Acknowledgement

I would like to sincerely thank all people how have contributed in one way or another to the accomplishment of this work. These three years of thesis was a very uplifting experience, full of ups and downs but more importantly full of lessons.
First of all, I would like to thank from the bottom of my heart my advisors, **Frédéric Cuppens**, **Nora Cuppens-Boulahia**, and **Vincent FREY**, for their support, their dedication, their trust and their advices throughout these past three years. It has been an honor for me to be one of their Ph.D. students. I would like to express my gratitude to **Mariem GRAA** and **Adam OUOROU** for their precious advices and for their optimism that helped me in reaching this important goal.

I would also like to thank **Abdelmadjid BOUABDALLAH** and **Pascal URIEN** who had the hard task of reporting my thesis and giving their advice to improve its content.

I would like to thank my family, mostly my sister **Georgia**, her husband **Raphael**, **Arsène** my brother from another mother for their daily encouragement and endlessly support and affection. I could never have accomplished this thesis without them.

Finally, A huge thank to all my friends and colleagues, **Louis-Philippe SONDECK**, **Kevin CORRE**, **Yanhuang LI**, **Youssou NDIAYE**, **Lyes BAYOU** and **Fabien AUTREL**. A special thank to **Benoit HERARD**, this adventure started thanks to you. Thank you Marvin Johnson for involving me in your exciting projects.

Contents

CHAPTER 1

Introduction

The ever-reduction of printed circuits, processors and sensors prices is fostering the creation of tiny computers also named connected devices. They are expected to be more and more present to ease human lives and by the way, revolutionize the human-machine relationship. A basic example of such devices includes thermostats and HVAC (Heating, Ventilation, and Air Conditioning) monitoring and control systems that enable smart homes. Applications also include transportation, healthcare, industrial automation, and emergency response to natural and man-made disasters where human decision making is difficult.

We are now living the age of the Internet of Person Things and Services (IoPTS) according to Eloff et al. [Eloff et al. 2009]. It consists of billions of people, things and services having the potential to interact with each other and their environment. It is actually the meeting of three concepts:

The Internet of Person (IoP) which is envisaged as a world of interacting people and represented by social networks like Facebook or Twitter.

The Internet of Things (IoT) which is defined by the ITU as a world where physical and digital objects are seamlessly integrated into the same network (Internet) to become active participants [ITU 2005].

The Internet of Services (IoS) which is envisaged as the "webification" of services that we are witnessing through the democratisation of web Application Programming Interfaces (APIs).

The IoPTS is therefore the vision where people, things (physical objects) and services are seamlessly integrated into the network of networks as active participants exchanging data over a web-based infrastructure, about themselves and their perception of surrounding environments.

Depending on the use case we can distinguish two categories of protocols for connected devices: **short-range** and **long-range**. With the new world of opportunities brought by these new kinds of computers, comes also new security challenges because they can prove to be high risk breaches into systems. Consequently, some security properties must be watched closely. Especially, **identity** and **consent**. In fact, if one connected device equals to a person in terms of identity, then multiplying the number of devices attached to this person equals multiplying the number of back-doors to attach this person as well. Which causes a serious issue in both identity and consent managements.

1.1 Internet of Things Protocols

Lot of thrifty protocols are emerging to cope with the energy consumption of connected devices and without sacrificing the use-case experience. In this paragraph we stress on the most used or deployed protocols.

1.1.1 Short-Range Protocols

Short-Range protocols or *Last Miles Protocols* are meant generally for in-house use-cases.

Z-Wave

Z-Wave is a 50 meters range protocol designed for home automation applications mostly. Z-Wave devices communicate in a sub-gigahertz frequency range at a nominal rate of 20 kb/s with a 40 kb/s maximum bandwidth. Z-Wave uses the mesh architecture to expand the network and connect up to 232 appliances. It was designed to be easily integrated into battery operating devices. At the application layer, Z-Wave defines classes like binary switch, thermostat, alarm, controler, etc... The constructors of Z-Wave operating devices are gathered in the Z-Wave Alliance to structure and enhance the protocol. A future version called Z-Wave+ is a work in progress [Galeev 2006] [Thuresson 2006].

EnOcean

EnOcean is a 30 meters range protocol also made for home automation and buildings applications. The particularity of EnOcean technology is that wireless sensors, actuators and transmitters of the network are self-powered. This is possible thanks to slight mechanical excitations and other potentials from the ambiance using the principles of energy harvesting like motion, pressure, light and temperature. EnOcean devices are equiped with electromagnetic, piezo-generators, solar cells, thermocouples, and other energy converters to transform physical into electrical energy [EnOcean 2016b] [EnOcean 2016a].

Bluetooth

Bluetooth is a famous wireless technology for exchanging data over short distances. The creators of this protocol wanted to unify communication over all kind of computer machines. It was officially categorized as an IoT protocol with the release of the fourth version: Bluetooth LE as *Low Energy*. The Bluetooth Special Group (SIG) was then aiming at novel applications in the healthcare, fitness, security, home entertainment and system monitoring in smart industries. The last generation of this protocol is called Bluetooth 5 which range is up to 400 meters with a nominal rate of 2 Mbit/s. Since Bluetooth LE, the SIG defines models thanks to Generic Attribute Profile (GATT). This software model contributes to making of Bluetooth the most customizable IoT protocol [Gomez et al. 2012].

1.1.2 Long-Range Protocols

Long-Range protocols support long distance communications. They are adapted for server based use-cases. There is no Internet of Things without Long-Range protocols.

Wi-Fi (802.11ah / 802.11ax / 6LoWPAN)

Wi-Fi is a bundle of wireless IEEE standards (802.11 a/b/g/n) [Bhoyar et al. 2013] initially meant to replace the ethernet. Each of these standards comes with various pros/cons related to data speed, signal interference from outside sources, and cost. Cost is a factor because different hardware is needed for different standards, though newer versions are made to be backwards compatible with older versions. Because of its inter-operability, it has to be used for IoT. However, Wi-Fi is energy intense thus, connected devices using it need to be directly plugged into the power outlet. This

standard is a high frequency one and is designed to send a lot of data at once. Such a characteristic is needed for streaming content use-cases or other home security system use-cases.
However, there are attempts to make Wi-Fi more thrifty. We can cite:

- **IEEE 802.11ah** [Khorov et al. 2015] also known as Wi-Fi *Low Power* and *Halow*. It is an amendment of the IEEE 802.11 which operates on bands around 900 MHz with a lower rates, between 150 kbit/s and 40 Mbit/s.
- **IEEE 802.11af** [Flores et al. 2013] also known as *White-Fi* and *Super Wi-Fi* is another amendment of the IEEE 802.11 which operates on bands close to VHF and UHF (between 54 and 790 MHz). The figure 1.3 depicts the difference between 802.11ah and 802.11af.
- **6LoWPAN** [N. Kushalnagar 2007] [G. Montenegro 2007] [Raza et al. 2012a] stands for *IPv6 over Low-Power Wireless Personal Area Networks*. It is a simple low cost communication network that allows wireless connectivity in applications with limited power. It conforms to the IEEE 802.15.4 standard and is characterized by: a low bandwidth (up to 250 kbit/s), small packet size (81 octets for data packets), a support for both 16-bit short or IEEE 64-bit extended media access control addresses and both star and mesh topologies. It is often compared to Wi-Fi because they both aims at replacing Ethernet. But 6LoWPAN has always been IoT oriented. Figure 1.2 shows the 6LoWPAN layer towards other protocols.

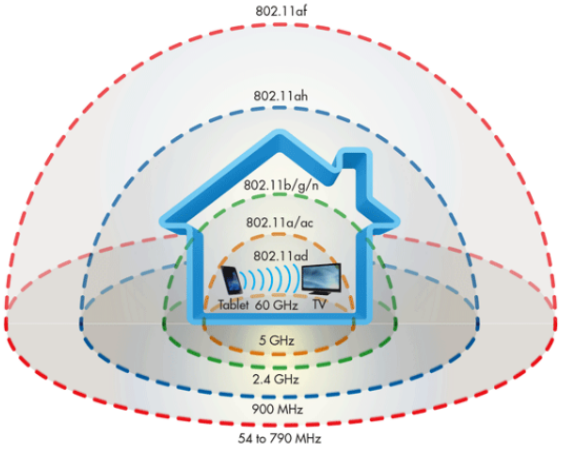


Figure 1.1 – Range difference between IEEE 802.11 amendments

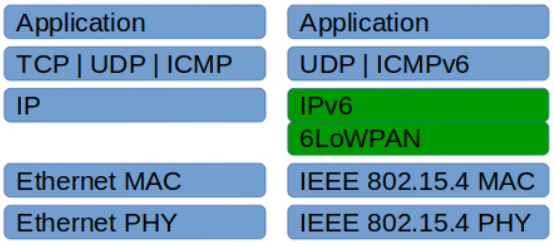


Figure 1.2 – Location of 6LoWPAN on OSI model

LoRa/LoRaWAN

The Long Range Wide-Area Network (LoRaWAN) is a 50 kbit/s rate protocol with a range which can go up to 15 kilometers. This range is possible thanks to gateways deployed in a star topology and connected to servers via IP connections on the Long-RAnge (LoRA) physical layer. The technology of this layer relies on a different modulation known as *chirp spread spectrum* [Berni and Gregg 1973] or *LoRA modulation*. The LoRa Alliance developed LoRa and LoRaWAN from the bottom up to optimize Low Power Wide Area Network (LPWAN) connections for battery lifetime, capacity, range and cost. Long Term Evolution (LTE) [Cox 2012] is another LPWAN technology and direct concurrent to LoRa, it is an evolution of GSM/EDGE norm developed by 3GPP consorptium. But for IoT purposes, this technology is being left behind to the benefit of LoRaWAN and Sigfox [Sigfox 2012]. Sigfox uses 200 kHz of the publicly available and unlicensed bands to exchange radio messages over the air. Unlike LoRa which defines its own modulation, Sigfox relies on Ultra Narrow Band (UNB) technology combined with DBPSK and GFSK modulation. Telecommunication companies like Orange are moving towards an integration of LoRaWAN into the 5G network.

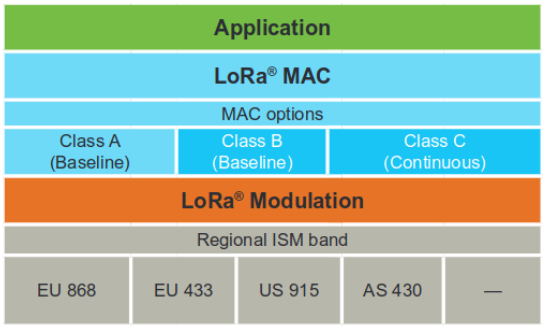


Figure 1.3 – Illustration of the components of the complementary LoRa and LoRaWAN layers within an application

1.2 Internet of Things Use cases and Security Challenges

The IoT offers a great market opportunity for equipment manufacturers, Internet service providers and application developers. By 2022, Machine-to-Machine (M2M) traffic flows are expected to constitute up to 45% of the whole Internet traffic [Evans 2011], [Gantz and Reinsel 2012], [Taylor 2013]. Moreover, McKinsey Global Institute reported that the number of connected machines (units) has grown 300% between 2008 and 2013 [Manyika et al. 2013] and Gartner estimated a 31% growth of connected things between 2016 and 2017 [Gartner 2017]. These figures are symptomatic of the tendency of manufacturers to provide a better Quality of Service (QoS) depending on client usage of the provided service and perceptions of the physical world throughout sensors.

1.2.1 Internet of Things Use cases

The cutting edge technology of the industrial world is more and more used to solve problems in the household world. Therefore, IoT mostly solves problems in what is known as **smart home**, **smart grid** and **smart health**.
In fact, the smart home has been of interest to researchers over the last 30 years. There is a low level of scientific consensus on the subject of smart home definition. Several authors like D. Zhang et al. [Zhang et al. 2013], and M.A.A. Pedrasa [Pedrasa et al. 2010], agree on the definition proposed by L.C.D. Silva et al. [De Silva et al. 2012] as a *"home-like environment that possesses ambient intelligence and automatic control"* capable of reaction to the behaviour of residents and to offer various accommodations. There is, according to Silva et al., four types of smart homes: healthcare based, multimedia and entertainment based, security based and energy efficiency based smart homes. Smart Home is the backbone which will enable the management and the control of different areas of a residence binding four pillars of human livelihood inside a house: comfort and welfare, physical integrity and facilities' safety, rational management of domestic equipment's energy and the possibility to provide healthcare services to its inhabitants. The critical aspect for this backbone to work is to possess a cheap, reliable and easy designed structure of communication. Therefore, Smart Home can be defined as a concentrator and disseminator of information and services that intends to cover the totality of a home's functional areas, this function being operational not only for the particular elements that are in the house in order to improve the levels of comfort and quality but also to provide a gateway or interface to the exterior by the means of an interaction with

other paradigms such as a **smart grid** [Zhang et al. 2011] [Lugo-Cordero et al. 2014] [Li and Liang 2013].
Essentially, the Home Area Network (HAN) [Mendes et al. 2015] is a fundamental framework to enable the exchange of information and interoperability among several smart domestic appliances connected to other devices or networks through many protocols, such as Bluetooth, ZigBee, WiFi, Z-Wave, etc. inside or within the close neighbourhood of a house. It is totally extendable to smart grid or smart city though, by replacing short-range protocols by long-range one like LoRa or Sigfox 1.1.2. The figure 1.4 depicts the main characteristics and requirements for a HAN. This classification also apply, according to us, to the smart grid.

Figure 1.4 – The classification of the main characteristics and requirements for a Home Area Network

1.2.2 Internet of Things Security Challenges

A key element conditioning the development of IoT is security. The problem is that there are a lot security properties to fulfill (Authentication, Authorization, Confidentiality, Integrity, Privacy, ...). It is a difficult challenge to consider all of them with the same level of interest in a single solution. A method to come close to a suitable solution would be the ranking of security properties according to the context (use-case). A good survey on *"Smart Home Communication Technologies and Applications"* by Tiago et al. [Mendes et al. 2015] suggests that in smart home oriented use cases, it is required to carefully consider **secure communication**, **network operation** and **dependability**. Secure communication holds if Integrity, Authentication and Confidentiality are guaranteed. According to Tiago et al. we can consider having a proper network opera-

tion when Interoperability, Maintainability and Self-Management are guaranteed. And an acceptable dependability property holds if Availability, Resiliency and Reliability are guaranteed.
For a better understanding:

Integrity is defined as the capacity to certify if a message that is being sent reaches the receiver intact must be done in an effective and accurate manner. The sender of a message must be able to prove that a specific message has been sent and if the receiver has indeed received the message.

Authentication is a requirement used by one node to identify another node or to verify the source of origin of data in the network. It is different from Identity but related to it. In fact, is authenticated someone or something already known.

Confidentiality must hold even if the network has been hijacked by a malicious third party. Message content must be kept between the sender and the intended recipients. Even the network operator should not be able to decipher messages unless he is part of legitimates receivers. IoT solutions shall provide adequate and strong encryption method, as recommended for example in RFC3565 for AES128 [Schaad 2003].

Interoperability should be considered to ensure the delivery of services for all customers regardless of the specifications of the hardware platform that they use. It should not matter either it is a Person behind a smartphone, a connected thing on a Raspberry Pie or a service on a quantum computer.

Maintainability essentially reflects how durable and reliable is the provided service. The environment can change, for example, a default in energy supply. In that case the device should be able to monitor its own health.

Self-Management . Various sensor network applications are programmed to operate with no infrastructure support or the prospect of maintenance and repair. Consequently, in order to configure themselves, operate and collaborate with others, to adapt to failures, changes in the environment or environmental stimuli, it is a requirement of the sensor nodes to be self-managed, which means to be completely independent of human intervention.

Availability . The purpose of this property is simply to guarantee that the services of network are always accessible and will still operate either when few failures occur or to operate quick restarts when failures take place.

Resiliency defines the recoverability and fault tolerance of a network. It clarifies, mainly from a safety and security perspective, the capability to restore and recover from a range of disruptions or malfunctions through the robust fast-response process, especially the vulnerable digital elements in the network. Wireless mesh networks are more vulnerable to possible node and link failures when compared with wired networks. Consequently, when incidents occur, the degree of resiliency defines how trustworthy the network can truly be.

Reliability aims to increase the success rate of service delivery, especially in IoT. It has a close relationship with availability as by reliability, we guarantee the availability of information and services over time. Reliability is even more critical and has more stringent requirements when it comes to the field of emergency response applications.

We think however that in an heterogeneous internet, involving Persons, Things and Services (IoPTS), being able to recognize the author of a message during a dialog is a critical security property in use cases involving actions like energy management or payment. The property described is also known as **Identity**. We are therefore, particularly in this thesis, in favor of concepts trying to define persons', devices' and services' identity to apply in an easier manner access control policies or to finally defend, the end-user privacy.
We also stress in this thesis, on the **Consent** security property which is defined by the General Data Processing Regulation (GDPR) [E.U 2018] as a clear affirmative action to grant access to data that must be demonstrated by the consent manager. The definition includes the right to withdraw that grant.

1.3 Related works

To our knowledge, all of the solutions aiming at solving identity related security issues in IoT, end up focusing more on access control based solutions, when we recommend to consider the requesting constrained device as a hole new class of machine. This help addressing in a different manner security issues. For a better understanding of this thesis, each contribution describes the background and works that have served as inspiration. If UMA, FIAM and FIDO, 3 of the most related works, differ from our proposal on many points, the common point is that the Open Authorization Framework is the core of their architecture.

1.3.1 Open Authorization Framework 2 (OAuth2.0)

The Open Authorization framework (OAuth) [Hardt 2012] is built on top of the HTTP protocol. It enables a third-party application to obtain limited access to a service on behalf of a resource owner by orchestrating an approval interaction between the resource owner and the service. It is implemented by the most used identity providers (Facebook, Twitter, Google). OAuth defines 4 actors or roles and 4 protocol flows. The general architecture is depicted in figure 1.5.

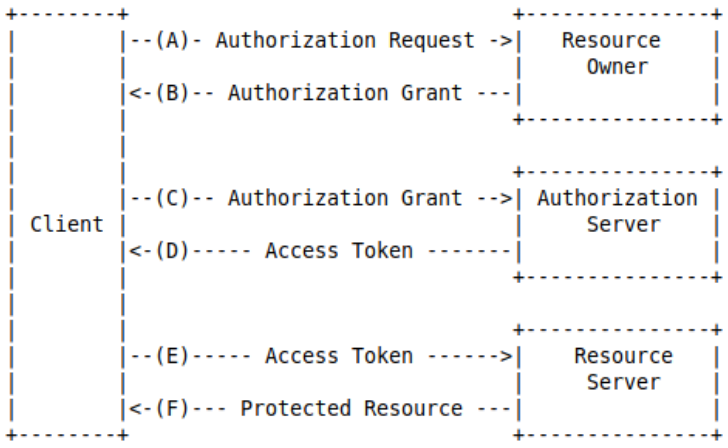


Figure 1.5 – OAuth abstract flow

OAuth Roles

- The **Resource Owner** is the entity which owns and is capable of granting access to the protected resource. It usually is a human-being and also referred to as end-user.
- The **Resource Server** which hosts the protected resources.
- The **Client** is the entity which requests for the protected resources on behalf of the Resource Owner.
- The **Authorization Server** is in charge of issuing the Client access tokens, provided that the Resource Owner is successfully authenticated.

OAuth Flows

The goal of OAuth 2 interactions is to give the Client a representation of the Owner's authorizations namely *access token*. Depending on the use case, OAuth 2 specification defines 4 ways for the Client to get authorizations from the Resource Owner.

- The **Authorization Code Flow**. This flow was made in cases where the Client and the Resource Owner are not sharing the same user-agent. In this flow the Authorization Server is an intermediary between the Client and the Owner in each round trip and Resource Owner's personal information are never shared with the Client.
  Instead of requesting authorization directly from the Resource Owner, the Client directs the Resource Owner to an Authorization Server via its user-agent, which in turn directs the Resource Owner back to the Client with the authorization code. The Authorization Server authenticates the Resource Owner and obtains authorization before directing the Resource Owner back to the Client and at last issues the access token to the Client of interest.

- The **Implicit Flow**. This flow is used in cases where Client's responsiveness and efficiency matter more than security. In fact, this case is a simplified version of the *code flow*. Instead of issuing the Client a code, the Client receives directly the access token from the Authorization Server without authenticating the Resource Owner. In this flow the Authorization Server verifies the Client by white-listing the address used to receive the access token and demanding the use of certificates to communicate with that address.

- The **Resource Owner Password Credentials Flow**. This flow is used when there is a high degree of trust between the Client and the Resource Owner. The resource owner password credentials like username and password can be used directly as an authorization grant to obtain an access token.

- The **Client Credentials Flow**. This flow is used when the Client is also the Resource Owner. Client credentials are used as an authorization grant when the client is acting on its own behalf or is requesting access to protected resources based on an authorization previously arranged with the authorization server.

OAuth 2 in extensible enough to define additional authorization flows. One connected devices have limited user-agents in comparison to devices used by humans (i.e smartphones, laptops), the flow which is the most adapted is *Implicit*. Considering that we have a user-agent for the device and for the end-user.

1.3.2 User-Managed Access (UMA)

The User-Managed Access [Maciej Machulak 2018a] [Maciej Machulak 2018b] [Maler 2019] defines a means for a client, representing a requesting party, to use a

permission ticket to request an OAuth 2.0 access token to gain access to a protected resource asynchronously from the time a resource owner authorizes access. UMA then defines the same actors as OAuth 2.0 but defines the **Requesting Party** as a new actor which is the legal or natural person that uses a Client to request for protected resources. The Resource Owner is like more like the legal User of the protected resources hosted by the Resource Server. The Authorization Server's role is still to protect the resources hosted by the Resource Server.
The process of delegation of authorization defined by UMA and depicted in figure 1.8, is similar to the OAuth 2.0 Authorization Code Flow. In fact, the authorization code is called *permission ticket* in UMA and 2 tickets instead of one are necessary to complete one UMA round-trip. Requesting Parties are identified with a unique token called RPT (Requesting Parties Token) which is associated with an array of claims on the Authorization Server. The end goal is to apply any access control model according to the use-case.

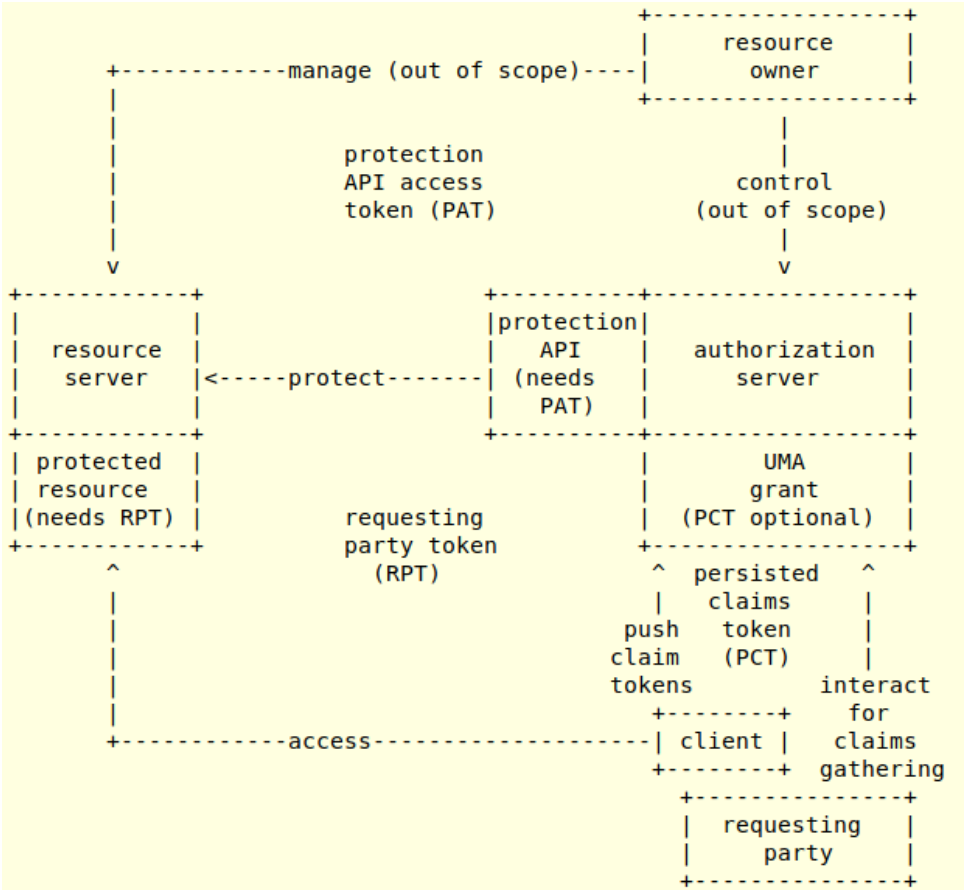


Figure 1.6 – UMA abstract flow

1.3.3 Federated Identity and Access Management for IoT (FIAM)

The proposal of the FIAM [Fremantle et al. 2014] is to combine two paradigms: Publish/Subscribe (Pub/Sub) and the Federated Identity Model (FIM) introduced in 2005 by Martin Gaedke et al. [Gaedke et al. 2005]. The FIAM authors chose respectively MQTT and OAuth for Pub/Sub and FIM implementations. The goal is to better include the owner of protected resources into the management of access control policies by notifying him. The Figure 1.7 depicts how the notification server (MQTT Broker) interacts with the Authorization server and connected device represented by an Arduino-prototyped connected object.

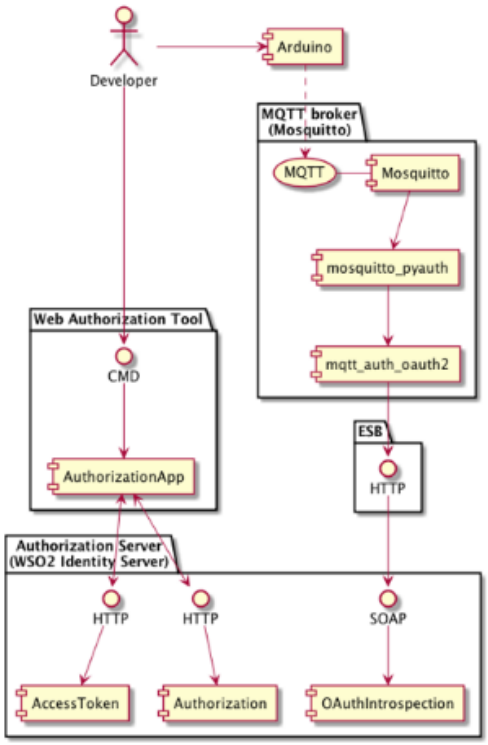


Figure 1.7 – FIAM component diagram

1.3.4 FIDO

The core ideas driving the FIDO Alliance's efforts are 1) ease of use, 2) privacy and security, and 3) standardization in use-cases involving powerful IoT devices like smartphones. FIDO architecture is built on top of 2 keys protocols:

- **Universal Authentication Framework (UAF) Protocol** [Machani et al. 2014]. The UAF protocol allows online services to offer password-less and multi-factor security. The user registers his/her device to the online service by selecting a local biometric authentication mechanism such as finger-print (swiping a finger), looking at the camera (face ID), speaking into the mic (voiceprint). Once registered, the users no longer need to enter their password when authenticating from that device.
- **Universal 2nd Factor (U2F) Protocol** [Sampath Srinivas 2014] [Balfanz 2015]. The U2F protocol allows online services to augment the security of their existing password infrastructure by adding a strong second factor to user login. The user logs in with a username and password as before. The service can also prompt the user to present a second factor device at any time it chooses. The strong second factor allows the service to simplify its passwords (e.g. 4-digit PIN) without compromising security. In FIDO the second factor device must be close to the user-agent using the service. It can be a USB or a NFC device

FIDO protocols (both UAF and U2F) are meant to be complement with Federated Identity Management (FIM) frameworks like OAuth or SAML. In fact, according to the use-case, a FIM party can leverage an initial authentication event at an Identity Provider (IdP).

1.4 Objectives and Contributions

We tackle in this thesis, the relationship between a person and one or many connected devices. In other words, we are working on a secure and seemless integration of the IoP and the IoT. More accuratly, we stress on how a person would delegate rights to a tiny machine, how security assertions are expressed and transmitted from persons to objects and vice-versa. Regarding the use-cases requiring the use of long-range protocols or simply IP-based like WiFi.
On one side, persons' and devices' identities are different and must clearly be identified to define a clear and secure relationship between the 2 kinds of identities. To meet that objective, we propose the following contributions:

Contribution 1. With connected things, one service can be used with more than one device, all sharing the same user identity. In this context, the need to figure out whether the service is being used through a desktop computer, a smartphone, or a

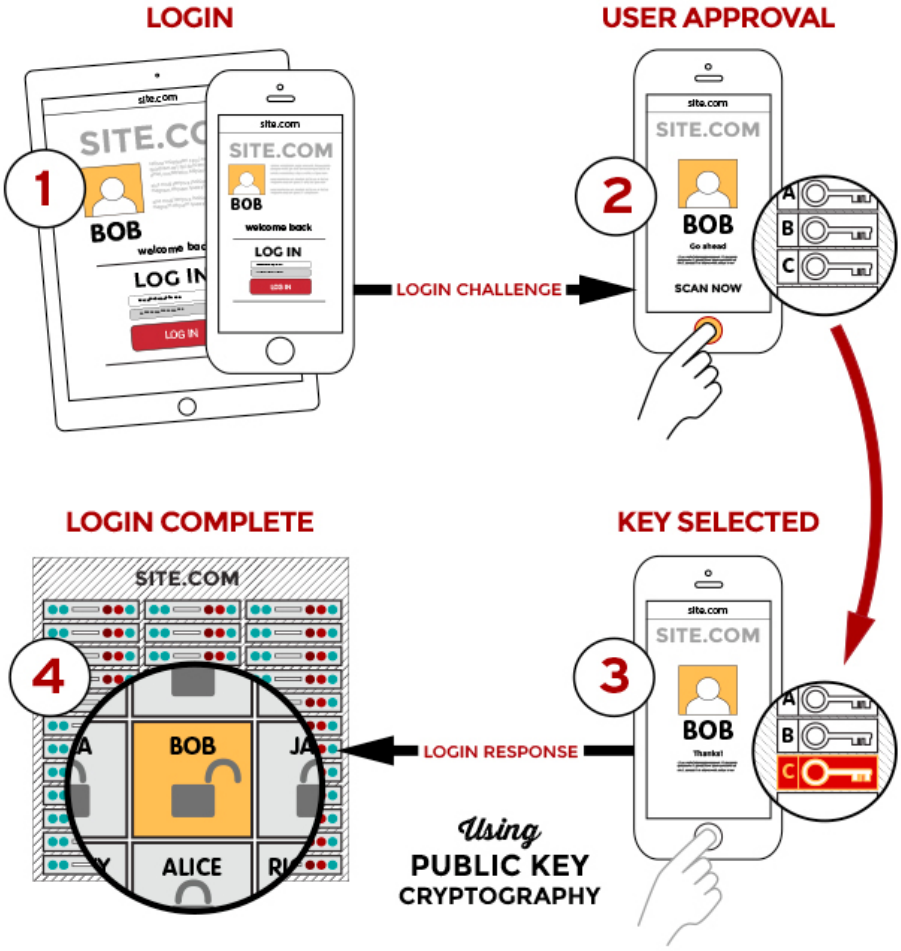


Figure 1.8 – FIDO login

more constrained device is essential in order to better manage user identity. Given that constrained devices are less tamper resistant, they are more vulnerable to attacks than other appliances. We identified two challenges which make it difficult to apply robust security mechanisms: the limited resources available on devices and the sharing of a user's identity with the device. To address these challenges, we propose, the **DIscovery and REgistration (DIRE)** protocol that ensures secure device and person identities management. Our protocol has been formally proven and implemented. The runtime of the whole protocol is short and the code the device must embed is lightweight. As a result of our experiment, we produced a command line client for a user, a device firmware and a server handling the filiation of a user and its devices.

Contribution 2. The need to connect things is skyrocketing and the Internet of Things is drawing a clear pattern of the need for smarter things. We are proposing an upgrade of Constrained Application Protocol (CoAP) built upon the 3 main

networking needs of a connecting thing: the discovery, the synchronous and asynchronous communications and the publish/subscribe. **CoAP 2.0** as we call it, will allow building smart things independently of the use-case, with a single protocol, fewer lines of codes and with less impact on the memory.

On the other side, some security mechanisms must be adapted or rethink to fit with constrained devices CPU limitations for cryptographic uses or tamper resistance. Meeting that objective led us to the following contribution:

Contribution 3. Diffie-Hellman key exchange is a popular cryptographic algorithm that allows Internet protocols to agree on a shared key and negotiate a secure connection. It is used in many protocols including SSH, IPsec, SMTPS, and protocols that rely on TLS. In the Internet of Things (IoT), we cannot rely on the PKI architecture to secure communications due to the growing number of connected things. We are proposing to decentralize the encryption keys management while maintaining the property of authentication and secrecy. We use the ability of each node to build a private channel to create a shared key, safe from the eye of an attacker. Our solution provides a means to build a certificate-less trusted ecosystem for IoT.

Contribution 4. Hackers can exfiltrate sensitive data stored from an IoT device such as Android smartphones. He/She can abuse the Android pairing mode and targets a personal computer system previously trusted by the device user. The existing protocols that allow files transfer from Android IoT devices to the computer cannot detect this attack. In this paper, we propose an approach to detect attacks exploiting trusted relationship between a third party system such as personal computer and an Android device to exfiltrate user data from the victim device to an attacker. We implement a protocol to secure communication between IoT Android device and third party system. Our approach has been proved to be effective in detecting this category of attacks with reasonable performance overheads.

1.5 Organization of the dissertation

This dissertation is divided into 2 parts described below.

Part I – Persons' and devices' Identity Biding in the Internet of Persons and Things – this part focuses on how should a Person be bond to its Internet-connected things in use cases like automatic payments. This part is driven by the 2 following assertions:

1. *"Consumers, not companies, own the data collected by Internet of Things devices"* by Limor Fried published in the "Bill of Rights" [Fried 2013]

2. *"Users must be empowered to execute effective controls over their personal information"* by Cavoukian when presenting the concept of *privacy by design* in "Privacy in the clouds" [Cavoukian 2008]

We have 2 chapters in this part:

Chapter 2 – Discovery and Registration Protocol for Device and Person Identity Management in IoT – we propose a protocol for secure biding of Persons' and Devices' identities. This architecture helps raising confidence between a person and its connected devices because it allows dynamic access control policies.

Chapter 3 – CoAP Enhancement For a Better IoT Centric Protocol – proposes to combine the strenghts from other popular protocols in IoT to build the HTTP for IoT.

Part II – Security Mechanisms for Constrained Devices – this part focuses on defining approaches allowing to specify and enforce security and utility requirements over IoT architectures. In this part we are more concerned about security requirements like consent, authentication and confidentiality.

This part is divided into 3 chapters:

Chapter 4 – A Certificate-less Key Exchange Protocol for IoT – proposes a method to encrypt communications without a Public Key Infrasture.

Chapter 5 – Detection and Response to Data Exfiltration from Internet of Things Android Devices – proposes to solve the problem of data exfiltration from Android devices by applying a process-filtering method.

Chapter 6 – Conclusions and Perspectives – this Chapter concludes the dissertation by summarizing the contributions and presenting the perspectives for future work.

Part I

Persons' and devices' Identity Biding in the Internet of Persons and Things

CHAPTER 2

DIscovery and REgistration Protocol: For Device and Person Identity Management in IoT

2.1 Introduction

The Internet of Things (IoT) constitutes a set of objects such as sensors, actuators, smart devices that communicate through the Internet to form a collection of identifiable smart entities [Gubbi et al. 2013a]. The number and variety of devices that are used to collect data have increased at an accelerated rate in recent years. According to a Cisco study [D.Evans 2011], the number of Internet-connected devices exceeded the human population in 2010. This number is expected to reach 50 billion in 2020.
In current architectures of the IoT, users share their credentials with connected objects to have access to services. But, the limited processing, memory and power resources of these objects make it difficult to apply security mechanisms such as high strength encryption and signature algorithms. As a matter of fact, the current authentication mechanisms of internet-connected objects can easily be bypassed and those devices can be infected by a malware such as Mirai [MalwareMustDie 2016], making them a botnet of large-scale network attacks [Hunt 2016] [Muresan 2016]. Therefore, balancing the utility of running internet-connected objects and the privacy risks is a critical challenge for the IoT.

On the other hand, when the connected object and the user have the same identity, the behavior of the device can be changed by the manufacturer without asking the user. As a real example, Fitbit users are unwittingly sharing details of their sex lives

with the world because the manufacturer makes this sexual activity public by default [thenextweb 2013].

Based on the two previously identified challenges: the limited resources available on devices and the sharing of a user identity with its devices, the important question that arises is the following: how can we securely bind devices' and persons' identities? In this chapter, we propose an original protocol that ensures secure device and person identities management. Our proposed protocol allows us to discover and register smart devices into an Identity Provider in a seamless and user-friendly way. We take our inspiration in discovery mechanisms like mDNS [Cheshire and Krochmal 2013] and in OAuth2.0 [Hardt 2012] for the registration process. Our contribution is twofold: the protocol adapts OAuth2.0 mechanisms to the IoT context and establishes solid foundations to ease access control policy management. In fact, managing device identity as easily as human identity is a strong base to dynamically adjust access control policy. Our protocol has been formally proven and implemented. As a result of our experiment, we produced a command line client for a user based on a zeroconf browser, a device firmware built with Arduino and a Python server handling the filiation of a user and its devices.

The rest of the chapter is organized as follows: section 2 introduces the background knowledge on which we base our approach. Section 3 introduces the protocol by defining prerequisites and depicting discovery and registration processes. In section 4, we deliver a formal verification that the protocol binds human and device identities in compliance with security requirements needed in the smart object networking context. In section 5, we present an implementation of our protocol based on a scenario and we discuss about the attacks detected by the protocol and the remaining work to prevent more. Finally, in section 7 we provide a comparison with the related works before concluding this chapter in section 8.

2.2 Background

2.2.1 ACE architecture

Our work is inspired by the security architecture provided by Authentication and Authorization for Constrained Environments (ACE) working group. They are proposing an architecture for authorization in constrained environments [Gerdes et al. 2016]. We take from this architecture the actors and redefine interactions between them. There are two reasons why the proposed protocol is based on ACE architecture: First of all, it takes into account the constraints on code size, state memory, processing capabili-

ties, user interface, power and communication bandwidth of some nodes. Secondly, the entity managing the device is also taken into account unlike other architectures.

Integrating those nodes in the IoT requires a definition of those main actors:

- The **Constrained Device** or **Thing (T)**. This actor is complex enough to embed a client and a server both limited by the device resources. The constrained client attempts to access one user's resources while the server is able to produce data related to a user.
- The **Manufacturer (M)**. Known by ACE as Requesting Party (RqP). This actor is in charge of the device's identity management. We grant the manufacturer all ACE Client Authorisation Server (CAS) attributes. Identity data collection is part of their business model so manufacturers are maintaining a strong link with their devices.
- The **Resource Owner (RO)** or **User (U)**. This actor is in charge of the resource management. It defines the access control policy of the resource server through a less constrained device like a smartphone or a desktop computer.
- The **Identity Provider (IDP)**. The IDP can embed a lot of functionalities, in this architecture this actor is in charge of the authentication and authorization of clients capable of having access to owner resources. It is seen as Authorization Server (AS). In many implementations, it also hosts owner resources so it is also considered as Resource Server (RS).

2.2.2 Open Authorization Framework 2 (OAuth2.0)

One of the challenges we are addressing in this chapter is to allow the constrained device to use owner's resources on its behalf. So we delegate authorizations from owner to device using OAuth framework depicted in section 1.3.1.

In our proposed DIscovery and Registration (DIRE) protocol, we adapt OAuth2.0 to the context of IoT as there are two different user-agents in the process. The missing consideration in OAuth implementations is the fact that one of the user-agents is constrained and considered a tiny machine. [Bradley and Denniss 2017] propose a new OAuth flow to cope with constrained devices but it does not consider a non-negligible amount of screenless devices.

2.2.3 The Things Description document (TDD)

We define device identity in the DIRE protocol based on the model proposed by the Simurgh framework [Khodadadi et al. 2015]. It is a framework whose goal is to make an effective discovery, programming and integration of services exposed in IoT. The framework introduces the *Things Description Document (TDD)*. It is a file responding to the JSON format with two main objects:

- **Entity properties** dedicated to describe properties of the entity with a name, location and last modified date. This information is mandatory.
- **Entity services** dedicated to the description of services the entity is exposing through RESTFul routes.

In the proposed protocol, the devices are declining their identity as if they are responding to the questions: *what am I capable of?* And *what do I need to achieve that?* The TDD must then inevitably contain the following items, organized as the JSON file 2.2.3 illustrates:

- A **thing_id**: a unique identifier for the device. It could be a MAC address for example.
- **capabilities**: an array of strings representing the list of what the device is capable of. An example could be: [temperature, gyroscope, gps]
- **intents**: an array of strings representing the list of what the device intends to do with the data. An example could be: [send-mail, read-mail, social-network-broadcast]
- **scopes**: an array of strings representing the list of R.O's data the device needs to fulfill the intent. An example could be: [email, contact, user-info]

Example of TDD

```
{
"properties": {
  "thing_id":"5E:FF:56:A2:AF:15",
  "name":"Connected flower pot",
  "description":"This is the description of the connected pot",
  "last-modified":"2016-07-20",
  "capabilities":[
    "temperature",
```

```
    "moisture",
    "luminosity"
  ]
},
"services":[
  "api":"connected_flower.raml",
  "intents":[
    "send-mail",
    "social-network-broadcast",
  ],
  "scopes":[
    "profile",
    "contact"
  ]
]
}
```

Properties and *Services* objects can be filled with additional information for a richer discovery and registration experience.

2.2.4 Message Queue Telemetry Transport (MQTT)

The MQTT [Andrew Banks 2014] is a standardized protocol designed to have a very low message overhead in order to ease constrained networks. It contributes to the growing interest in the IoT networks as it has been shown to use a significantly less battery power than HTTP polling for similar scenarios [Nicolas 2012].

MQTT protocol is also very interesting for our implementation because it is based around a publish/subscribe (pub/sub) model [Eugster et al. 2003]. This model is also very useful to make our downstream messages, from server to client. The pub/sub model works as following: clients can connect to one or more central servers (brokers) where they may publish information or subscribe to receive information, or both. Publishers and subscribers are decoupled from knowing about each other. They are also decoupled in time as interactions are asynchronous.

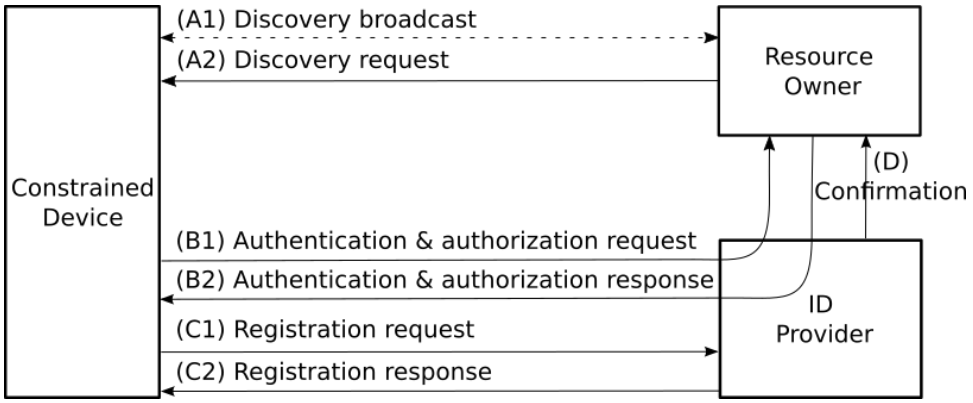


Figure 2.1 – Discovery and Registration protocol flow

2.3 The approach: description of the protocol

The proposed approach consists in discovering then registering a new connected device into the user's favorite identity provider. In order to achieve that, we use two important paradigms: The first one relies on the idea that each user and connected device has an identity. If we have an idea of what the user identity looks like, we must define a device identity with the TDD, prior to depicting the protocol. The second paradigm is based on the idea that the user and the device do not trust each other, therefore, each actor must show proof of their respective identity. We use OAuth2.0 to achieve that.

We built the DIRE protocol depicted in figure A.1, based on the two paradigms listed above.

The Resource Owner uses its user-agent to manage the whole process. It triggers the discovery (A) then the device uses the manufacturer privileges to ask the resource owner to authenticate itself to the ID Provider(B). The same request is used to ask for authorization to access resources. Once this mutual authentication is performed, the device requests for its registration to the ID Provider (C). Finally, the ID Provider sends to the Resource Owner a message about the registration process (D).

2.3.1 Discovery

The discovery phase depicted in the sequence diagram in Fig A.2, aims at being aware of all available devices' identity within a network. It is always triggered by a user (resource owner) through its user-agent by broadcasting its *user_id*.
For confidentiality purposes, both the user and devices will disclose public information until they are verified. While the user broadcasts its *user_id*, the device broadcasts a minimal version of the TDD (mTDD) which is a TDD without private information such as MAC address, API resource link.

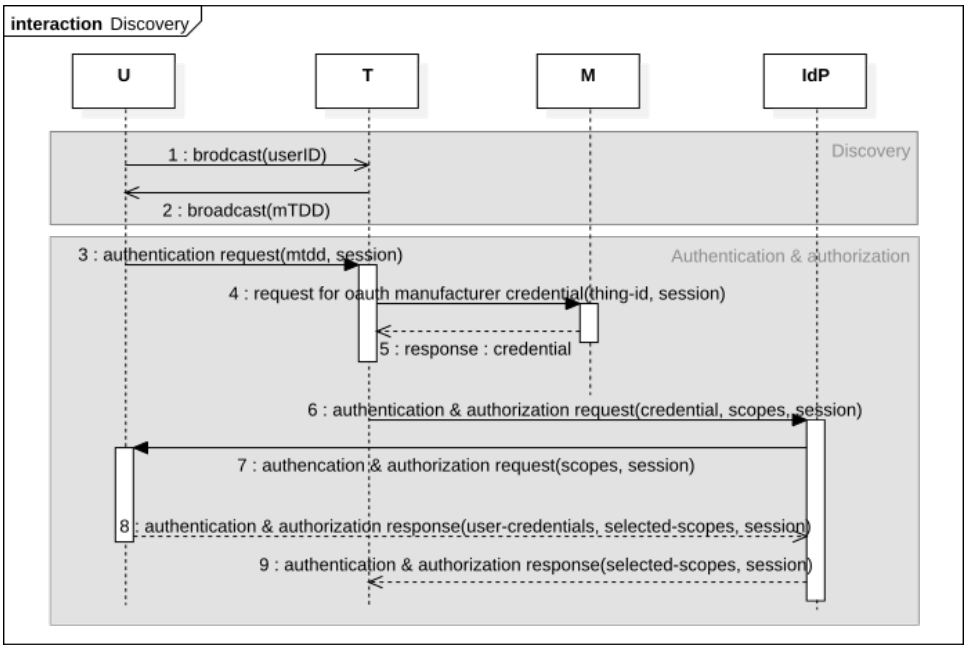


Figure 2.2 – Discovery sequence diagram

The principle of mutual authentication relies on the following assertion: *If the user receives an authentication request from its ID Provider - triggered by the device chosen to be registered - then, the device's identity is verified. And if the user correctly responds this request, then its identity is also verified.* At the end of the discovery process, the device will have authenticated the user and vice-versa.

The only role of the manufacturer at this stage is to disclose secrets, like OAuth client_id, to the requesting devices. The OAuth authentication request from the ID Provider to the user contains a list of services (scopes). So that, the user will be aware of the resources the device wants to access. The user can modify the scopes list before sending the OAuth authentication response.

Once the mutual authentication has been perfomed, the device triggers the second phase of the protocol.

2.3.2 Registration

The registration phase depicted in sequence diagram A.3, aims at binding user's and device's identities. It starts when the device requests for its TDD to the manufacturer.

The principle of registration stands in the following assertion: *the user's ID Provider must grant credentials to the device the user is introducing.* The user introduced the device in the discovery phase by disclosing its ID. Once the authorization grant is received, the device requests for its registration with its TDD as a parameter. The

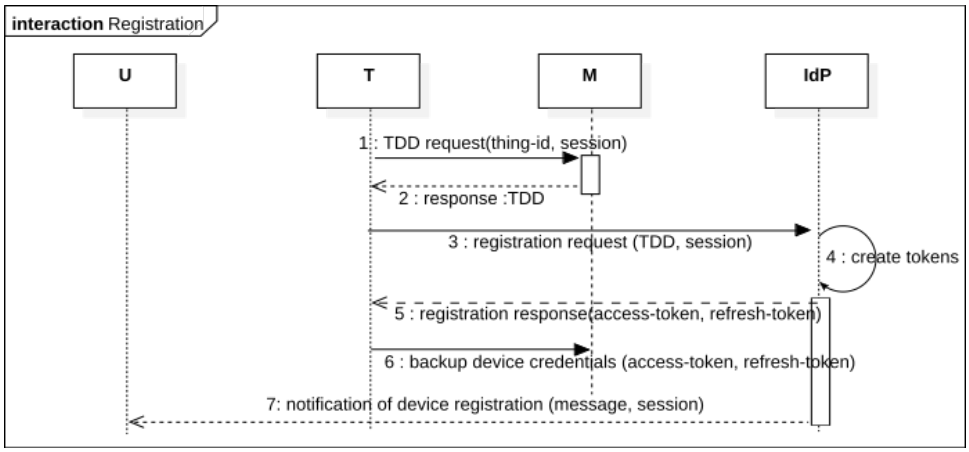


Figure 2.3 – Registration sequence diagram

ID Provider responds with the tokens (Access and refresh token) corresponding to the scopes the user validated. The tokens issued by the ID Provider are manufactured and meant to be used only by the device. That means that those tokens are associated to a TDD and a user in the ID Provider database.

At this stage, the manufacturer has two roles. The first role is to save device's credentials (tokens) in a much more secure database. Thanks to its priviledges the device will be able to fetch them when needed. The second role is to reconfigure the device server and client according to authorizations granted by the user.

At the end of this stage, the device is bound to the user and its actions have been set according to the rights granted by the user. The device can then act on behalf of the user with a separate identity.
A formal analysis of the DIRE protocol shows that our protocol reaches its goals while respecting the required security properties.

2.4 Formalisation

We present in the following, Event-B [Abrial 2008] [Butler and Yadav 2008] that we use to formally specify and analyse our approach and we describe the formal specification of the protocol.

2.4.1 Event-B

Event-B is a formal method for system modelling and analysis based on B method and associated to Rodin Platform [Abrial et al. 2010]. It is an Eclipse-based IDE for Event-B that provides effective support for refinement and mathematical proof.

Unlike other formal verification tools like ProVerif [Kobeissi et al. 2017] or Avispa [Armando et al. 2005], Event-B provides a very expressive notation rooted in predicate logic, arithmetic and set theory. Models in Event-B are described in terms of contexts and machines.

- **Contexts** contain carrier sets, constants, axioms, and theorems of a model. Axioms are statements assumed to be true in the rest of the model. They describe properties that cannot be derived from other axioms. Theorems present properties that are expected to be able to be derived from the axioms.
- **Machines** define dynamic behaviour. Machines contain the variables, invariants, theorems, and events of a model. Invariants allow defining interaction and security properties. We use them to define security properties of our protocol. A machine event represents a transition. It is composed of guards (the necessary condition for the event to be enabled) and actions (the way the variables of the corresponding machine are modified).

2.4.2 Formal specification of the protocol

We define two machines: the discovery and the registration machines. Both machines are working with the same context. We define the following sets in the Event-B context to formally specify our protocol.

Sets & Constants:

- USR, THG, MAN, IDP, as sets of, respectively, users, things (constrained devices), manufacturers and identity providers
- MODE as the set of different phases of the system: INIT, DISCOVERY, AUTHENTICATION, REGISTRATION, EXIT
- MSG as a set of messages exchanged between actors.
- PUBLIC_INFO and SECRET_INFO as sets of, respectively, public and secret information shared between the actors. Table 2.1 presents public and secret information.
- KEYS as a set of keys used to encrypt information.
- NONCE as a set of nonce.

Table 2.1 – Info classification

Actors	PUBLIC_INFO	SECRET_INFO
USR	id, user agent	credentials (pwd)
THG	id, capabilities, scopes	api
MAN	scopes	tdd, AT
IDP	id(domain)	AT, client_secret, client_id

Variables & Invariants:

We use the following variables and invariants to specify events and to formally define security properties. The cartesian product $\mathsf{S} \times \mathsf{T}$ denotes the set of pairs where the first element is a member of S and second element is a member of T. A partial function $\nrightarrow$ from S to T is a relation that maps an element of S to at most one element of T. If r is a relation between the sets S and T, the domain dom(r) is the set of the elements of S that are related to at least one element of T by r. Similarly, the range ran(r) is the set of element that are related to at least one element of S by r.

- ut_req, mi_vreq, tm_vreq, iu_auth_res, ui_reg_req present the type of messages that has been sent and received by actors in the discovery and the registration machines
  - $ut\_req \in ((USR \times PUBLIC\_INFO) \times NONCE) \nrightarrow MSG$
  - $mi\_vreq \in ((USR \times PUBLIC\_INFO) \times (THG \times PUBLIC\_INFO) \times (MAN \times SECRET\_INFO) \times NONCE) \nrightarrow MSG$
  - $tm\_vreq \in ((THG \times PUBLIC\_INFO) \times NONCE) \nrightarrow MSG$
  - $iu\_auth\_res \in ((USR \times SECRET\_INFO) \times (THG \times PUBLIC\_INFO) \times NONCE) \nrightarrow MSG$
  - $ui\_reg\_req \in ((USR \times SECRET\_INFO) \times (THG \times SECRET\_INFO) \times NONCE) \nrightarrow MSG$
- $umininfo$, $usecinfo$, $tmininfo$, $tsecinfo$, $mansecinfo$, $manmininfo$, $imininfo$ associate public and secret information to each actors
  - $umininfo \in user \nrightarrow PUBLIC\_INFO$
  - $usecinfo \in user \nrightarrow SECRET\_INFO$
  - $tmininfo \in thing \nrightarrow PUBLIC\_INFO$
  - $tsecinfo \in thing \nrightarrow SECRET\_INFO$

  - $mansecinfo \in man \nrightarrow SECRET\_INFO$
  - $manmininfo \in man \nrightarrow PUBLIC\_INFO$
  - $imininfo \in idp \nrightarrow PUBLIC\_INFO$

- $keyuser$, $keyth$, $keyman$, $keyidp$ attribute a key to each actor
  - $keyuser \in user \nrightarrow KEY$
  - $keyth \in thing \nrightarrow KEY$
  - $keyman \in man \nrightarrow KEY$
  - $keyidp \in idp \nrightarrow KEY$

- $knownnonce$ contains nonce that are already used in the session: $knownnonce \subseteq NONCE$

- $known\_user\_thing$, $known\_user\_man$, $known\_user\_IDP$ to indicate that the user is known by the other actors
  - $known\_user\_thing \in (user \times thing) \nrightarrow BOOL$
  - $known\_user\_man \in (user \times MAN) \nrightarrow BOOL$
  - $known\_user\_IDP \in (user \times IDP) \nrightarrow BOOL$

Events:

We use events to describe messages exchanged between two actors. There is one event for each message. We describe the necessary conditions for the message to be enabled in the guards. If these conditions are verified, we modify the message content. Let's consider for example the event where the device sends a message to the manufacturer, in order to trigger the user authentication. That corresponds to the arrow 4 in Fig.A.2. We have to make sure that the previous operation (arrow 3) was realized with the following assertion: $((u \mapsto umininfo(u)) \mapsto n) \in dom(ut\_req)$. We also have to make sure the user is not known by the device ($known\_user\_thing(u \mapsto t) = FALSE$) and that the nonce is already existing. Afterwards, the message is modified to contain user public information, device public information and a nonce. In addition, the message is encrypted before being sent.

manufacturer_creds_request $\widehat{=}$ **ANY** u, t, n **WHERE** $u \in user \wedge t \in thing \wedge n \in NONCE \wedge n \in knownnonce \wedge ((u \mapsto umininfo(u)) \mapsto n)) \in dom(ut\_req) \wedge known\_user\_thing(u \mapsto t) = FALSE \wedge mode =$

$AUTHENTICATION$ **THEN** $msg := tm\_vreq((t \mapsto tmininfo(t)) \mapsto n)$
$e\_msg := enc(keyth(t) \mapsto msg)$
END

We use events also to describe the phases transition in the protocol. At the beginning of the protocol, we set the variable *mode* to INIT. The only condition to satisfy to switch from INIT to DISCOVERY is formally expressed in the following guard of the *discovery_request* event (arrow 1 in Fig.A.2):

grd: $mode = INIT \wedge u \in user \wedge u \in dom(umininfo) \wedge (u \mapsto umininfo(u)) \in dom(ut\_req)$

We can switch from DISCOVERY to AUTHENTICATION mode when the user is not known by the device and received the mTDD beforehand. This condition is formally expressed in the following guard of the *discovery_response_unknown* event (arrow 2 in Fig.A.2):

grd: $mode = DISCOVERY \wedge known\_user\_thing(u \mapsto t) = FALSE \wedge (u \mapsto umininfo(u)) \in dom(ut\_req)$

There are two conditions to satisfy the switch from discovery to registration phase. Both are expressed in the guard of the *authentication_response* event. The first condition is that the device must have received the TDD and the authentication and authorization response coming from the IdP. The formal expression of that condition is the following:

grd: $\forall m, msg \cdot (m \in MODE \wedge msg \in MSG \wedge m = AUTHENCATION \wedge msg \in ran(ut\_res)) \wedge ran(iu\_auth\_res) \notin \varnothing \Rightarrow mode := REGISTRATION$

The second condition is that the previous message must have been encrypted with the manufacturer key and must come from the manufacturer. The formal expression of that condition is as following :

grd: $msg \in ran(dec) \wedge m \in dom(keyman) \wedge msg = dec(keyman(m) \mapsto e\_msg) \wedge (t \mapsto tsecinfo(t)) \in dom(tm\_vres)$

Intruder model:

We use the Dolev-Yao intruder model [Cervesato 2001]. We assume that the intruder can overhear, intercept, and modify any message exchanged between the actors. In

addition, we suppose that the adversary is only limited by the constraints of the cryptographic methods used. So, the adversary can try to know the devices attached to the user in order to extract user information provided by devices. In addition, the intruder can launch man-in-the middle attack to relay and possibly alter the communication between two actors. So, he/she can intercept and use the user public or secret information. Also, the adversary can initiate a replay attack to know devices attached to the user. Finally, the intruder can try to use the actor cryptographic key.

Security properties:

We define security requirements as invariants that hold before events are triggered.

1. **Integrity (Uniqueness of keys).** Each actor (user, thing, manufacturer and ID Provider) must have one and only one key. In both machines, uniqueness of keys is presented by the following invariants. We don't want the intruder to be able to use any actor cryptographic key. Therefore, we can detect the man in the middle attack because she/he cannot use the same key as another actor.

   - $\forall u, v, k, k1 \cdot ((u \in user) \wedge (v \in user) \wedge (k \in KEY) \wedge (k1 \in KEY) \wedge u \in dom(keyuser) \wedge v \in dom(keyuser) \wedge (k = keyuser(u)) \wedge (k1 = keyuser(v)) \wedge (k = k1) \Rightarrow (u = v))$
   - $\forall u, v, k, k1 \cdot ((u \in thing) \wedge (v \in thing) \wedge (k \in KEY) \wedge (k1 \in KEY) \wedge u \in dom(keyth) \wedge v \in dom(keyth) \wedge (k = keyth(u)) \wedge (k1 = keyth(v)) \wedge (k = k1) \Rightarrow (u = v))$
   - $\forall u, v, k, k1 \cdot ((u \in man) \wedge (v \in man) \wedge (k \in KEY) \wedge (k1 \in KEY) \wedge u \in dom(keyman) \wedge v \in dom(keyman) \wedge (k = keyman(u)) \wedge (k1 = keyman(v)) \wedge (k = k1) \Rightarrow (u = v))$
   - $\forall u, v, k, k1 \cdot ((u \in idp) \wedge (v \in idp) \wedge (k \in KEY) \wedge (k1 \in KEY) \wedge u \in dom(keyidp) \wedge v \in dom(keyidp) \wedge (k = keyidp(u)) \wedge (k1 = keyidp(v)) \wedge (k = k1) \Rightarrow (u = v))$

2. **Anonymity**. We define the anonymity property to detect the intruders that try to know devices attached to the user in order to extract user information provided by devices. To define anonymity property we verify that messages containing both user and thing information cannot be intercepted by the intruder. When nonce is used in the previous session of communication, this implies that the actor that sends the message is an intruder. In addition, we can detect the replay attack using anonymity property.
   In the discovery machine we have:

- $\forall u, t, m, n \cdot ((u \in user) \land (t \in thing) \land (m \in man) \land (n \in NONCE) \land (n \notin knownnonce) \land u \in dom(umininfo) \land t \in dom(tmininfo) \land m \in dom(mansecinfo) \Rightarrow ((u \mapsto umininfo(u)) \mapsto (t \mapsto tmininfo(t)) \mapsto (m \mapsto mansecinfo(m)) \mapsto n) \in dom(mi\_vreq) \land (knownnonce = knownnonce \cup \{n\}))$
- $\forall u, t, n \cdot ((u \in user) \land (t \in thing) \land (n \in NONCE) \land (n \notin knownnonce) \land u \in dom(umininfo) \land t \in dom(tmininfo) \Rightarrow ((u \mapsto umininfo(u)) \mapsto (t \mapsto tmininfo(t)) \mapsto n) \in dom(tm\_vreq) \land (knownnonce = knownnonce \cup \{n\}))$
- $\forall u, t, n \cdot ((u \in user) \land (t \in thing) \land (n \in NONCE) \land n \notin knownnonce \land u \in dom(usecinfo) \land t \in dom(tmininfo) \Rightarrow ((u \mapsto usecinfo(u)) \mapsto (t \mapsto tmininfo(t)) \mapsto n) \in dom(iu\_auth\_res) \land (knownnonce = knownnonce \cup \{n\}))$

In the registration machine there is only one relevant message:

$\forall u, t, n \cdot ((u \in user) \land (t \in thing) \land (n \in NONCE) \land n \notin knownnonce \land t \in dom(tsecinfo) \land u \in dom(usecinfo) \Rightarrow ((u \mapsto usecinfo(u)) \mapsto (t \mapsto tsecinfo(t)) \mapsto n) \in dom(ui\_reg\_req) \land (knownnonce = knownnonce \cup \{n\}))$

3. **Confidentiality**. We define the confidentiality property to detect intruder that tries to intercept and use secret information of user. We attribute secret information to user only if she/he is know by all other actors.
   In the discovery machine we have:

   $\forall u, t, m, idpro \cdot ((u \in user \land t \in thing \land m \in man \land idpro \in idp \land ((u \mapsto t) \in dom(known\_user\_thing)) \land ((u \mapsto m) \in dom(known\_user\_man)) \land ((u \mapsto idpro) \in dom(known\_user\_IDP))) \Rightarrow \exists secretinfo1, secretinfo2 \cdot (secretinfo1 \in secret\_info \land secretinfo2 \in secret\_info \land u \in dom(usecinfo) \land secretinfo1 = usecinfo(u) \land t \in dom(tsecinfo) \land secretinfo2 = tsecinfo(t)))$

   And in the registration machine we have:

   $\forall u, t, idpro \cdot ((u \in user \land t \in thing \land idpro \in idp \land ((u \mapsto t) \in dom(known\_user\_thing)) \land ((u \mapsto idpro) \in dom(known\_user\_IDP))) \Rightarrow \exists secretinfo1, secretinfo2 \cdot (secretinfo1 \in secret_info \land secretinfo2 \in secret\_info \land u \in dom(usecinfo) \land secretinfo1 = usecinfo(u) \land t \in dom(tsecinfo) \land secretinfo2 = tsecinfo(t)))$

Based on proof obligations (PO) generated by Rodin and that we have proved, all of the security properties modelled in the invariant were respected in the events and preserved by the protocol. Therefore, our protocol ensures uniqueness of keys, anonymity and confidentiality to all actors.

2.5 Implementation: application of the protocol

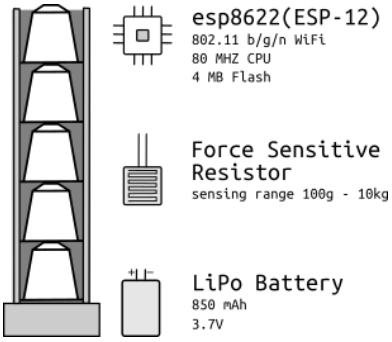


Figure 2.4 – Coffee Supplier (CS)

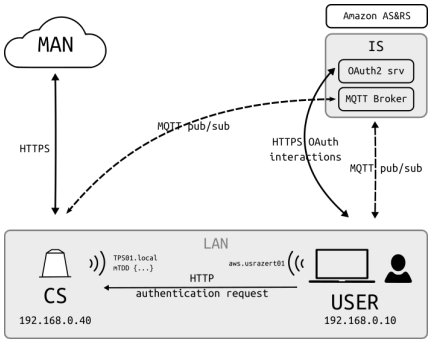


Figure 2.5 – Implementation architecture

In order to illustrate our approach, we consider the case of automatic refills. The owner resources at stake here is its wallet. The use-case consists in never running out of coffee by automatically purchasing coffee capsules on Amazon, each time there is only one capsule remaining. The architecture in Figure 2.5 shows the actors involved in the scenario.

We use MQTT in a majority of exchange messages to take advantage of asynchronous interactions. It is very suitable to use because there are a lot libraries available in various languages and platforms implementing the last version of the protocol. For microcontroller-based systems, we use Arduino Client for MQTT [O'Leary 2009]. On the server side, we use eclipse paho-mqtt-python [Eclipse 2014], as well as for the user application in command line. We secure communication channels using of TLS.

We use Flask-OAuthlib [Yang 2013] to implement the OAuth server. It provides a convenient framework to deal with OAuth interactions, tokens and errors.

As the originality of this protocol resides on the OAuth authentication with two different user-agents, our implementation focuses, on one hand, on interactions between the device and the user's client. On the other hand, we focus on interactions between the user's client and the Identity Provider.

2.5.1 User's client - device interactions: zeroconf discovery

Applying this protocol requires having total access of the constrained device resources and of the device - manufacturer architecture. We then made a custom connected device we called: Coffee Supplier (CS). It is made of components depicted in Figure 2.4. The device firmware is made of:

- an mDNS server where the device hostname and the mTDD are configured.
- an MQTT client for asynchronous interactions.
- an HTTP client. We use this client for non-asynchronous interactions. Those are exclusively for communication with its manufacturer.

What we implemented on the user's client to run the discovery is the following: the user's client looks for available devices, in the Local Area Network (LAN), with a mTDD object set in the mDNS service properties. The user chooses in the available devices, the one to register, which triggers the authentication request as illustrated in Listing 2.1. We are using the following formalism for the user_ID: $\langle IDP \rangle\_\langle userName \rangle\_\langle nonce \rangle$, which communicates the device registration IDP to the device itself. The nonce is the session ID used for the whole process session.

Listing 2.1 – Discovery interactions between the TPS and the user's client

```
Starting ...
Looking for available devices ...
Available devices (1) :
1.  Name: CS | Description : Coffee Supplier | Update : 2016-07-20
    Capacities: pression
    Authorizations: user-profile, payment
    Actions: purchase

Select the device you want to register: 1
You want to register the device : coffeesupplier.local
starting authentication ...
Successfully connected to broker : localhost
Subscribed to topic : /authN/usr01/azerty123
Starting authentication...
*   Trying 192.168.1.31...
* Connected to coffeesupplier.local (192.168.1.31) port 80 (#0)
> GET /authentication/aws_usr01_azerty123 HTTP/1.1
> Host: coffeesupplier.local
```

```
> User-Agent: curl/7.47.0
> Accept: */*
>
< HTTP/1.1 200 OK
* no chunk, no close, no size. Assume close to signal end
<
* Closing connection 0
```

2.5.2 User's client - identity provider interactions: authentication and registration

As Amazon does not support this architecture we implemented what we call an Identity Server (IS) which has the role of handling filiation by forging a derivative access and refreshing token for each device bound to the user. The server makes the filiation after having authenticated the user and getting his consent to grant the rights expressed in the scopes. Tokens to derive are delivered by Amazon Authorization Server (AS) and are forged for the manufacturer.
As depicted in Fig 2.5, the IS is made of:

- an MQTT broker to handle asynchronous messages.
- an OAuth2 server to make OAuth interactions.

The client wants to access Amazon owner's resources and it needs an Access Token (AT) for that. This AT is issued by Amazon AS if the client requests for it with a valid client_id. The only way to get a client_id is to be registered on Amazon AS, as the OAuth specification requires. The problem here is that the constrained device, by definition, is not resourceful enough to securely store client_id, client_secret, AT and Refresh Token (RT). Our solution, described in Fig 2.7, is the following:

In the DIRE authentication phase, the device wants to authenticate and ask for the consent of the resource owner at the same time. To do this, the device needs an AT, which it can acquire by using a manufacturer secret: its client_id. After having proceeded to the authorization request, the IS receives the AT and RT, manufactures a thing_AT and thing_RT with the formalism $\langle thing\_id \rangle . \langle AT \rangle$ and $\langle thing\_id \rangle . \langle RT \rangle$. These derived tokens are sent to the device in exchange of the TDD. The device then securely store them on its manufacturer database.

The DIRE registration phase consists in getting the complete identity of the device: the TDD, and appends the device to the list of user devices. The user's client is

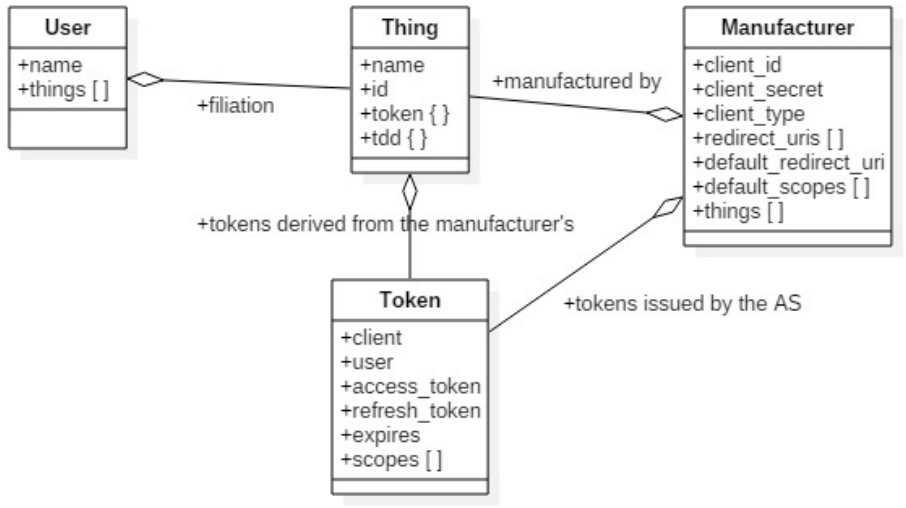


Figure 2.6 – Class diagram of the IS implementation

notified at the end of the process so that it can destroy the nonce. Fig 2.6 explains the architecture of the implemented IS.

There are two steps out of the scope of this implementation. Firstly, the manufacturer have delivered the connected device a *thing_id* and a *thing_secret* for the device to authenticate to the manufacturer. Secondly, the manufacturer have registered itself on Amazon AS as a constrained client. It then gets a client_id and a client_secret.

2.5.3 Results

Overall, the system works as intended and shows the following aspects:

- We can keep track of the use of the Amazon API for coffee replenishment as the IS clearly identifies which device did what.
- The system is deterministic. Each error message is notified to the owner. There is no blocking state as the formal verification demonstrates.
- Discovering and registering a new device takes less than 30 seconds. The owner has 2 attempts to rightfully authenticate itself.
- Messages can't be replayed as each session key is generated randomly and the user's client and the IS are keeping a track of used session keys.
- The firmware uses 2046 kB of flash. Which leaves a good amount of memory to build more complex connected device.

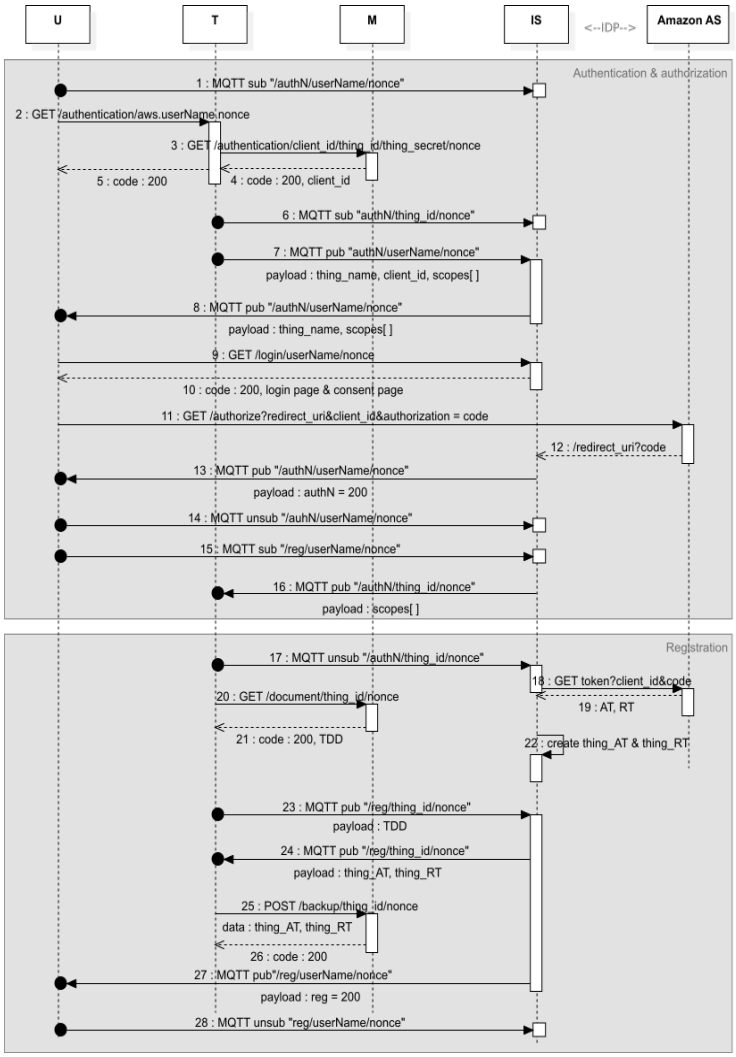


Figure 2.7 – Sequence diagram of the implementation

Meanwhile, the implementation can be further enhanced.

2.5.4 Discussion

We identified two ways to enhance the protocol implementation.

Enhance the discovery. The device have to embed two clients: an HTTP and an MQTT one. Because there is a need for a direct relationship between the manufacturer and its devices in the protocol, HTTP would be preferred. In fact, it doesn't require a third party (broker) such as MQTT does.

Enhance OAuth implementation. In the implementation depicted in Fig 2.7, we can see that the authentication and authorization request of the DIRE protocol

requires two interactions. In fact, we send an MQTT message to the user's client which then requests for authentication using an http client. We did it this way because OAuth2 with MQTT is not adopted yet by IDPs.

We propose, to tackle both issues, to implement the DIRE with another efficient protocol for IoT like the Constrained Application Protocol (CoAP) [Raza et al. 2013]. In fact, CoAP is able to deal with both synchronous and asynchronous messages, that also may be the reason that motivates the ACE group to define a specification for Authentication and Authorization for Constrained Environments using OAuth 2.0 and CoAP [ACE 2017].

2.6 Related works

There are a lot of solutions addressing the challenges the DIRE protocol is dealing with. One of those solutions is the OAuth 2.0 Internet of Things (IoT) Client Credentials Grant [Tschofenig 2014]. This chapter proposes to trade HTTPS for CoAP and DTLS [Raza et al. 2012a] [Erdtman 2016] [Kothmayr et al. 2013] to avoid communication latency due to the use of HTTPS. But building an effective encrypted channel only helps to cope with some node resource limitations. In fact, the case where the resource owner and the constrained device are using two different user-agents is not taken into account. In addition, the notion of device identity is missing.

Other solutions derived from OAuth are the one depicted in section 1.3: User Managed Access (UMA) 1.3.2 and Federated Identity and Access Management (FIAM) for IoT 1.3.3. UMA is really inspiring as the aim is to give back the end-user control over his data. But the notion of device identity is missing for both of solutions. Each device is always assumed to be the resource owner. Furthermore, it is assumed that the end-user has an exclusive access to the connected device whereas she/he is involved in a tripartite relationship with the device and the manufacturer. In FIAM for IoT, the solution consists of combining OAuth with MQTT before injecting access control management. As it is depicted in the FIAM results section, their approach requires the device to have a user interface with a browser in order to renew device authorizations. In addition, unlike FIAM, with the DIRE protocol the owner is able to change the scopes to grant the device specific rights.

The Delegated CoAP Authentication and Authorization Framework (DCAF) [Gerdes et al. 2014] depicts in an IETF draft a solution very similar to the one we described in this chapter. But DIRE takes into account the discovery where DCAF does not. Plus, there is no link with the user identity and it is specifically designed for CoAP. The same remark applies to IoT OAuth based Authorization Service architec-

ture (IoT-OAS) [Cirani et al. 2015], this protocol does not link the user identity to his devices. This solution stresses more on access control management.

Other related works include the OAuth 2.0 Device Flow for Browserless and Input Constrained Devices [Bradley et al. 2017]. This draft is also very close to the proposed protocol as the device and the user are discussing through two different user-agents. The device authenticates itself first to the authorization server before requesting for resource owner permissions. One remaining problem is that the device must have a screen to interact with the owner.

These related works clearly suggest that the use of OAuth together with IoT devices is worth exploring. Still, there are no solutions proposing to clearly establish a filiation relationship between a resource owner and its devices.

2.7 Conclusion

In this chapter, we propose an approach to better manage a person identity by decoupling a person identity from its devices. The proposed DIRE protocol allows a connected object to act on its owner's behalf. It also allows Manufacturers to build more trusty and interoperable devices. We have proved that our protocol respects the integrity, anonymity and confidentiality properties.
As a result of our experiment, we clearly identified and implemented the element each IDP needs to add in order to solve identity problems in IoT: the IS. We also showed that constrained devices have enough resources to operate the protocol.
This experiment shows encouraging results. We propose, as a recommendation, to standardize that approach because it is a solid foundation for identity management in IoT. For future research emerging from this work, we wish to investigate the management of access control policies in the context of IoT. We also propose to enhance the implementation of this use-case with CoAP.

CHAPTER 3

CoAP Enhancement For a Better IoT Centric Protocol: CoAP 2.0

3.1 Introduction

The growing number of things connected to the Internet is drawing a pattern showing the behaviors and the resources needed by smart things. Let us consider the automatic refill use-case depicted previously in 2.5. It consists in never running out of coffee by automatically purchasing coffee capsules on Amazon, each time there is only one capsule remaining. A connected device named *Coffee Supplier* is using its sensors to count the number of capsules, and the Wi-Fi to order and notify the device owner. We observe that the Coffe Supplier, like constrained nodes [Bormann et al. 2014] in general, has common behaviors that can be summed up in 3 concepts:

- **Advertising** themselves to be seamlessly discovered. Constrained nodes need to have a zero-configuration setup.
- **Notifying** data to a given address or subscriber.
- Providing **synchronous and asynchronous communications**

Another important concept needed to secure IoT systems is **access control**. In fact, we want connected devices to be aware to whom data is being delivered. For example, depending on the security policy, not everybody is either allowed to know the number of capsules delivered by the Coffee Supplier or needs to be aware of the constrained device presence on the network.
The current method to implement the Coffee Supplier use-case requires first to get all IP addresses of nodes on the local network and second, for each node available, looks for the one with the parameters corresponding to the wanted device. Depending on

the number of available nodes, discovering nodes this way can prove to be very long. Zero-configuration protocols like Bonjour [Apple 2003] or mDNS [Apple 2013] are used to perform that part. Moreover, to notify the user and make orders, Pub/Sub protocols like MQTT [Andrew Banks 2014] are recommended. This implementing method is costly in terms of memory and battery consumption.
Meanwhile, implementing that kind of use-case can be more efficient using a single protocol built for this purpose. The Constrained Application Protocol (CoAP) [Shelby et al. 2014], among all IoT protocols, is the best fitted to efficiently realize that kind of use-case. Each of the protocols used in IoT has its own strengths and weaknesses as depicted in table 3.1. mDNS and CoAP are built on top of UDP. But, when mDNS has no other purposes than advertising, we choose CoAP over MQTT as a protocol to enhance because it generates lower additional traffic than MQTT to ensure message reliability [Thangavel et al. 2014].

Table 3.1 – Protocols strengths and weaknesses.

Protocol	Advertising	Sync. & Async.	Notification
mDNS	++	- -	- -
MQTT	- -	+ -	++
CoAP	+ -	++	+ -

++ built for that. +- can be done but not efficient. - - inexistent functionality.

We propose and implement a protocol each IP constrained device should be talking because it is built with IoT concerns at heart. Those concerns are: a) dynamic advertisement and discovery, b) synchronous and asynchronous requesting and c) notification. If *Observe* [Hartke 2015], the solution built to fulfill the notification function, is well documented and implemented in the literature, it is not the case for concerns a) and b). Our protocol redefines the Discovery and the Publish/Subscribe functionalities in CoAP.
The rest of this chapter is organized as follows: Section 3.2 discusses the existing protocols that are used in IoT to fulfill Advertisement, Direct and Indirect communications and Publish-Subscribe. Section 3.3 presents the proposed CoAP 2.0. Section 3.4 provides implementation details. We give a performance evaluation and a security analysis of our approach in section 3.5. Finally, section 3.6 concludes with an outline of future works.

3.2 Background and Related Work

mDNS, MQTT and CoAP are the main protocols used in IoT. In this section, we introduce them, expose their strengths and weaknesses and why each of them is interesting for our proposed protocol.

3.2.1 multicast Domain Name System (mDNS)

The mDNS protocol [Apple 2013] is a zero-configuration [Palmila 2007] [Apple 2005] service using UDP paquets to perform DNS-like operations on the local link in the absence of any conventional Unicast DNS server. The main advantage of this service is that the DNS resolution works independently from any infrastructure and it requires little or no administration or configuration to be set up. There are popular implementation names like Bonjour by Apple [Apple 2003], the open source Avahi software packages [Avahi 2010] or the Network Service Discovery (NSD) from Android [Google 2018]. To achieve the zero-configuration goal mDNS sets the following parameters by default:

- Exclusively resolves host names ending with the *.local* domain
- Multicast UDP packet are sent to 224.0.0.251 IPv4 address or FF02::FB IPv6 address
- Multicast default port is *5353*

When an mDNS client needs to resolve a host name *hostname.local*, it sends a query message asking for the host having that name to identify itself. That request is sent to the default multicast address. The target machine then multicasts a message that includes its IP address. All machines in that subnet can then use that information to update their mDNS caches. mDNS Query and Resource Record are depicted respectively in Table 3.2 and 3.3.

mDNS was designed to perform node discovery within a given network in a very simple and lightweight manner. Therefore, synchronous and asynchronous applications and Notification are the weaknesses of this protocol according to our need. On one hand, this protocol was designed so that the implementation is easy and efficient, but on the other hand, the discovery abides by a simple binary logic: either a node is discoverable on the network or not. This protocol was not meant to handle access control at all. Our proposed protocol is filling that gap by modifying the mDNS query.

Table 3.2 – mDNS Query Fields.

Field	Description	Length/Bits
QNAME	The hostname to resolve	variable
QTYPE	Type of resource expected in the response. Examples: URI, TXT, MX, ...more on [Eastlake 3rd 2013]	16
U-R	Boolean flag	1
QCLASS	Set to 'IN' for Internet. More on DNS IANA Considerations [Eastlake 3rd 2013].	15

U-R stands for Unicast-Response. If set to 1, responders should send a directed-unicast response to the requesting node instead of broadcasting the response to the entire network.

Table 3.3 – mDNS Resource Record (RR) Fields.

Field	Description	Length/Bits
RRNAME	The hostname to resolve	variable
RRTYPE	The type of resource record	16
CACHE-FLUSH	Boolean flag	1
RRCLASS	set to 'IN' for Internet	15
TTL	Time interval cached by RR	15
RDLENGTH	Integer, RDATA length	15
RDATA	Resource data of RRTYPE structure	15

RRTYPE is QTYPE. RRCLASS is QCLASS. If CACHE-FLUSH is set to 1, outdated cached records should be purged.

3.2.2 Message Queue Telemetry Transport (MQTT)

MQTT [Andrew Banks 2014] is a standardized protocol designed to have a very low message overhead in order to ease constrained networks. It contributes to the growing interest in the IoT networks as it has been shown to use a significantly less battery power than HTTP polling for similar scenarios [Nicolas 2012]
MQTT was designed around the *topic-based* Pub/Sub philosophy, which is fitted for only a certain type of event pattern but not all of them. In fact, there are 3 famous Pub/Sub schemes, namely *topic-based*, *content-based* and *type-based*. They are depicted in detail in "The many Faces of Publish/Subscribe" [Eugster et al. 2003] published

in 2003. Figures 3.1, 3.2 and 3.3 illustrate each scheme. The main idea behind each pub/sub concept is the following:

- **Topic-based** scheme is based on the notions of *Topics* or *Subjects* which are very similar to the notion of *Groups*. Network actors can publish events and subscribe to individual topics, which are identified by keywords. This scheme is very easy to understand and to implement but also static and offers a limited expressiveness.
- **Content-based** scheme improves on topics by introducing a subscription scheme based on the actual content of the considered events. That can include meta-data associated with the event and not limited to topic name. Consumers subscribe to selective events by specifying filters using basic comparison and logic operators (=,<,>,≤,≥,and,or, etc...) to form complex subscription patterns. This scheme enforces a finer granularity than a *topic-based* one but can lead to higher runtime overhead.
- **Type-based** scheme changes the paradigm of topic names by introducing types. In other words, given that topics regroup events with similarities in content and structure the *type-based* scheme filters events according to those commonalities. In addition to this classification, a fine grained filtering can be performed using basic comparison and logic operators. This scheme is close to the *type-based* and pushes the limit of event filtering a little further. It has the same advantages and drawbacks of the *type-based* which has a higher runtime overhead compared to the *topic-based* scheme.

Neither the advertisement of nodes nor the synchronous and asynchronous communications are suited for MQTT. Furthermore, the Pub/Sub logic implemented on this protocol is too basic and tends to add complexity in a lot of use-case implementations. Our proposal brings content-based logic in the Pub/Sub design of CoAP.

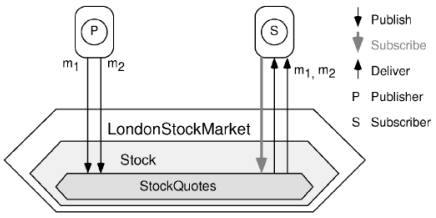


Figure 3.1 – Topic-based Pub/Sub interactions

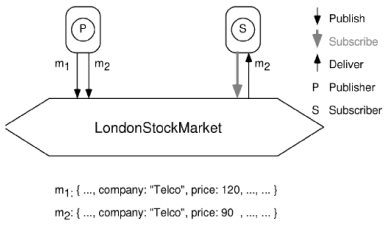


Figure 3.2 – Content-based Pub/Sub interactions

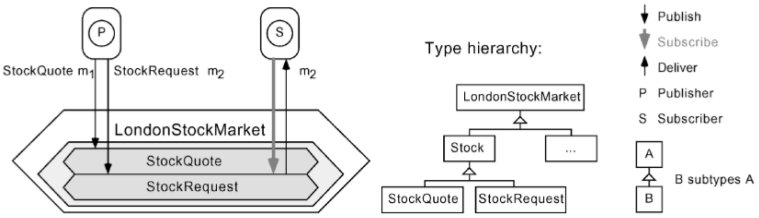


Figure 3.3 – Type-based Pub/Sub interactions

3.2.3 CoAP

CoAP is a service layer RESTful protocol built on top of UDP that is intended for use in resource-constrained Internet devices, generally in wireless 8-bit micro-controllers with small amounts of ROM and RAM. As defined in the RFC7252 [Shelby et al. 2014], it is designed for Machine-to-Machine applications and to be easily translated to HTTP for simplified integration with the web of Persons. It is also described as a less verbal version of HTTP. A typical CoAP message is depicted in Fig A.4. Each CoAP message occupies the data section of one UDP datagram. It consists of a fixed-size 4-byte header followed by a variable-length Token, a sequence of CoAP options and the payload.
In Fig. A.4, **Ver** is a 2-bit uint (unsigned integer) that mentions CoAP version number. **T** is a 2-bit uint with a value of 0, 1, 2 or 3 which stands respectively for Confirmable, Non-Confirmable, ACK or RESET. **TKL** is a 4-bit uint that indicates the length of the token(0 to 8 bytes). **Code** is a 8-bit uint that indicates Request method(1-10) or Response Code (40-255). Message ID is a 16-bit uint which is an identifier for matching responses.

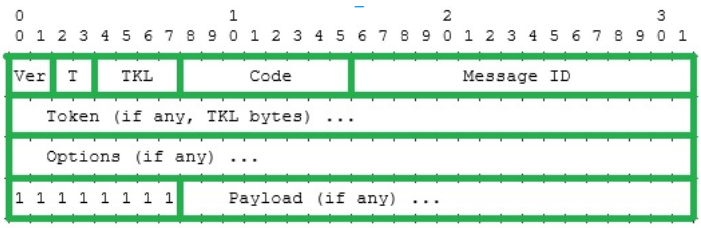


Figure 3.4 – CoAP Message Format.

The Constrained RESTful Environments (CoRE) Working Group has designed this protocol with the following features in mind:

- Overhead and parsing complexity.
- URI and content-type support.
- Support for the resources discovery.
- Simple subscription for a resource, and resulting push notifications.
- Simple caching based on max-age.

Even though CoAP was more intended to enlighten HTTP, there are tools allowing *Synchronous and asynchronous communications* and *Notification. Node Advertisement* was unfortunately out of the scope of CoAP RFC.

Synchronous and asynchronous communications

These communication mechanisms are well defined in CoAP and are known under the names of **Piggybacked** Response and **Separate** Response.

- A **Piggybacked Response** is a response carried directly in the message that acknowledges the request. That means that a response is expected right after request.
- A **Separate Responses** is composed of 2 CoAP messages: the first message is the acknowledgement response (ACK) to avoid the client to repeatedly retransmit the request message. The second message is the Confirmable message (CON) along with the response to the client request.

Fig 3.5 and 3.6 show examples of Requests with Piggybacked and Separate Responses.

Notification

In CoAP, the notification mechanism is implemented on the IETF (Internet Engineering Task Force) standard *Observing Resources in CoAP* [Hartke 2015], which enables CoAP clients to "observe" resources. In other words, CoAP clients can retrieve a representation of a resource and keep this representation updated by the server over a period

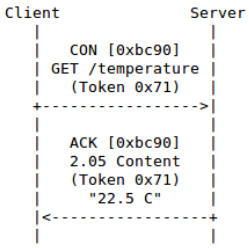


Figure 3.5 – Piggybacked Response

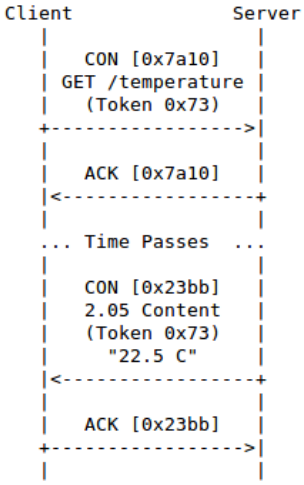


Figure 3.6 – Separate Response

of time. The Observe extension is normalized as depicted in Table 3.4. The value of the Observe Option is encoded as an unsigned integer for which possible values are **0** or **1**. If set to 0, the server adds an entry to the observers list, if not present. But if set to 1, the server removes an entry from the observers list. The table containing the observers is made of 3 rows: the Resource (for example /temperature), the Observer Endpoint (for example 0x4d45) and the Observer Token (for example 0x21).

When included in a response, the Observe option identifies the message as a notification and the server notifies the corresponding observer in the Observe list of the changes to the resource state. Fig 3.7 shows an example of notification with Observe.

Table 3.4 – Observe option.

Number	C	U	N	R	Name	Format	Length	Default
6		x	-		Observe	uint	0-3 B	(none)

Number=Option Number, C=Critical, U=Unsafe, N=No-Cache-Key, R=Repeatable

3.2.4 Related Works

The IETF has been working on projects aiming at bringing both the discovery and the Pub/Sub functionalities into CoAP [Palombini 2018]. While Discovery concerns are more about broadcasting node resources, proposed solutions for the Notification are built around Topic-based Pub/Sub scheme. Moreover, there is no single implementation

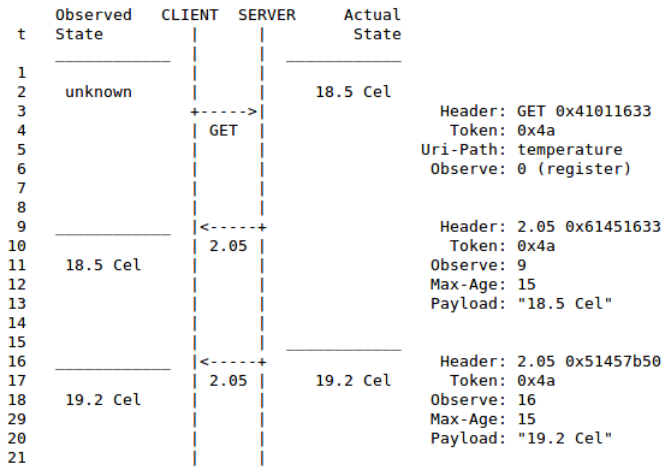


Figure 3.7 – A client registers and receives one notification of the current state and one of a new state upon a state change.

of these works as working groups are focusing their efforts on making cryptographic methods used in HTTP efficient enough to run on constrained nodes.
When referring to *Discovery*, CoAP addresses the *Resource Discovery* problem rather than the *Node Discovery* one. The CoRE working group proposed a solution which consists in exposing discoverable resources on the path *./well-known/core* of a given server identified on the network by a given IP address. There is therefore a need to embed zero-configuration mechanisms for 2 reasons: 1) Discovering the different nodes on the network and choosing them dynamically, depending on device access control policy; 2) mDNS is a good inspiration for our approach.
Where applications and resources are available on constrained devices, the Authentication and Authorization for Constrained Environments (ACE) working group recommends the adoption of the Open Authorization Framework (OAuth2.0) approach [Hardt 2012] to better manage access control. As numerous works following that recommendation [ACE 2017], our proposition is also inspired by OAuth2.0.

3.3 Our approach: CoAP 2.0

In this section, we describe the new protocol based on the three main IoT concerns: *Advertisement*, *Synchronous and Asynchronous Requesting* and *Notification*. As explained throughout section 3.2, designing a better CoAP will require having a Node Discovery and a more dynamic notification system.
Our approach, depicted in Fig. 3.8, redefines the Advertisement and Notification mechanisms without changing the Requesting architecture.

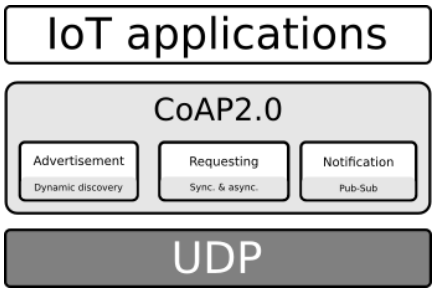


Figure 3.8 – CoAP 2.0 protocol stack

3.3.1 Discovery

We are proposing a Node and a Resource Discovery in our proposed protocol. The Node discovery is mostly based on the mDNS architecture. To be zero-configuration congruent, we use the same default configuration as mDNS. The multicast queries are then sent to 224.0.0.251 IPv4 and FF02::FB IPv6 addresses and on port 5353. Applying a dynamic access control policy and realizing use cases depicted in Fig.3.9 requires the querying node to be identified, not only with its network characteristics but also with credentials. We then added the field **QAUTHORIZATION** of a variable length to the mDNS query to make the **CoAP2 Discovery Query** depicted in Table A.1. The authorization in our proposition is expected to be delivered by an OAuth2.0 Authorization Server and will be verified before submitting any response. Browsing the entire network requires setting the QNAME field to *.local*. In that case, the querying client is expecting to receive multiple **CoAP2 Discovery Response** with RDATA field filled with the list of discoverable resources. However, when the QNAME field is set to a specific hostname, only one Response is expected.

Table 3.5 – CoAP 2 Discovery Query Fields.

Field	Description	Length/Bits
QNAME	The hostname to resolve or just ".local"	variable
QTYPE	Set to 'COAP'	16
U-R	Set to '1'	1
QCLASS	Set to 'IN' for Internet	15
QAUTHZ	The authorization token	Variable
FILTER	list of paths to resolve	Variable

Whatever the value of U-R, if QAUTHZ is not null then the response will be unicast for obvious security purposes.

The **CoAP2 Discovery Response** has the same exact structure as the mDNS Resource Record. The idea is to perform Node and Resource discovery with one request so that all needed information about a node is collected within a single request. The information is:

- IP address for the named entity
- hostname of the entity
- discoverable resources if any.

A type is needed to formalize the above information. According to DNS/mDNS RFC, two complementary types are needed to qualify this bundle of information. We are proposing in CoAP 2.0, to gather these information under the type **COAP**. It can be seen as a concatenation of 'A' and 'TXT' types defined in DNS IANA Considerations [Eastlake 3rd 2013]. It can also be seen as a 'TXT' type with *IP address*, *hostname* and *discoverable resources* as mandatory fields.

Listing 3.1 – Example of CoAP2 Discovery Response

```
1 Type:       COAP
2 Hostname:   coffee-machineAZ012.local
3 TTL:        5 minutes
4 Addr:       192.168.0.3
5 Resource:   /sensors/temp
6 Resource:   /sensors/light
7 Resource:   <etc>
8 Text:       additionnal information 1
9 Text:       <etc>
```

Another particularity of the CoAP Discovery Query is the *query filtering* thanks to the field **FILTER**. Inspired by the filtering from the CoRE RFC [Shelby 2012], it is designed to reduce the number of incoming responses when browsing the entire network. Only servers with discoverable resources similar to the values given on the QUERY field will respond.
With the new version of CoAP, the access-control policy in the use-case depicted in Figure 3.9 is easily configured. For example, Service 1 is private and reserved to a given set of users. These users are distinct from users using Service 2 by a token. This token was issued by an authorization server in a previous enrollment step. This step is out of the scope of this chapter but OAuth-like mechanisms are recommended for the enrollment because they are meant to guarantee a secure delegation of authorization.

Unlike the actual CoAP Discovery, our proposal combines the Node Multicast from mDNS, which retrieves a node among others thanks to the hostname, with the Node Resource Discovery.
Subsequently, we can browse the network on a single request using selective querying. In other words, in a network comprising of 10 nodes where only one node has the desired resource "light", the existing method queries all 10 nodes and receives 10 IP addresses. Then, for each address received, a Resource Discovery is performed to identify the node with the right resource. With CoAP 2.0, only the node of interest responds to the broadcast, rather than the totality of nodes. In fact, the querying node expects a response only from nodes concerned by the content of the filter. Moreover, access control is at heart because if the resource "light" is part of the Service 1 then only users with the right token will be able to find the node on the network, on a single query.

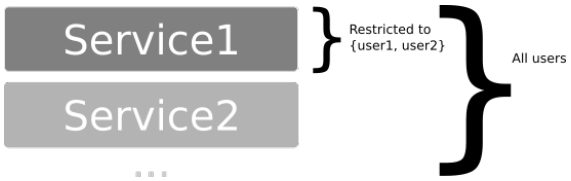


Figure 3.9 – Basic access control use case in IoT.

3.3.2 Notification

As described in paragraph 3.2.2, there are three concepts of Pub/Sub: *Topic-based*, *Content-based* and *Type-based* concept. They are listed from the simplest to the most elaborate concept. *Observe*, which is the notification system in the current version of CoAP, is based upon the simplest Pub/Sub concept but also the most static one. In CoAP2, we are proposing a smarter notification system by adding rules on the Subscription Request. Adding a Content-based concept to CoAP requires to modify: the *Subscription*, the *Observers Management* and the *Unsubscription*.

CoAP2 Subscription

All of our improvements are based on the *Observe* option. With CoAP, subscribing requires a GET method with an empty payload. In CoAP2, if the **Rule** field is set, the subscriber will only receive notifications abiding by the subscribing rule. Thanks to basic comparison and logic operators (=,<,>,≤,≥,and,or, etc...), rules can easily be defined by the client and simply be digested by the server. The Figure **??** shows an example of CoAP2 subscription.

CoAP2 Observers Management

It should be possible to subscribe to a path with different rules. The observer, the path and the rule are linked by a **subscription number**. This relationship is stored on a database. Another reason to use this number is to identify the observer, the path and the rule with less data on the request.

CoAP2 Unsubscription

The response to a successful Subscription Request is an integer corresponding to a **subscription number**. It is used to unsubscribe the observer from a path and a rule with less data in the request. DELETE is used in place of GET for the Unsubscription Request in order to delete the entry corresponding to the subscription number in the Subscription table. The Fig **??** shows an example of CoAP2 subscription and unsubscription.

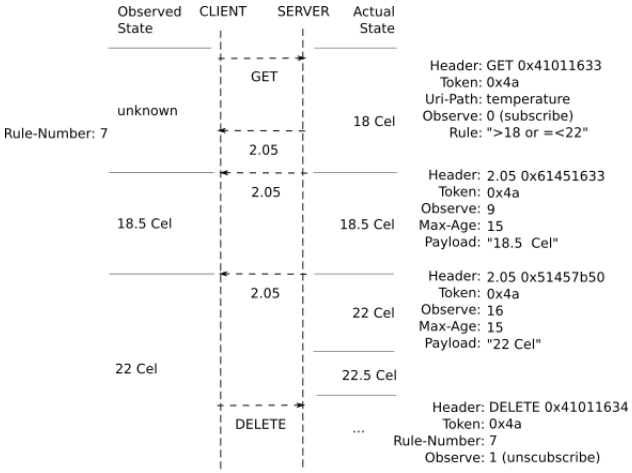


Figure 3.10 – A Client Registers and Receives Notifications of the Current State according to the *Rule*: $18 < State \leq 22$

As a result, performing data filtering before data communication plays an important role in curbing the risk of network congestion.

3.4 Implementation of CoAP 2.0

This section provides implementation details about each of the 3 main functionalities of CoAP 2.0. Regarding UDP messages, we are not changing the Requesting functionality proposed by the first version of CoAP. That said, only the Discovery/Advertisement and

the Notification are fully implemented using Python3. The implemented functionalities are evaluated in section 3.5.

3.4.1 Advertisement and Discovery

The two features are part of one property offered by the newly designed protocol as they work in pairs. The actions associated with these features are **Advertise** and **Discover**.

- **Advertise** is used to publish the domain name (referred to as QNAME in section 3.3) the device uses to advertise itself and give an appropriate response according to the defined advertisement policy. The Listing 3.1 gives an example of advertisement response. We define the *CoAP2.advertise.Response* object to add or modify some response attributes like TTL or Text. The response also contains a list of resources. By default a resource is "discoverable". Depending on the requester, they are removed from the list or not. More details on Listing 3.2

Listing 3.2 – Overview of CoAP 2.0 Advertise class

```
1  # Advertisement policy. 'req' is depicted in Table 5
2  def on_advertise(req):
3      response = coap2.advertise.Response
4      if req.authnz == "azert013":
5        reponse.text = "some additional information..."
6        response.ttl = 2000 # in milliseconds
7      else:
8        response.resource.remove = '/sensor/light' #the identified
             resource will not appear on the response
9      return response
10
11 coap2.advertise(name, on_advertise) # 'name' is the hostname to
       advertise
```

- **Discover** is used to retrieve the IP address and available resources of each node using authorization token and filters. This functionality makes more sense when the hostname is unknown. However, with CoAP2, to reduce the number of responders, a discovery request can contain a string to filter nodes according to their hostname. Only the node which hostname contains the string character has to repond. Listing 3.3 provides details.

Listing 3.3 – Overview of CoAP 2.0 Discovery class

```
from coap2 import Discover as coap2
# Query elements from Table 5
req = {'hostname':'device01', 'authz':'azert01', 'filter':['/temp']}
def on_discovery(rsp):
# Process advertisement responses here. 'rsp' is depicted in
    Listing 1.1
 pass

coap2.discover(req, on_discovery)
```

The client-server logic is a bit blurred because we consider each node can behave as a client and as a server at the same time. As a server, the node defines a resource and its access control policy on a function as depicted in Listing 3.4. We use Python decorators to ease resource declaration.

Listing 3.4 – Example of resource declaration

```
@coap2.resource('/sensor/temp', ['GET'])
def getTemp():
  temp = '12'
  return temp
```

3.4.2 Requesting and Notification

The Requesting and Notification functionalities are handled within the same Class *CoAP2.Request*. The major difference between synchronous, asynchronous and notification requests are the expected responses. The options Create Read Update and Delete (CRUD) are taking into account respectively with POST, GET, PUT, DELETE functions. Each of these functions accepts *uri*, *data*, a *timeout* and a *callback function* as parameters. The callback function is used to handle asynchronous requests. When set to Null, the request is considered synchronous.
Two functions are handling Notification: SUBSCRIBE and UNSUBSCRIBE. The first one accepts 3 parameters: *uri*, *rule* and a callback function *on_response* to handle incoming responses abiding by the rule and coming from the uri. The second function accepts *uri* or a *number*, which represents a list of subscription numbers received as a response when subscribing to a uri with a given rule. More details on Listing 3.5

Listing 3.5 – Overview of CoAP 2.0 Requesting class

```
from coap2 import Requests as coap2
'''
:type uri: string
:type data: string
:type on_response: function
'''
# Examples of synchrone request
req1 = coap2.get(uri="coap://192.168.0.3/sensor/temp")
req2 = coap2.post("coap://192.168.0.3/sensor/light", data={'val':13})
# Example of asynchrone request
values=(1,2,3,4,5,6,7,8,9)
def on_delete(resp):
 # Handle the resp asynchronously
    pass
req3 = coap2.delete("coap://192.168.0.3/sensor/light", data=values,
    on_response=on_delete)
# Examples of subscription requests
rule1=">18 or =<22"
rule2=">=100"
def subs_response(resp):
 # Handle received data
    pass
req4 = coap2.subscribe(uri="coap://192.168.0.3/sensor/temp",
    rule=rule1, on_response=subs_response)
req5 = coap2.subscribe(uri="coap://192.168.0.3/sensor/temp",
    rule=rule2, on_response=subs_response)
number = req4.subscribe_number # Get the subscribe number
# Examples of unsubscription requests
req6 = coap2.unsubscribe("coap://192.168.0.3/sensor/temp")
req7 = coap2.unsubscribe(number)
```

3.5 Evaluation and Security analysis

In this section, we evaluate the performance of the newly designed CoAP2 in terms of Discovery and Notification. We also discuss security concerns.
The hardware used for each experiment is a $9 computer called *CHIP* [Things 2016] with an ARM Cortex-R8 processor (@1 GHz) as a CPU, 4GB of ROM, 512MB of RAM, 802.11b/g/n WIFI built-in and Bluetooth 4.0 LE (Low Energy). The hardware

is running a lightweight linux distribution as an OS. The company was launched in 2016 with a huge kickstarter campaign but officially closed in the spring of 2018. The project is however still surviving thanks to an active community [Community 2018].

3.5.1 Performance evaluation

We evaluated the Flash memory consumption and the Response Time for the Discovery and Notification. We have two experimental sets: one for the Discovery evaluation and the other for the Notification evaluation. Both sets are composed of three CHIPs as nodes. For the Notification tests with MQTT, the broker used is provided by the Eclipse Foundation.

Discovery Evaluation

The experiment set contains three nodes on the same local network.

- Firstly, we evaluate the impact of the advertisement on the node. We run the COAP2 discovery daemon for half an hour on the node. Every 2 seconds another node broadcasts a discovery requests to which the evaluated node is responding. As a result, advertisement averages 2.05MB in memory consumption which represents 0.4% of available resources on this node.
- Secondly, we evaluate the impact of the filtering on the broadcast. As depicted on Figure 3.11, the more there are nodes responding, the more there are packet collisions. We can see that it takes 4 seconds to discover the node3 which is however located in the same place as all the other nodes. Figure 3.12 shows how efficient are *hostname* and *resource* filters in terms of memory consumption. In fact, in this set of connected things, there is only one node with the hostname *coffeemachine.local* and two nodes with the resource */sensor/*. The experiment shows that whatever the number of nodes on the network, the hostname filtering requires the same amount of memory, which on the CHIP represents 6.9% of RAM. Filters are playing an important role in reduction of CPU consumption and making the discovery faster.

Notification Evaluation

Figures 3.11 and 3.12 are depicting the Notification experiment. In fact, depending on the rule, the number of notifications is dramatically reduced, irregardless of the

evolution of temperature. As a result, rules have a direct impact on the reduction of CPU consumption. In comparison, the systematic use of a broker makes MQTT the slowest protocol among the Internet connected-nodes protocols.
Based on the results of these experiments, we managed to gather a smart discovery and notification with a reduced impact on memory.

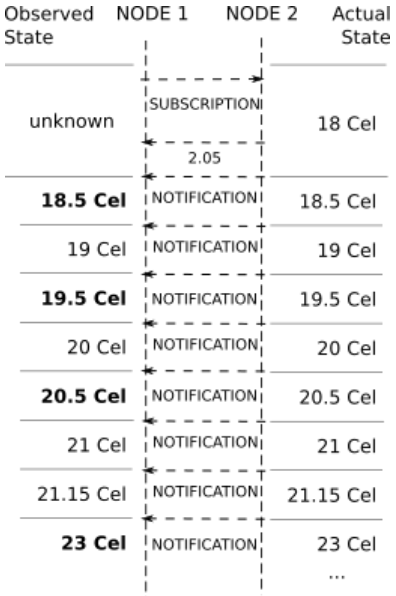


Figure 3.11 – CoAP subscription experiment

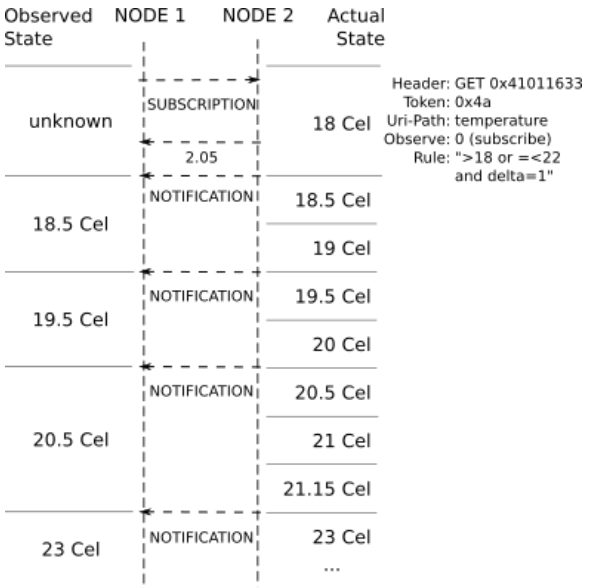


Figure 3.12 – CoAP2 subscription experiment

3.5.2 Security analysis

Our protocol is fulfilling one of the security requirements in IoT systems, dependability. In fact, the protocol is guaranteeing thanks to filters and the use of authorization field in the broadcast, that only authorized clients are able to discover and access protected resources.
For obvious reasons, there is a need to encrypt messages sent over the network. Like HTTP with TLS, CoAP uses a version of TLS on top of UDP called Datagram TLS (DTLS) [Raza et al. 2013]. This feature is still available in CoAP 2.0. As many works of the Internet Engineering Task Force (IETF) are recommending, CoAP 2.0 messages can abide by CBOR Web Token (CWT) [Jones et al. 2018], an adapted version of JSON Web Token (JWT), or CBOR Object Signing and Encryption (COSE) [Schaad 2017]. These formats are guaranteeing the integrity of exchanged messages. They are equivalent to certificates in PKI infrastructures.

However, with the growing number of connected things with an IP address, it is more and more painful and costly to maintain a PKI infrastructure to insure encrypted communications in IoT. As a future work, we suggest to exploit two abilities specific to connected things as an alternative to PKI architecture: the proximity of nodes and the ability to create access points. These specificities can be used to build a private channel dedicated to forge a shared key.

3.6 Conclusion

There is a need to have a protocol specifically designed for IoT and with security at heart. Before our proposition to upgrade CoAP, building a connected device would have required to gather at least 3 different technologies: one to perform the discovery, another one to perform the publish-subscribe and finally one for asynchronous communications. It would have required a study to determine which one of the publish-subscribe or the asynchronous communications would be more adapted to the use-case. Given that the role of the connected device will have to change during its life cycle, choosing the right protocol can be a real issue.
With this proposed version of CoAP, whatever the use-case involving Internet-connected things, the protocol will always be the most efficient and most easy to maintain because the discovery, the synchronous and asynchronous communications and the publish-subscribe philosophy are in the foundations.
However, when introducing resource filtering we are only referring to simple types. Our

implementation only considers int and float. We propose as a future work to develop a syntax to filter complex types inspired by the SQL syntax. Another work in perspective is to bring CoAP 2.0 into browsers by developing an extension.

Part II

Security Mechanisms for Constrained Devices

CHAPTER 4

A certificate-less key exchange protocol for IoT

4.1 Introduction

Diffie-Hellman (DH) [Diffie and Hellman 1976] key exchange aims at allowing two parties that have no prior knowledge of each other to jointly establish a shared secret key over an insecure channel. Its security is based on the presumed difficulty of solving the discrete logarithm problem (DLP) [Smart 1999] [Nyberg and Rueppel 1994]. To keep this promise, it is strongly recommended to use keys greater than 2048-bits [Velvindron and Baushke 2017] to avoid man-in-the-middle attacks. In the context of Internet of Things, increasing the size of the key is not always a viable solution given the reduced amount of memory, computation power and energy of devices. In fact, when referring to IoT, we are stressing more on class 0 and 1 devices [Bormann et al. 2014]. That describes fewer computation power devices, meaning less than 10KB of RAM and 100kB of Flash. These devices are meant for basic communication to transmit sensor data to servers, most of the time using a gateway. To avoid using longer keys, many are implementing 512-bits Elliptic Curve Cryptography (ECC) which offers the same level of security for smaller key size [Gura et al. 2004].
Public Key Infrastructure (PKI) is vastly used to establish a link between user's identity and its public key. Knowing that issuing and using certificates are often costly for the IoP, no engineer could imagine using a PKI in the context of IoT where billions of connected things are expected to join the Internet, this architecture is not maintainable and will be more and more costly.
Based on the two identified challenges namely: the need for more space and computation power for the Diffie-Hellman problem to remain difficult to solve and the cost of a PKI architecture in the context of IoT, the problem we address in this chapter is: **How to**

secure IoT communications with Diffie-Hellman algorithm in a certificate-less architecture ? In this chapter we propose an original method, that allows two connected things to mutually create an encryption shared key, with the Diffie-Helman algorithm on a private channel. The proposed solution lightens encrypted communications in IoT as it makes certificates useless. The performance analysis of this solution shows promising results.
The rest of this chapter is organized as following: Section 4.2 discusses the Diffie-Hellman challenge in IoT context and why a PKI architecture is not maintainable. It is also dealing with OpenID Connect which is part of our solution. Section 4.3 presents our solution. We give a security and performance evaluation of our approach in section 4.4 . We present related work about certificate-less protocols in section 4.5. Finally, section 4.6 concludes with an outline of future work.

4.2 Background

4.2.1 The Diffie-Hellman Problem (DHP)

Encrypting communications relies on functions that are fast to compute but hard to reverse called one-way functions. It requires the use of mathematical problems in cryptographic protocols. The function must be hard to reverse for attackers. The Diffie-Hellman problem is a mathematical problem for the attacker which consists in finding the value of x and y given g^x and g^y. Where g is the generator of some group (elliptic curve or multiplicative group for example) [Miller 1985] [Gupta and Murty 1986] [Koblitz 2012] and assumed to be public. x and y are randomly chosen integers kept secret. In the most considered of the many variants to the DH problem, the Decisional Diffie-Helman Problem (DDHP) [Boneh 1998], Alice and Bob choose x and y and compute g^x and g^y, respectively. Those values are sent to each other so that each party can compute the shared key $g^{xy} = g^{yx}$. Table 4.1 depicts who knows what during the protocol exchange.
The problem the attacker must solve is finding x giving g^x or y giving g^y. But it is not easy to compute discrete logs. Otherwise, it would be easier to find g^{xy} after having computed $log_g g^x = \frac{lng^x}{lng}$ for the value of x and $log_g g^y = \frac{lng^y}{lng}$ for the value of y.
The security of many DHP cryptosystems is then based on what is called the Computational Diffie-Helman (CDH) assumption. But with computation progresses, we can observe man-in-the-middle attacks on DH when keys sizes are ranging from 512-bits to 1024-bits. A famous one is the **Logjam**.

Table 4.1 – Decisional Diffie-Helman protocol execution: Actors knowledge. Eve is an attacker. The protocol can be easily expandable with other actors using bilinear applications [Delsarte 1978]. For instance, if Carol is the third actor, the shared key would be g^{xyz}. With z randomly chosen by Carol.

Alice	Bob	Eve
g	g	g
g, x	g, y	g
g, x, g^y	g, y, g^x	g, g^x, g^y
$(g^y)^x$	$(g^x)^y$	?

Logjam attack [Adrian et al. 2015].

It is the most popular form of man-in-the-middle attack on DH. It is a vulnerability against DH from 512-bits to 1024-bit keys. It consists in first downgrading vulnerable TLS connections to 512-bit export-grade cryptography, then computing discrete logs algorithm on that 512-bit group. IETF then made the recommendation to use keys greater than 2048-bits to guarantee security in protocols using DH. The logjam exploits the method based on the number field sieve algorithm [Lenstra et al. 1993] to find discrete logarithms.
The attack was performed by David Adrian et al. on TLS and is fully documented on https://weakdh.org/. TLS consists in three phases:

1. **Hello messages**: The client sends a random nonce *cr* and a list of supported cipher suites with the **ClientHello** message. The server selects the appropriate cipher suite from the list and responds with its nonce *sr* with the **ServerHello** message.
   An example of cipher suite: *ECDHE-RSA-AES128-GCM-SHA256.*

2. **Key exchange messages**: The server chooses a group *(p,g)*, where p is the prime and g the base. He computes g^b and sends the tuple *(cr,sr,p,g,*g^b*)* signed using the key sk_S from its certificate $cert_S$. This message is called the **ServerKeyExchange** message. The client responds with g^a in the **ClientKeyExchange** message.

3. **Finished messages** to ensure agreement on the negotiation. Both party computes the secret key g^{ab} and calculates a MAC which exchange in a pair on messages.

The Logjam attack in TLS depicted in figure 4.1 mostly relies on the fact that DHE_EXPORT cipher suites is identical to DHE but were restricted to primes no

longer than 512 bits. The discrete log algorithm performed on the most used 512-bit prime produced a database. With a sufficient precomputation, the attacker quickly finds $b = dlog(g^b mod p_{512})$
Establishing a cipher key requires each party to be authenticated. The IETF adopted certificates and Public Key Infrastructures to solve this problem.

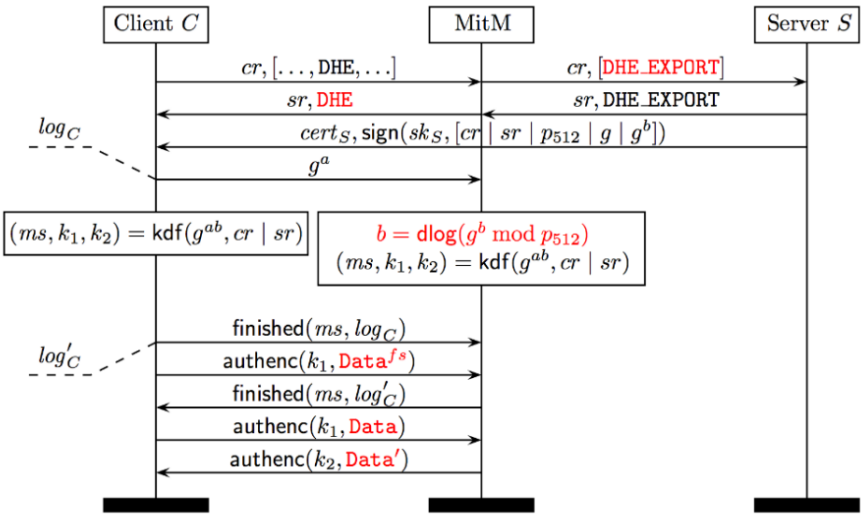


Figure 4.1 – The Logjam attack on TLS [Adrian et al. 2015]

4.2.2 Public Key Infrastructure (PKI)

A Public Key Infrastructure guarantees the confidentiality of data, the non-repudiability and the strong authentication of a numeric identity through certification authorities (CA). These organisms are supposed to be trustworthy and publish their public keys. In this architecture, a numerical identity is represented as a certificate [Housley et al. 1998]. It is a proof of identity made by the certification authority public key and the service identity, all together signed with the certification authority private key. When a user is requesting a service resource, the request is sent with the user certificate. If the certificate is authenticated, the service can then use the public key of the corresponding certificate authority to verify the integrity of received data.
There are no problem in theory with a PKI but vulnerabilities come from CAs and key management. These vulnerabilities are possible because of the lack of control of CAs, which is not always an easy task given that a certificate can certify other certificates: this is called the chain of trust, depicted in figure 4.2. An attacker can certify its public key or certificate authorities private keys can be stolen. For instance, the Stuxnet attack

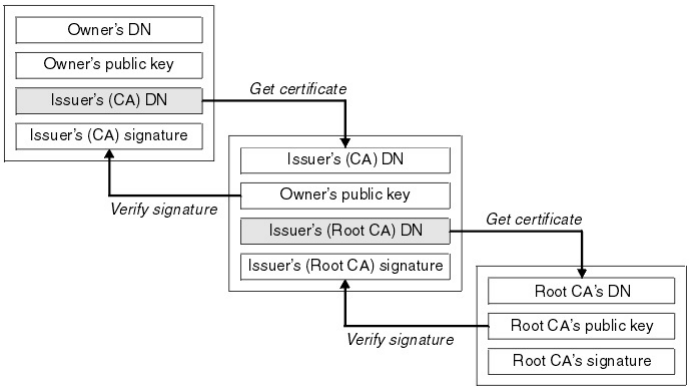


Figure 4.2 – A certification path from the certificate owner to the Root CA

against the Iranian nuclear program was performed thanks to certificates stolen from integrated circuit manufacturers, Realtek Semiconductors and JMicron [Langner 2011] [Kelley 2013].

In the context of IoT, the number of service providers is increasing along with the need to certify them. In fact, there are use cases where connected devices can be considered as service providers.

However, a certificate based architecture has some flaws which can prove to be huge source of attack. Bruce Schneier and Carl Ellison listed ten risks of PKI in [Ellison and Schneier 2000] and two of them are particularly dangerous for IoT if not corrected. The flaws are the following:

- Untrusted authorities can be part of the chain of trust. It is then not obvious to know who is using the key. The secrecy property cannot always be guaranteed either. Moreover each time there is a new untrusted party in the chain brings more complexity and latency into the authentication process.
- Two different actors can have different keys but identical names. And Certificate verification does not use a secret key, only public keys. The risk is that an attacker public key is added to the list of verifying public keys. The property of authentication is not always guaranteed.

For the reasons listed above, a PKI is not adapted to IoT. A more distributed and lightweight architecture is more appropriate. We chose another solution to couple authentication with DH.

4.2.3 OpenID Connect

OpenID Connect 1.0 (OIDC) [Sakimura et al. 2014] is a simple identity layer on top of the OAuth 2.0 depicted in section 1.3.1. It allows Clients to verify the identity of the End-User based on the authentication performed by an Authorization Server, as well as to obtain basic profile information about the End-User in an interoperable and REST-like manner. It is implemented by the most used identity providers (Facebook, Twitter, Google). OIDC uses the same architecture defined by OAuth:

- A **Resource Owner (R.O).**
- A **Client (C)** as the application requesting information about the resource owner.
- An **Authorization Server (AS)** which delivers authorization grants representing the resource owner's authorization.
- A **Resource Server (RS)** hosting owner's resources.

We want in our solution to establish a shared key between two authenticated connected devices. Depending on the properties expected, OIDC has three authentication flows: *authorization code flow*, *implicit flow*, *hybrid flow*. The table 4.2 depicts OIDC flows according to properties.

Table 4.2 – OIDC flows according to properties. A.E stands for Authorization Endpoint. T.E is Token Endpoint. U.A is User Agent

Property	Authorization Code Flow	Implicit flow	Hybrid flow
Tokens returned from A.E	no	yes	no
Tokens returned from T.E	yes	no	no
Tokens not revealed to U.A	yes	no	no
Client can be authenticated	yes	no	yes
Refresh Token possible	yes	no	yes
One round trip comm.	no	yes	no
server-to-server comm.	yes	no	varies

Because they are supposed to run the authentication phase without human intervention, we consider a connected device as servers in OIDC paradigm. We then chose the **hybrid flow** to verify connected things identity and to prove their trustworthiness.

Our use of the hybrid flow follows the following steps:

- Alice (considered as a C) sends the request to the A.S. Alice needs a *client_id* to make that request. Only a registered C can successfully make that request.
- A.S authenticates Bob (considered as a $R.O$). Bob can only authenticate with a *client_id* and a *client_secret*
- Alice obtains an authorization code, proof of Bob authentication.

Among all the authorization flows, the hybrid flow is the lighter one. There are less data transmitted and there is no need for the device to handle tokens with this flow.

4.3 Our solution: Ad Hoc private channel

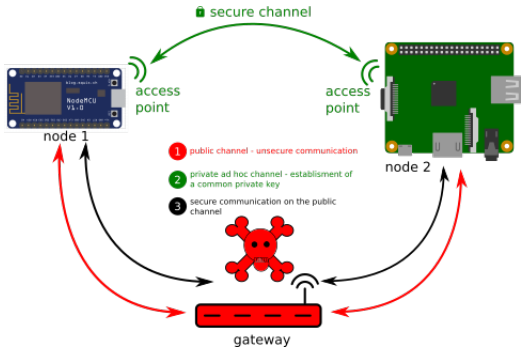


Figure 4.3 – Solution overview

The concept of our solution is to use the ability of the connected things to build an ad hoc access point and use it as a private channel. The private channel is a Wi-Fi access point protected with a Wi-Fi Protected Access (WPA) passphrase and configured to receive only one device connected at a time. The ssid and passphrase should be at least 128-bit long so that brute force attacks are long enough for the protocol to be entirely executed hidden from the attacker.
This new and ephemeral channel is used to run another DH so that Eve, the attacker, can not be aware neither of the new base nor any other element computed by Alice and Bob, preventing a man-in-the-middle attack. The Figure A.6 gives an overview of the architecture of the solution. There are four phases in our solution:

The Authentication. It is the first step of the protocol and it performed on the public channel. It consists in making sure that Alice is not initiating a key exchange dialog with Eve, the attacker, but with Bob. Alice, who initiates the dialog, must send a list of the Authentication Servers (AS) where Bob's identity can be verified. Bob

responds with one element of the given list. Then follows the OIDC hybrid flow as described in section 4.2.3. We consider Alice and Bob honest, that means that each of them is supposed to have its OIDC credentials: (*client_id* and *client_secret*). After Bob was authenticated with his credentials, Alice receives an authorization code from the AS, which proves Bob's identity. This authorization must be a nonce. Furthermore, Bob is sure that Alice is trustworthy because he received a request for authentication from the right AS, right after having communicated its AS name to Alice.

The First run. It consists in the running of the DH for the first time, on the public channel, after each actor has authenticated themselves. The first message of this phase, from Alice to Bob, contains the chosen generator g in clear, since it is supposed to be public. At the end of this run, Alice and Bob will have computed the shared key g^{xy}.

Before initiating the second run, Alice will communicate to Bob the access point to connect to and a secret encrypted with the freshly computed shared key. Alice and Bob should be at a Wi-Fi range from each other. The range should be inferior to 46 meters for the traditional 2.4GHz band and up to 92 meters for the 5GHz band.

The Second run. It consists in switching the generator from g to a new one h. The final shared key is then h^{xy}. This step is always performed on the private channel.

The key_id negotiation. Each actor is in charge of the secure storage of the shared key. After the establishment of the shared key, Alice and Bob must set a common *key_id* so that neither Alice nor Bob have in their respective database, two keys with the same id. A simple way of achieving this task is that Alice proposes first a number and Bob responds with a different number if the one proposed by Alice corresponds to an id in the database. Alice must be able to make an *echo* with a number to end the negotiation. We propose the formalism **key_id:key**

Once the shared key is established, there is no need for a certificate when initiating the communication. In fact, the communicants only need to know the *key_id* corresponding to the key used to cypher the communication.

4.4 Security and performance analysis

4.4.1 Security analysis

The protocol guarantees a strong authentication of each participants and the secrecy of communications.

Authentication.

It is necessary to authenticate each party to avoid man-in-the-middle attacks. With the use of OIDC we can identify each actor taking part in the protocol: the client and the resource server. Unlike a PKI, with OIDC, Carl Ellison and Bruce Schneier questions like *who is using my key ?* and *who John Robinson is he ?* are not unanswered [Ellison and Schneier 2000]. The authentication property is always satisfied. Each actor is uniquely identified by the AS through the tuple (client_id, client_secret) and each key_id is unique. This property therefore holds if and if the temper resistance of the tuple. There is no ambiguity compared to a certificate based architecture. Furthermore, every time an authorization code is released, the AS is able to list the client and the resource owner bounded. This has a better impact on the database than a certificate and their life cycle. In fact, there is no need for the AS to keep track of revoked certificates as each connected device has a ledger of valid keys. Revoking a device is reduced to erasing the couple (key_id, key) from the ledger.

Secrecy.

With the proposed solution, there is always at most 3 entities and the encryption is symmetric. So there is no room for an untrusted party. The life expectancy of the ad hoc access point is long enough to build the shared key (less than 30 seconds) but too short for an intruder to succeed a brute force attack before the end of the protocol.

Attacks on the private channel.

We use WiFi Protected Setup with PIN protection mode to implement the private channel. There are two famous implementations of attacks on WPA: **Reaver-wps** [Viehböck 2011] [Murphy 2013] and **Pyrit** [Lueg 2013b] [Lueg 2013a]. Both attacks are following the methodology depicted in Figure 4.4. These attacks were perfomed on WPS-Certified routers with 128-bit encryption keys. It takes a minimum of 4 hours to

recover the correct Wi-Fi Protected Access (WPA/WPA2) passphrase.
Establishing an ad hoc access point (AP) and running the protocol takes less than **10 seconds**. At the end of the protocol, the attacker has no element he can use to guess the key and the AP is no longer available. Since the AP's life expectancy is shorter than 10 seconds, it is impossible, in our context, to eavesdrop an AP passphrase and then get the shared key. Our protocol is then invulnerable to brute force attacks.

Attack on DH.

As explained in section 4.2.1, the DH key exchange depends on the presumed difficulty of solving the DHP and on the CDH assumption. The Logjam attack consists of a *precomputation stage* that depends only on the prime p, and a *descent stage* that computes individual logs. The first stage can take days to compute when the second stage only takes seconds. With sufficient precomputation, an attacker can find within seconds the discrete log x from $y = g^x modp$ and break any Diffie-Hellman instances that use a particular p. If the logjam attack takes about a week to break the key (less than 1024-bit keys), in our context, this problem is unsolvable.
Let g be the base in a cyclic group, x and y randomly computed, respectively by Alice and Bob. x and y are the private keys. If Eve, the attacker, is unaware of g, g^y and g^x, computing a sufficient log database can take months or years. The problem now consists in solving the following system of equations:

$$\begin{cases} y = \frac{\ln h^y}{\ln h} & \text{with } h^y \text{ and } h^x \text{ unknown to the attacker} \\ x = \frac{\ln h^x}{\ln h} & \text{and } h \text{ the base in a cyclic group, also unknown to the attacker} \end{cases}$$

The proposed protocol is raising the level of difficulty of the DHP. It then remains difficult and we can therefore still use a DH modulus size less than 2048-bits and be Logjam-proof.

Another advantage of this architecture is that key management becomes easier. In fact, it is easy to build new keys and revoke old ones.

4.4.2 Performance analysis

We focus on the impact on the memory for this analysis. An objective analysis in term of execution time would require to consider a large variety of processors. This experiment is a work in progress and will be presented in future papers.
Let us consider a node A with 100 keys in the keys database, corresponding to 100

different nodes sharing secrets with A. The table 4.3 gives an evolution of the database size in terms of keys bit size. The node A is one of the ESP8266 series, the *ESP-12*. It is a system-on-chip (SoC) produced by Espressif Systems [Espressif 2014] and used on lot of IoT projects. It integrates a 32-bit processor (@80 MHz) as a CPU, 64KB of ROM, 80KB of RAM and 802.11b/g/n WIFI built-in. The hardware is running Micropython [George 2014], a lightweight implementation of Python as an OS.

Table 4.3 – Performance evaluation of key storage. The database size corresponds to 100 keys.

Key size (-bit)	DB size (Ko)
256	3.7
512	6.9
1024	13.3
2048	26.1

Storing and managing encryption keys with our solution have a very low impact on the memory. In fact, in the same amount of memory, with our solution we can store about 7 2048-bit keys where only one certificate with 2048-bit key (about 1.9Ko) can be stored.

4.5 Related works

There are many Diffie-Hellman based authenticated key agreement protocols. One of the challenges of the DH protected architecture is to authenticate the parties involved. We can therefore cite solutions like the well-known MTI protocol by Matsumoto, Takashima and Imai [Matsumoto et al. 1986] [Wang et al. 2008] which provides implicit key authentication in the classical Diffie-Hellman key agreement protocol. Flaws were detected on MTI and led to the creation of the MQV protocol [Krawczyk 2005] and a whole family of solutions based on MQV [Blake-Wilson and Menezes 1998]. Both MTI and MQV solutions are certificate-based, thus not adapted for IoT for the reasons listed in section 4.2.2. Another family of solutions are ID-based key agreement protocols which consist in associating a Personal Identification Number (PIN) to token [Scott 2002] [Smart 2002] [Shim 2003] [Chen et al. 2007]. The ID escrow is actually the new certificate which, compared to our solution, requires a heavier infrastructure. Other certificate-less protocols like Identity Based Cryptography (IBC) [Joye and Neven 2009] [Nicanfar et al. 2014] have been proposed and has some analogy with our proposal. In an IBC system, user's ID is considered as her/his public key and

the user's private key is generated by a trust authority, called Key Generation Center (KGC) or Private Key Generation (PKG). The disadvantage of that solution is that the trust authority knows the private key and with the example of the attack against the Iranian nuclear program [Kelley 2013], the IBC solution proves to be more dangerous.

4.6 Conclusion

Our solution shows promising results. In fact, we have demonstrated that it is easier to maintain and cheaper to build a shared encryption key using an ad hoc private channel. Moreover, scalability is one the key characteristics and challenges our solution is fulfilling. As recommended by [Barki et al. 2016], when it comes to IoT, having a fully centralized key management solution which is also guaranteeing scalability and security is hardly conceivable.
With our solution, the DHP remains difficult and neither the brute force attack reawerwps nor the logjam attack are effective to eavesdrop the key. A performance evaluation of the memory consumption reveals that our solution is 7 times more efficient that certificate-based architecture. In other words, where we can store only one certificate, there is room for 7 keys in our paradigm. Another advantage of this method is that the shared keys can be refreshed without a third-party involved. Certificates are replaced by key_ids and there is no need to keep track of revoked certificates.
There is however, one limitation with our solution which is the range. In fact, there cannot be a private channel if the nodes are out range from of each other. The range should be within 46 meters for the traditional 2.4GHz band and up to 92 meters for the 5GHz band. The specificity of this method can also be seen as its limitation depending on the use case.
This solution can be very useful to easily build and maintain a trusted ecosystem of connected things. We suggest as a future work, on one hand to implement the four steps of this protocol (authentication, run 1, run 2 and key negotiation) and compare the performances with DTLS in the Constrained Application Protocol (CoAP) [Shelby et al. 2014] [Raza et al. 2012b], in terms of packets transmitted and computation time. To optimize security and energy consumption, we are also considering working with the ESP Header Compression (EHC) framework [Migault et al. 2017] [Migault et al. 2016], to benefit from a 3% energy overhead instead of 95% with a standard Encapsulating Security Payload (ESP). On a second hand, we suggest to measure the impact of this solution in a smart home or smart grid architecture.

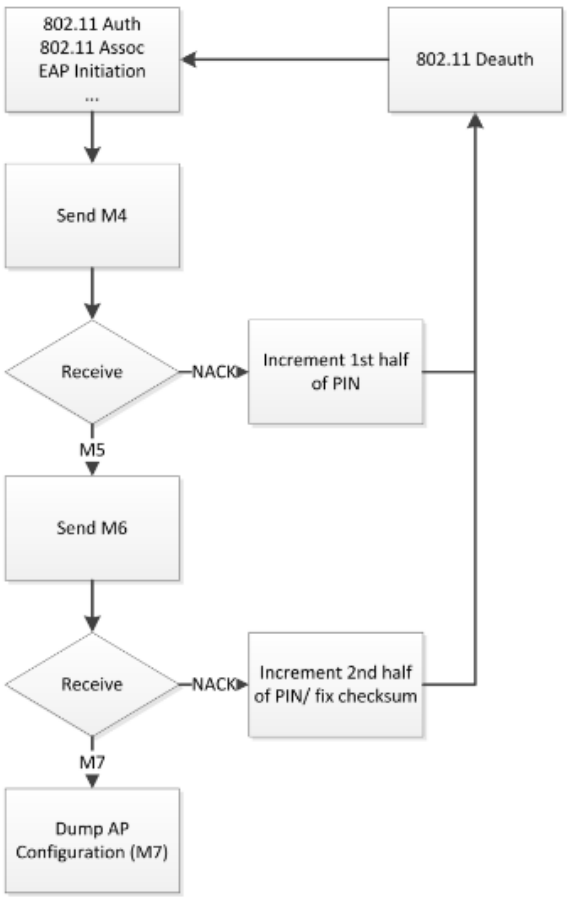


Figure 4.4 – Brute force methodology. M4,.., M7 are messages exchanged during the WPA authentication with PIN.

CHAPTER 5

Detection and response to Data Exfiltration from Internet of Things Android Devices

5.1 Introduction

The Internet of things (IoT) is the network of physical devices, sensors, actuators and smart devices that are connected through the Internet to exchange data [Gubbi et al. 2013b]. The number and variety of devices that are used to collect data have increased at an accelerated rate in recent years. Experts estimate that the IoT will consist of about 30 billion objects by 2020 [Nordrum 2016]. It is also estimated that the global market value of IoT will reach 7.1 trillion by 2020 [Hsu and Lin 2016]. With the increasing number of IoT devices, the user privacy threat is growing. Hackers aim to exfiltrate personal data stored in the IoT devices such as smartphones through USB port. [Do et al. 2015] propose an adversary model for Android covert data exfiltration, and demonstrate how it can be used to construct a mobile data exfiltration technique to covertly exfiltrate data from Android devices. [D'Orazio et al. 2017] investigate how an attacker could exfiltrate data from a paired iOS device by abusing a library and a command line tool distributed with iTunes. Existing security tools [Enck et al. 2014], [Fuchs et al. 2009], [Graa et al. 2016], [Graa et al. 2014], [Arzt et al. 2014] in Android systems focus on detection of the sensitive data leakage. While such tools may be effective in protecting against third party applications attacks, they are less suitable when the data exfiltration is performed by application requesting connection to the Android IoT device installed in the personal computer. In this chapter, we propose an effective approach that allows detection and response to attacks exploiting trusted relationship between a third party system such as a personal computer and an Android IoT device

to exfiltrate user data from the victim device to an attacker. We implement a protocol that ensures security of IoT Android device and personal computer communication. The rest of this chapter is organized as follows: Section 5.2 discusses the existing protocols that allow files transfer from Android IoT devices to the computer. Section 5.3 describes the threat model. Section 5.4 presents the proposed approach. Section 5.5 provides implementation details. We give a security and performance evaluation of our approach in section 5.6. We present related work about data exfiltration attacks and countermeasures in section 5.7. Finally, section 5.8 concludes with an outline of future work.

5.2 Background

Older Android devices support USB mass storage for transferring files back and forth with a computer. Modern Android devices use the media transfer protocol (MTP) or picture transfer protocol (PTP). The Open Authorization framework (OAuth) depicted in section 1.3.1 is also an inspiration to our solution, the delegation of authorization. The proof of this authorization relies on a token defined as JSON Web Token [Jones et al. 2015b] standard.

5.2.1 USB Mass Storage

The USB mass storage device class is a set of computing communications protocols that makes a USB device accessible to a host computing device. In addition, it enables file transfers between the host and the USB device. The USB device acts for a host as an external hard drive; the protocol set interfaces with a number of storage devices. The USB mass storage was the way older versions of Android exposed their storage to a computer. Using USB mass storage, users and applications running in the computer will be able to access to all files (system files, media and picture files...) in the Android devices.

5.2.2 Picture transfer protocol

Picture Transfer Protocol (PTP) [Bigioi et al. 2002] is a protocol developed by the International Imaging Industry Association. It allows the manipulation and the transfer of photographic images from Android devices to computers without the need of additional device drivers. When Android uses this protocol, it appears to the computer

as a digital camera. The protocol has a strong standard basis, in ISO 15740. It is standardized for USB as the still image capture device class.

5.2.3 Media transfer protocol

The Media Transfer Protocol (MTP) [Lamparter et al. 1992] is an extension to the PTP. Whereas PTP was designed for transferring photos from digital cameras, Media Transfer Protocol allows the transfer of music files on digital audio players and media files on portable media players from devices. MTP is standardised as a full-fledged Universal Serial Bus (USB) device class in May 2008. A main reason for using MTP rather than the USB mass-storage device class (MSC) is that the latter is designed to give a host computer undifferentiated access to bulk mass storage, rather than to a file system, which might be safely shared with the target device. Therefore, a USB host computer has full control over the connected storage device. When the computer mounts an MSC partition, it may corrupt the file system and makes it unsupported by the USB device. MTP and PTP specifically make the unit of managed storage a local file rather than an entire unit of mass storage at the block level to overcome this issue.

5.2.4 JSON Web Token

JSON Web Token (JWT) is an open standard [Jones et al. 2015b] that defines a compact and self-contained way for securely transmitting information between parties as a JSON object. This token can be verified and trusted because it is whether digitally signed or encrypted. When signed, it is referred to as JSON Web Signature (JWS) [Jones et al. 2015a], and a JSON Web Encryption (JWE) [Jones and Hildebrand 2015] is an encrypted JWT. JWTs can be signed or encrypted using a JSON Web Algorithms (JWA) [Jones 2015]. Table 5.1 depicts the list of algorithms defined in the specs.
JWT is compact and self-contained allowing to transmit rich information on a lightweight payload. It consists of three parts separated by dots, which are:

- **Header** that gives the type of the information and the algorithm being used.
- **Payload** which contains statements about an entity called claims. There are *Registered*, *Public* and *Private* claims.
- **Signature** is the result of the encoded header and the encoded payload signed using a secret and the algorithm specified in the header.

Table 5.1 – The set of algorithms "alg" defined in JWA specification

"alg" value	Signature method	Signing Key
HS256	HMAC using SHA-256	Required
HS384	HMAC using SHA-384	Optional
HS512	HMAC using SHA-512	Optional
RS256	RSASSA-PKCS1-v1_5 using SHA-256	Optional
RS384	RSASSA-PKCS1-v1_5 using SHA-384	Optional
ES256	ECDSA using P-256 and SHA-256	Optional
ES384	ECDSA using P-384 and SHA-384	Optional
ES512	ECDSA using P-521 and SHA-512	Optional
PS256	RSASSA-PSS using SHA-256 and MGF1 with SHA-256	Optional
PS384	RSASSA-PSS using SHA-384 and MGF1 with SHA-384	Optional
PS512	RSASSA-PSS using SHA-512 and MGF1 with SHA-512	Optional
none	No digital signature or MAC performed	Optional

A JWT is then a Base64URL encoded header, a Base64URL encoded payload and the signature putting all together and separated by dots. In our proposed protocol, we use a JWT to guarantee the integrity of the rights granted by the identity provider to one application installed in the personal computer.

5.3 Target Threat Model

Figure 5.1 – Target Threat Model

The picture and media protocols limit the access to Android system files. But, all the existing transfer protocols in Android devices (USB Mass Storage, picture transfer protocol and media transfer protocol) allow an application running in the computer and requesting connection to the Android device to transfer the Android files. Let us consider the data exfiltration model based on a client-server TCP/IP architecture as

presented in the figure 5.1. In this model, the attacker can exfiltrate data from the Android devices connected to personal computers over USB. The client application is installed on victim computers to interact with connected Android devices. It is a malicious application that can be installed from the Internet such as a virus. Let us assume that the anti-virus cannot detect this application. The server application resides on a remote computer controlled by the attacker. A socket for each client requesting connection is created by the server application to facilitate data exfiltration. The client application monitors events occurring on the target computer, such as the connection and disconnection of Android devices to and from USB ports. When the Android device is connected, the client application starts requesting for device media and picture files. Then, it sends files to the server application, which is actively listening for incoming connections. Therefore, the data exfiltration attack is launched and the server application gets data stored in the Android devices.

5.4 The proposed protocol

The proposed approach allows a secure communication between the Android IOT devices and applications running in the computer that request a connection to the Android device, to get access to user's files. Thus, we define a protocol that controls access to files stored in the Android device. It is designed on top of a USB transfer protocol (MTP and PTP) and a web authorization protocol. It involves three entities: the Android IoT devices, the Computer and the Authentication Server (AS) (see Figure 5.2). When the application installed in the computer requests a connection to the Android IOT devices, our proposed protocol obliges it to send its *id* and an authorization token to be able to access to media or picture data stored in the device. The token was delivered to the application by the AS after being authenticated in the registration phase. The id is used for the application authentification and the token is used for the application authorization. The Android device sends this information (*id* and authorization token) to the Authentication Server. The AS responds favorably to the Android device when the application is registered and the authorization token is valid. In this case, the application can access to Android media or picture files. When the application id is not known by the AS or the token is not valid, the AS responds unfavorably. In this case, our protocol blocks application access to Android files and we notify the user that he/she is probably under a data exfiltration attack. So, our protocol allows detecting malicious application that exploit trusted relationship between the personal computer and an Android device to get sensitive data.

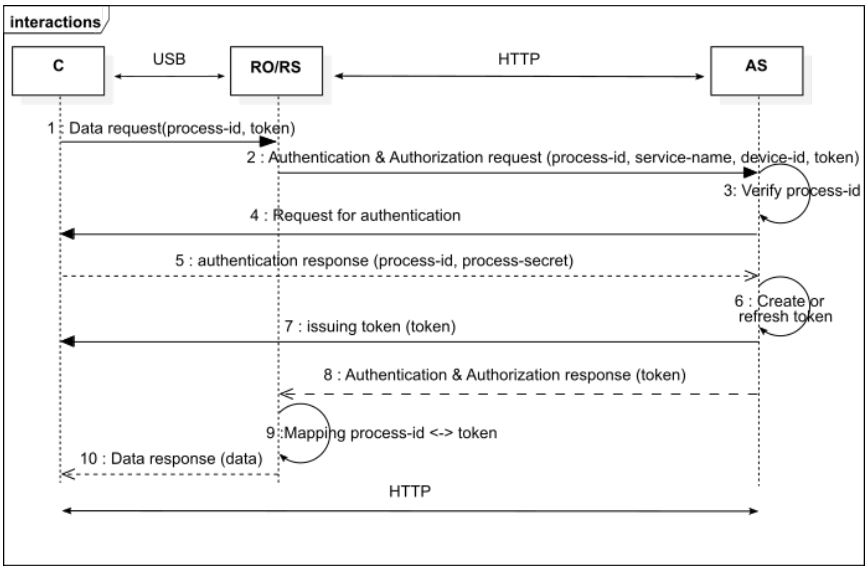


Figure 5.2 – The proposed protocol

5.5 Protocol design and deployment

The following sections explain how to design the proposed protocol that involves the Android IoT devices, the Computer and the Authentication Server entities to detect and react to data exfiltration attacks.

5.5.1 Android IoT devices

We modify the Android OS code to implement our protocol. We instrument the Java class UsbSettings in the package *"com.android.settings.deviceinfo"*. When the device is connected, Android blocks the access to media and picture files until the user chooses the type of transfer protocol. After having selected the type of transfer protocol (MTP or PTP), we verify the application installed in the personal computer identity by asking the Authentication Server. If the one installed in the personal computer is authenticated and has a valid token, we authorize the access to the device data exclusively to the given application. Since we are preventing the mounting process of the file system, awaiting for the application authentication, we create a **RAM disk** with the following characteristics:

- A **5 MB** of memory, which is big enough to contain a file with an *id* and a *token*.
- A **temporary file storage facility (ramfs)** type of memory. We can also use **tmpfs**, either way this virtual disk has a very short lifetime.

- And a **mount point**. Here the disk is mounted on */mnt/ramdisk*

We consider the application malicious only when the token has been modified. When detected, the user is notified that he/she is probably under a data exfiltration attack. When the application is not known by the authentication server, the response message invites this latter to register itself. In both cases, we block access to media and picture files. The user must update his Android system to integrate our protocol.

5.5.2 Computer

The device delegates the authentication and authorization to the Authentication Server. The application on the computer will not have access to data unless the AS responds favorably to the Android device authentication and authorization request. To be able to perform the authentication, the application should be registered to the AS and then communicate its *id* to the Android system device. When initiating data request on the device, in order to receive a fresh token, the application will have to send a token *null* with its *id*. The implementation of this step is possible with the MTP library (libmtp) and the PTP library (libptp). It consists of writing on the unique file shown by the device, the application id and token. Before generating a new token, the application will have to authenticate itself. The case with a null token corresponds to the first time a registered application is requesting for a device data. If the authorization token is outdated but not modified, a new application will be forged and delivered also after having been authenticated.

5.5.3 Authentication Server

The authentication server is the trusted third party communicating via HTTPS and in charge of delivering and verifying tokens. In a registration phase, the application is granted an *id* and a *secret*. The editor of the application will have to register to the AS with information of his corporate like: the official website, an email address. The exhaustive list of information required for the registration depends on each AS. In our implementation, the editor is registered with an email address. The token is forged on the demand of the android device when the user authorizes access to its data. The application communicates its *id* to the android device when initiating the data request. In fact, the server needs to make a link between an application, an android device, the type of data requested (MTP or PTP) and a period of validity. This information is represented in a way that it only makes sense for the AS before being issued to the authenticated application. We use JWT [Jones et al. 2015b] to forge the token because

Table 5.2 – The authentication server responses

	id	*token*	application state	AS responses
1	unregistered	all cases	all cases	error : unregistered
2	registered	null	authenticated	fresh token
3	registered	null	not authenticated	exfiltration attack
4	registered	is modified	all cases	exfiltration attack
5	registered	not modified & valid	all cases	access allowed
6	registered	not modified & not valid	all cases	authentication

this formalism guarantees that the information represented by the token cannot be modified. We use PyJWT [Lindsay 2011] to implement the JWT token. There are 6 types of AS responses depending on the state of the application and the two sent parameters: *id* and *token*. Those responses addressed to the Android device are depicted in table 5.2. If there is no *id* communicated, then we consider that the application is not supporting the protocol. It is therefore, redirected to the registration page. We consider the state *not authenticated* only when the application failed to authenticate itself.

5.6 Evaluation and Results

In this section, we evaluate the security and the performance of our proposed solution.

5.6.1 Security evaluation

In this chapter we are evaluating the security of the protocol. To include all scenarios, we model the protocol considering that the token is refreshed on each run. That includes the case when the token is null. As the main goal of this protocol is to secure USB data transmission, we model HTTP communications as secure channels. Those channels are out of the scope of the attacker because they are meant to be secured with the use of protocols like TLS.

Model

We state the security goals of the protocol from each of the three entities point of view (application, device, server) and in terms of messages sent and received over the

protocol. We model the protocol as depicted in Fig. 5.3 and verify that the security properties are fulfilled using **ProVerif** tool.

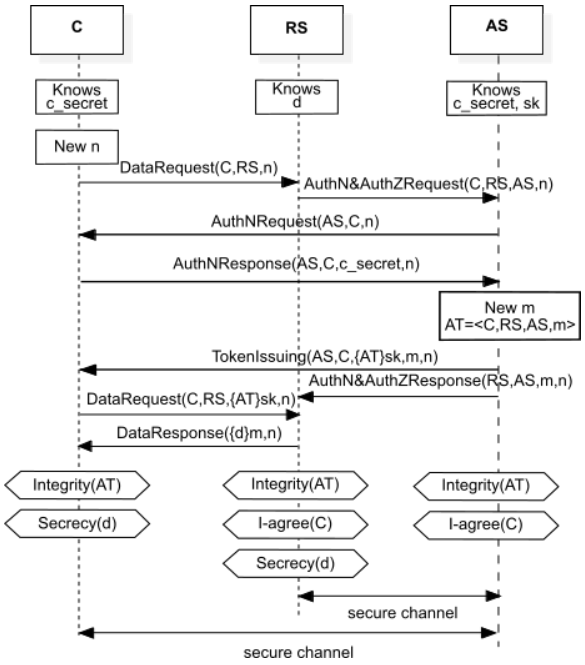


Figure 5.3 – Security model of the protocol.

ProVerif.

ProVerif [Kobeissi et al. 2017] is an automatic cryptographic protocol verifier, based on the formal model (so called Dolev-Yao model). This protocol verifier is based on a representation of the protocol by Horn clauses. Its main features are:

- It can handle many different cryptographic primitives, including shared- and public-key cryptography (encryption and signatures), hash functions, and Diffie-Hellman key agreements, specified both as rewrite rules or as equations.
- It can handle an unbounded number of sessions of the protocol (even in parallel) and an unbounded message space. This result has been obtained thanks to some well-chosen approximations. This means that the verifier can give false attacks, but if it claims that the protocol satisfies some property, then the property is actually satisfied. The considered resolution algorithm terminates on a large class of protocols (the so-called "tagged" protocols). When the tool cannot prove a property, it tries to reconstruct an attack, that is, an execution trace of the protocol that falsifies the desired property.

Security goals.

We assume that the attacker cannot break the cryptographic construction used to make the secure channel. The protocol guarantees the following security properties:

- **Secrecy:** If an application data message m is sent over the channel c, between an honest client C and an honest server S, then this message is kept confidential from an attacker.
- **Integrity:** If an application data message m is sent over the channel c, between a honest client C and an honest server S, then this message can be seen but cannot be modified by an attacker. The security property holds even if the message m was given to an attacker.
- **Authentication** via
  - **injective agreement:** This security property holds if each event is executed in the order defined by the protocol and for all n, each event from run n is different from events from run $n + 1$.
  - **integrity of m:** The authentication property is satisfied if the injective agreement holds and if the message "m" has not been modified.

Those security properties prevent from replay and man-in-the-middle attacks. In listing 5.1, we can see a part of the ProVerif model of our protocol. The public and private channels are respectively *c* and *cs*. We have 3 roles: application client, Android resource server and Authentication Server. We then declare 3 processes: *ProcessC*, *ProcessRS* and *ProcessAS*. Only *ProcessC* is depicted on that listing. Security properties are declared according to the attacker knowledge and events. The types *bitstring* is a predefined one in ProVerif unlike *skey* which must be declared.

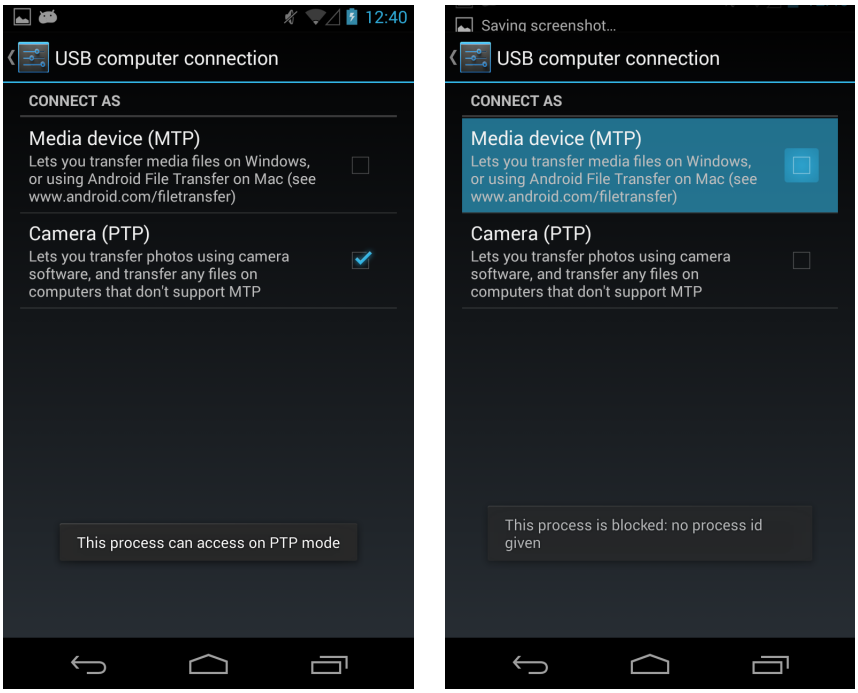
Listing 5.1 – Sample of the protocol modeled with ProVerif

```
1  (* Declaring honest host names C and RS and AS *)
2  free C, RS, AS: host.
3  (* Declaring channels *)
4  free c: channel.
5  free cs: channel [private]. (* secure channel *)
6  (* Declaring private names. *)
7  free c_secret, d:bitstring [private].
8  free sk:skey [private].
9  (* Declaring functions *)
10 fun enc(bitstring, skey):bitstring (*symmetric encryption*)
```

```
11 fun dec(bitstring, skey):bitstring (*symmetric decryption*)
12 equation forall x:bitstring, y:skey; dec(enc(x,y),y) = x (*equational
       theory for symmetric key encryption*)
13 ...
14 (* Declaring events*)
15 event startC(bitstring,host,host,bitstring);
16 event endAS(bitstring, bitstring)
17 (* Declaring security properties*)
18 not attacker(bitstring d). (* Secrecy assumptions *)
19 query a:bitstring, b:host, c:host, d:bitstring; inj-event(endAS(a,d)) ==>
       inj-event(startC(a,b,c,d)). (* Injective-agreement assumptions *)
20 ...
21 (* Queries : verify security properties*)
22 query attacker(d); (*Verify the secrecy of 'd'*)
23 ...
24 (*Application client role*)
25 let ProcessC(c_secret)=
26 new nc:bitstring;
27 out(c, (nc, C, RS, AS));
28 in(cs, (x:bitstring, y:host, z:host));
29 out(cs, (nc, AS, C, c_secret));
30 in(cs, (x1:bitstring, y1:bitstring, z1:bitstring));
31 out(c, (n, C, RS, z1)
32 in(c, x2:bitstring)
```

5.6.2 Results

We tested the cases presented in the table 5.2. In the first case, the application (process running in the computer) does not give an *id*. Our protocol blocks the access to Android files (see figure 5.4 (b)). In cases 2 and 5, the application can transfer Android media and picture files (see figure 5.4 (a)). The application is registered and authenticated in case 2. In case 5 (see Listing 5.3), it is registered and the token is valid and not modified. Cases 3 and 4 present attack scenarios. The application is not authenticated (in case 3) and the token is modified (see Listing 5.2). In both cases, we block access to Android user files. In case 6 (see Listing 5.4), the authentication of the application is required.



(a) (b)

Figure 5.4 – Android system device notifications

Listing 5.2 – Case 4: id registered and modified token

```
10.0.2.2 - - [02/Feb/2018 14:05:34] "POST /mtp/123/123456789 HTTP/1.1"
200 -
> The process 123456789 wants to get access to 123 in mtp mode
process 123456789 registered
Token decode result : This token has been modified !
123 may be under exfiltration attack !!!
```

Listing 5.3 – Case 5: id registered and valid token

```
10.0.2.2 - - [02/Feb/2018 13:36:17] "POST /ptp/123/123456789 HTTP/1.1"
200 -
> The process 123456789 wants to get access to 123 in mtp mode
> Verifying process_id : 123456789
process 123456789 registered
Token decode result : {u'iss': u'123', u'rec': u'123456789',
  u'sub': u'mtp', u'exp': u'201802022243'}
This token is still valid. Process 123456789 can have access
```

Listing 5.4 – Case 6: id registered and token is no more valid

```
10.0.2.2 - - [02/Feb/2018 14:05:53] "POST /mtp/123/123456789 HTTP/1.1"
200 -
> The process 123456789 wants to get access to 123 in mtp mode
> Verifying process_id : 123456789
process 123456789 registered
Token decode result : {u'iss': u'123', u'rec': u'123456789', u'sub':
u'mtp', u'exp': u'201802020910'}
This token is no more valid. Starting 123456789 authentication...
```

5.6.3 Performance evaluation

We evaluated the performance impact of the proposed architecture on the computer and on the Android device. As we can see on the figure 5.5, we have queried 50 times the AS in each case depicted in table 5.2. We have an average of the response time for each type of query. The zeros are not taken into account in the average calculus. They are representing a slow-down in the Android system due to an important use of memory resources by the application in charge of the test. In fact, the test consists of querying the AS every 5s, which requires a lot of the device resources. The experiment shows that this architecture is requiring about 1.5 milliseconds to make a response.

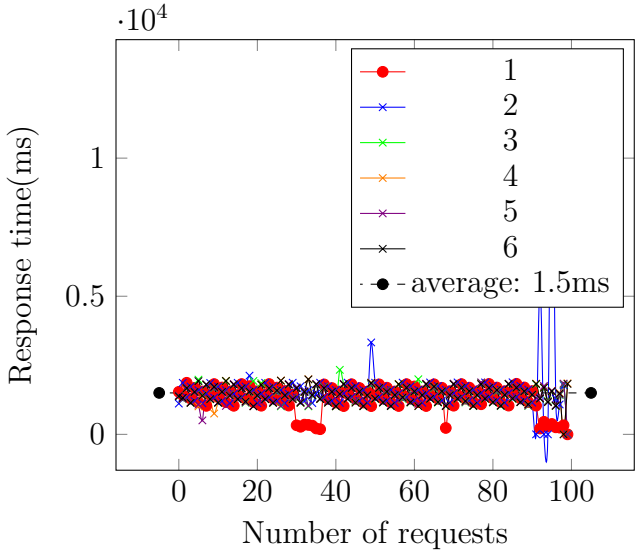


Figure 5.5 – Evaluation of A.S response time

In addition, we use the CaffeineMark [Pendragon 1997] to evaluate the influence of the protocol execution on the system performance. We test the not modified Android overhead when the MTP and PTP protocols are executed. Then, we test our Android

modified system overhead when our protocol is executed. We observe that our proposed protocol generates 2.5 percent execution time overhead with respect to the unmodified system. Thus, it does not really affect the performance of the system.

5.7 Related work

In this section, we present data exfiltration attacks. We also discuss existing countermeasures. [Do et al. 2015] present an adversary model for Android covert data exfiltration using communication mediums (SMS and audio) on mobile devices. [Spolaor et al. 2017] demonstrate how an adversary can exfiltrate data from the smartphone via a USB charging cable using an Android application that encodes sensitive information and transmits it via power bursts back to the public charging station. USBee [Guri et al. 2016] uses the USB data bus to generate electromagnetic signals, modulate and exfiltrate digital data over these signals. MACTANS [Lau et al. 2013] is an implementation of a malicious USB charger that injects a Trojan horse app with a payload to compromise an iOS device. All these data exfiltration attacks use malicious application installed in the phone to collect user data. In our approach, the malicious process is running in the personnel computer and exploit the USB connection to obtaint sensitive data stored in the Android phone. [D'Orazio et al. 2017] present the same data exfiltration attack from iOS devices and not from Android system. Many works exist in the literature to detect data exfiltration attacks. [Sharma et al. 2013] describe a framework to detect potential exfiltration events caused by infected USB flash drive on a machine. The detection system flags alerts based on temporally-related anomalous behavior detected in multiple monitored modules. GoodUSB [Tian et al. 2015] enforces permissions of devices by encoding user expectations into USB driver loading. GoodUSB includes a security image component and a honeypot mechanism for observing suspicious USB activities. All solutions cited above assume that maliciousness comes from devices connected to the computer. In our approach, the malicious process is installed in the personnel computer and not in the device. Many dynamic taint analysis approaches defined in smartphones like TaintDroid [Enck et al. 2014], AppFence [Hornyack et al. 2011], [Graa et al. 2012] and [Graa et al. 2016] allow detection of sensitive data leakage by third party applications running in the Android device using the data tainting. [Wang et al. 2012] enforce security policies on data flows for Android applications to prevent unauthorized USB hardware flows. [Hwang et al. 2015] propose the use of static analyzer to detect a malicious service installed from an infected PC using ADB [Liu 2014]. However, these solutions cannot detect data exfiltration attack performed by application requesting connection to the Android IoT device installed

in the personal computer. When the previous methods assume that the relationship between the personal computer and an Android device is trusted, our approach is the exact opposite. In fact, we assume that each program alien to the device needs to have the consent of the user before getting access to its data.

5.8 Conclusion

The transfer protocols defined in the Android system such as MTP and PTP can be bypassed by exploiting data exfiltration attacks. We have improved these protocols to protect sensitive user data stored in the Android IoT device. Our approach allows detection and response to data exfiltration attacks. We control identity of applications running in the computer and requesting connection to the device. The data transfer is allowed only if the application is registered and the authorization token is valid. We block access to sensitive data and we notify the user that he/she is probably under a data exfiltration attack when the application id is not known by the AS or the token is not valid. So, our protocol allows detecting malicious application that exploit trusted relationship between the personal computer and an Android device to get sensitive data. We evaluate the security and the performance of our proposed solution. We prove that our proposed protocol ensures secrecy, integrity and authentication. Those security properties prevent form replay and man-in-the-middle attacks. The experiment shows that the AS is requiring about 1.5 milliseconds to make a response. We observe that our proposed protocol generates 2.5 percent execution time overhead with respect to the unmodified Android system. Thus, it does not really affect the performance of the Android system.

CHAPTER 6

Conclusions and Perspectives

In conclusion, we give an overview on how the different research objectives presented in the introduction have been followed as well as the different contributions which have resulted. Afterwards, we reflect on how our contributions can be improved and provide new research directions.

Our main objective in this thesis was to provide strong and secure bridges to make the Internet of Persons, the Internet of Things and the Internet of Services meet. That requires on a first time to have a strong definition and knowledge of the boundaries of each concept. And on a second time, to consider security from the weakest party of the system, in our case, from a constrained machine point of view. We reach our goal in 4 steps.

The first step consists of defining a boundary between persons and connected devices. In Chapter 2, we introduce the Discovery and Registration Protocol (DIRE) for Persons' and Things' identity binding. The key element of this chapter is that connected things should have their own identity giving that they are strongly attached to their manufacturer. This consideration is essential to preserve the consent of the user which is - in cases excluding machine-to-machine communications - a person. The consent security property is the key to prevent abusive usage of rights or data exfiltration. In addition, like [Torres et al. 2013], we think that the Identity Management System corresponding to the Internet of Persons, Things and Services should neither be network-centric nor service-centric but user-centric, and should by default minize data disclosure. That being said, we still have to find the tools to implement this concept.

For the second step, we present in Chapter 3 what can be the tool to implement in an efficient way DIRE. We call it CoAP 2.0, an enhancement of the Constrained Application Protocol (CoAP), a well-known protocol in the IoT world for its thrifty impact on device battery and its complementarity with HTTP.

The third step consists of getting rid of a centralize architecture for ciphered communications. In an architecture involving tens of billions of constrained devices, it is not reasonable to consider a certificate chains as a long term solution guaranteeing the secrecy of communications. In fact, it has been proven that malicious actors can easily be part of the chain and then validate fraudulent certificates. We then propose in chapter 4, a certificate-less peer-to-peer secret key establishment limited to near field devices.

The fourth step consists of preventing data exfiltration from the most adopted device: smartphones. In chapter 5, we propose a method preventing malicious programs on the user's computer, from mounting the Android file system without the knowledge of the user. We bring back consent to smartphone users and force program editors willing to have access to smartphone filesystems, to be officially registered. That way it would be easier to identify them and verify whether there is a logic between their activity and the need for personal data.

6.1 Perspectives

The research described in this thesis can be extended along several directions.

- First of all, we are currently exploring the possibility to push our recommendations in a better CoAP, depicted in chapter 3, for a new IEEE standard for CoAP. That would help in implementing efforts and foster the use of CoAP not only on top of Wi-Fi but also on LPWAN for instance.
- A second perspective would be to have a complete implementation of the certificate-less key exchange protocol for a detailed evaluation of this architecture compared to a PKI. In chapter 4, the main concern of the implementation was memory consumption on constrained devices. It showed encouraging results. For a better understanding of our work, we assumed that the constrained device is temper resistant, which means that the secret keys used to cipher communications are securely stored on the device. We should now consider to secure the keys

thanks to a Secure Element (SE). Pascal Urien in [Urien 2016] proposes an SE-based architecture consisting of a third party playing the role of the authentication authority and a client/server user-agent embedded in the constrained device, using DTLS over CoAP. This second perspective also offers the opportunity to evaluate the impact of the SE in our architecture.

- As a third perspective, it would be interesting to see the method proposed in chapter 5 be applied on an IOS environment and pushed as a standard to prevent smartphone data exfiltration from computers. The future of software distribution is in marketplaces like Apple Itunes [Apple 2001], Google Play [Google 2008], Windows app. store [Microsoft 2004] or Sourceforge [Sourceforge 1999] (for open source distribution). It would then be interesting to see how an Authorization/Authentication Server, in charge of the verification of the computer's application can be set up within those platforms.

- As an ultimate perspective, we explore the possibility to guarantee more, the security property of consent when using connected devices. We patented a solution which consists in bringing that security property on cashless transactions involving connected devices. We give an overview of the content of the patent in section 6.2

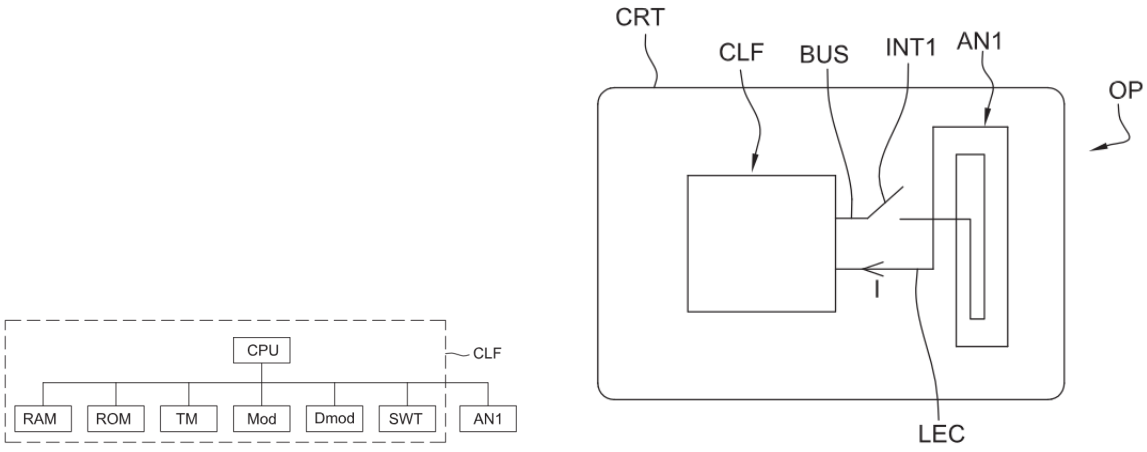
6.2 Bringing Consent Devices for Cashless Transactions (INPI file number: 17 60333)

The Context

Cashless payments are more frequently used, and even have become the norm. This payment method has become the standard in banking services, especially in debit cards, thanks to "contact-less" payments. Within the event market (concerts, exhibitions, amusement parks, etc.), cashless payments are also in demand. Currently, there exists multiple companies which satisfy the demand for cashless payments.
While banks prefer using the smartphone as a support for cashless payments, the event market has turned towards elastic bracelets, which are simpler for the user. However, a security analysis of these bracelets, as well as debit cards, shows the absence of the consent property. In fact, a payment could take place without the owner's knowledge with its card or bracelet. To succeed, the usurper doesn't need to steal the object from the owner, just as long as they come in close proximity to the owner.
We propose a method to retrieve the consent property while keeping this technology practical for bracelets and cards. The main elements of the RFID technology are the

antenna and the microcontroller, the microcontroller is powered by the antenna through the effect of the induced current. The proposed solution introduces a third element, the switch, that the user pushes to activate the object to perform a transaction. The figure 6.1 gives a schematic of the pattented solution.
There are two possible methods to realize this switch: a hardware and a software based solution, both depicted in Figure 6.2.

- The hardware solution consists in mechanically disconnecting the antenna from the chip, the user activates the switch to establish contact while the transaction is taking place.
- The software solution consists in delivering the information contained on the chip if and only if the switch had been activated.



(a) Principal componants of an OP (b) General overview of the OP

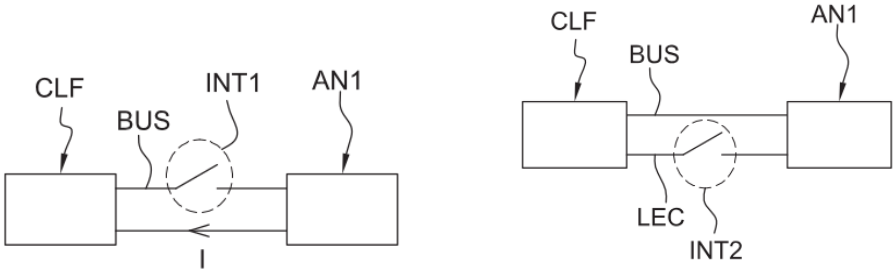
Figure 6.1 – Schematics of the pattent

Realisation modes

1. **The Software Mode**. It consists in steps referenced from E1 to E5. In the initial step E1, the user equipped with the object approaches the terminal BRN and enters in the magnetic field produced by antenna AN2 of terminal BRN. In the next step, the induced current I is generated in the first antenna AN1. Next, the microcontroller is powered by the induced current in step E2. At this point, the microcontroller is in receiving mode and can receive messages. In step E3, while it is being powered, the object receives a query message MINT emitted from the reader. During the step E4, the microcontroller emits a response REP over and

over. The switch is opened by default, so the response REP is not emitted. We assume the owner of the object pushes the button to close the switch. During step E5, the switch closes and response REP is transferred to the terminal BRN.
In another variant of this first mode, the microcontroller verifies the state of the switch before transmitting the response, if the switch is open the microcontroller suspends the transmission. The duration of the transmission can be limited in time. The variant can prove to be useful when using a biometric sensor like fingerprint as a switch.

2. **The Hardware Mode**. Like the Software Mode, the Hardware Mode consists in 5 steps. In step E1, the induced current I is generated in the first antenna AN1. As the switch is open, the microcontroller is not powered, therefore, cannot receive data from the reader causing it to not respond to the terminal BRN. We now assume that the user of the object pushes the button to close the switch. In step E2, the microcontroller is powered by the induced current. At this point, the microcontroller is in receiving mode and can receive messages. In step E3, while it is being powered, the object receives the query message MINT from the reader. During the fourth step E4, the microcontroller emits a response REP which is transmitted to the terminal BRN in step E5.



(a) Software Mode (b) Hardware Mode

Figure 6.2 – Different Realisation Modes

Another variant which can apply to both Software and Hardware Modes is for example, when the object generated an induced current, a notification signal can be emitted to inform the user that the object is in close proximity to the terminal. The user can push the button to activate the switch once he receives the notification signal.

6.3 Conclusion

To conclude, I would like to say that this research has been a great opportunity to investigate a wide variety of concepts, models and technologies in the information security domains. We provided novel approaches in response to IoT security challenges, and we have shown that our proposed work is encouraging research field.
Finally, we believe that interfacing the Internet of Persons, Things and Services is still plenty of challenges, of paramount importance, and full of opportunities. The Internet of Things opens up a new door to a flow of information still under exploited. In my opinion, we need to think use-case and security side by side in order to make good standards, before applying sophisticated concepts like machine learning, to finally create new use-cases and opportunities.

APPENDIX A

Identité et Consentement dans l'Internet des Personnes, des Objets et des Service

A.1 Introduction

Dans un premier temps, notre attention est dirigée vers les nouveaux cas d'utilisation impliquant les objets connectés et les personnes. Loin de la tendance actuelle qui consiste à calquer les mécanismes de sécurité de l'Internet des Personnes à l'Internet des Objets, par exemple, considérer qu'une personne et un objet devraient avoir la même identité numérique; notre travail définit l'identité de l'objet et la relation de filiation entre la personne et les objets connectés à Internet. Dans ce but, nous proposons d'abord un protocole permettant à une personne d'effectuer des achats automatiques en déléguant les permissions à un objet connecté. Le principal problème à maitriser est l'association sécurisée de l'identité des objets et des personnes. Nous avons prouvé formellement que le protocole préserve les propriétés de sécurité d'Intégrité, d'Anonymat et de Confidentialité. Cette première contribution crée un pont entre l'IoT et l'IoS à travers la délégation de l'autorisation. Par conséquent, les personnes pourront reprendre le contrôle sur leurs données car la solution proposée est basée sur une architecture permettant aux personnes de définir une politique de contrôle d'accès plus fine.

Deuxièmement, pour mettre en place de nouveaux cas d'utilisation comme celui précédemment énoncé, nous avons besoin d'une couche protocolaire qui s'y prête. Nous

proposons un protocole sur la couche applicative s'appuyant sur le User Datagram Protocol (UDP) et construit autour des trois principales préoccupations de l'IoT : la **Découverte** (Advertisement), les **Requêtes Synchrones et Asynchrones** et les **Notifications**. Nous proposons, d'améliorer le Constrained Application Protocol (CoAP) en le renforçant avec les atouts des autres protocoles très répandus de l' IoT comme MQTT et mDNS.

Finallement, nous abordons le problème de la sécurité des données gérées dans les objets connectés dits contraints. Nous avons à cet effet défini une couche de sécurité permettant à deux objets ou plus d'établir des clés cryptographiques partagées et de chiffrer les communications en s'affranchissant du poids et du coût supplémentaire de l'infrastructure à clés publiques (PKI). Selon Gartner [Gartner 2017], dans une décennie environ, le nombre d'objets connectés atteindra le milliard, ainsi le challenge est de sécuriser les communications de l'IoT, dans une architecture sans certificat, grâce au très répandu algorithme de Diffie-Hellman. Notre solution exploite la délégation des solutions d'authentification pour authentifier les parties, puis les capacités des objets connectés à avoir plusieurs cartes réseaux pour créer des clés partagées sur des canaux privés.

A.2 Protocole de Découverte et d'Enregistrement (DIRE): Pour la gestion de l'Identité des Personnes et des Objets dans l'Internet des Objets

Dans les architectures actuelles de l'IoT, les utilisateurs partagent leurs identifiants avec les objets connectés pour avoir accès aux services. Cependant, les ressources limitées de ces objets en terme de mémoire et de puissance de traitement rendent difficile l'application de mécanismes de sécurité tels que les algorithmes de chiffrement et de signature puissants. En fait, les mécanismes d'authentification actuels des objets connectés à internet peuvent facilement être contournés et ces dispositifs peuvent être infectés par un malware tel que Mirai [MalwareMustDie 2016], ce qui en fait un botnet pour des attaques reseau à grande échelle [Hunt 2016] [Muresan 2016]. En plus, lorsque l'objet connecté et l'utilisateur ont la même identité, le comportement de l'appareil peut être modifié par le fabricant sans demander à l'utilisateur. Par exemple, les utilisateurs de Fitbit partagent involontairement des détails de leur vie sexuelle avec le monde parce que le fabricant rend cette activité sexuelle publique par défaut [thenextweb 2013]. Nous proposons donc un protocole original qui assure une gestion sécurisée des iden-

tités des appareils et des personnes. Le protocole que nous proposons nous permet de découvrir et d'enregistrer des dispositifs intelligents au près d'un fournisseur d'identité d'une manière transparente et conviviale.

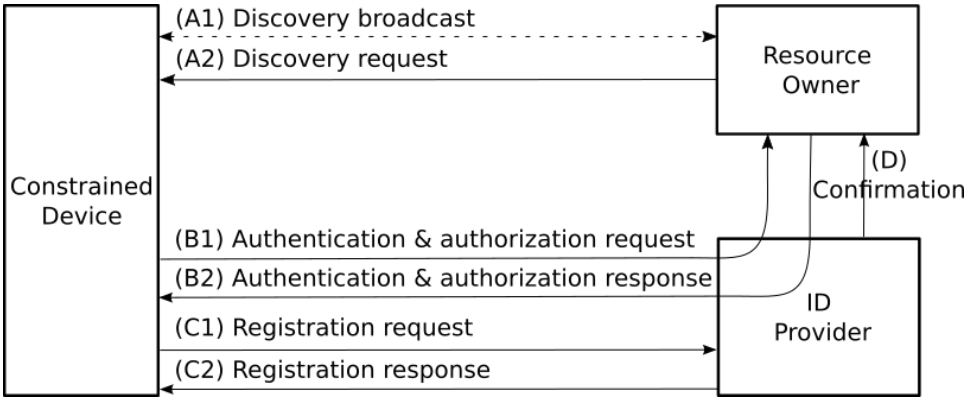


Figure A.1 – Discovery and Registration protocol flow

L'approche proposée consiste à découvrir puis enregistrer un nouvel appareil connecté au près du fournisseur d'identité préféré de l'utilisateur. Pour y parvenir, nous utilisons deux paradigmes importants : Le premier repose sur l'idée que chaque utilisateur et appareil connecté a une identité. Si nous avons une idée de ce à quoi ressemble l'identité de l'utilisateur, il convient de définir une identité de l'objet grâce au Things Description Document (TDD), avant de décrire le protocole. Le deuxième paradigme est basé sur l'idée que l'utilisateur et l'appareil ne se font pas confiance, donc, chaque acteur doit prouver son identité. Nous utilisons OAuth2.0 pour y parvenir. Nous avons construit le protocole DIRE représenté dans la figure A.1, basé sur les deux paradigmes énumérés ci-dessus.
Le propriétaire des ressources utilise son user-agent pour gérer l'ensemble du processus. Il déclenche la découverte (A) puis le périphérique utilise les privilèges du fabricant pour demander au propriétaire de la ressource de s'authentifier auprès du fournisseur d'identité (B). La même requête est utilisée pour demander l'autorisation d'accéder aux ressources. Une fois cette authentification mutuelle effectuée, l'appareil demande son enregistrement auprès du fournisseur d'identification (C). Enfin, le fournisseur d'identifiant envoie au propriétaire de la ressource un message concernant le status de l'enregistrement (D).

A.2.1 La Découverte

La phase de découverte décrite dans le diagramme de séquence de la Fig. A.2, a pour but de receuillir l'identité de tous les dispositifs disponibles sur un réseau. Pour des

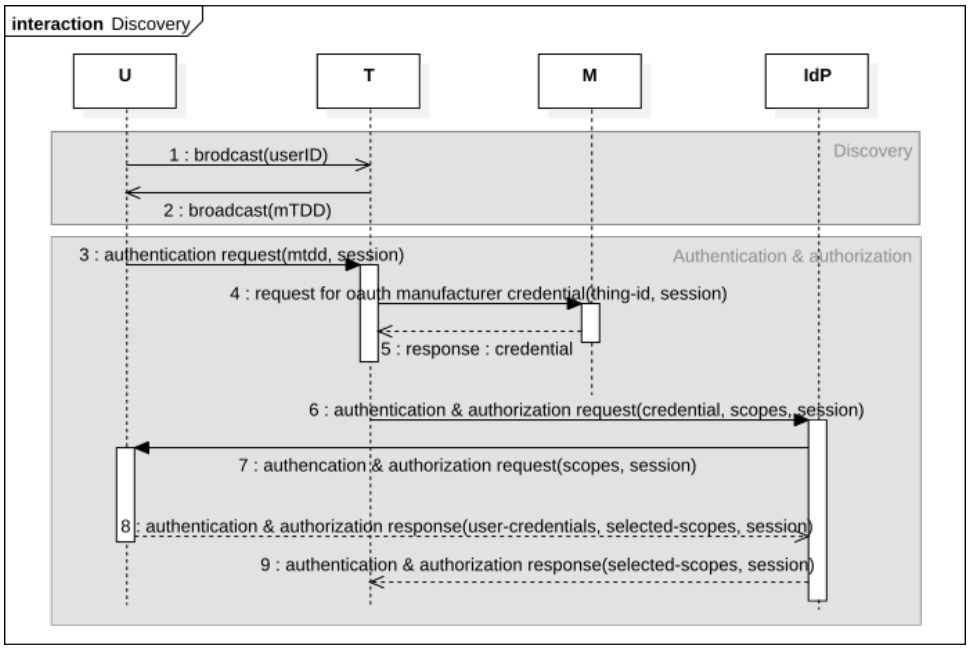


Figure A.2 – Discovery sequence diagram

raisons de confidentialité, l'utilisateur et les appareils divulgueront uniquement des informations publiques jusqu'à ce que les deux parties soient respectivement authentifiées. Pendant que l'utilisateur diffuse son *user_id*, le périphérique diffuse une version minimale du l'identité du périphérique (mTDD). Le principe de l'authentification mutuelle repose sur l'affirmation suivante : Si l'utilisateur reçoit une demande d'authentification de son fournisseur d'identité - déclenchée par le périphérique choisi pour être enregistré - alors l'identité du périphérique est vérifiée. Et si l'utilisateur répond correctement à cette requête, son identité est également vérifiée. A la fin du processus de découverte, le périphérique aura authentifié l'utilisateur et vice-versa. Le seul rôle du fabricant à ce stade est de divulguer des secrets, comme le client_id, aux dispositifs demandeurs. Une fois l'authentification mutuelle effectuée, l'appareil déclenche la deuxième phase du protocole.

A.2.2 L'Enregistrement

La phase d'enregistrement représentée dans le diagramme de séquence A.3, vise à lier les identités de l'utilisateur et du périphérique. Le principe de l'enregistrement repose sur l'affirmation suivante : le fournisseur d'identité de l'utilisateur doit attribuer des identifiants au périphérique que l'utilisateur introduit. L'utilisateur a introduit l'appareil dans la phase de découverte en divulguant son identité. Une fois l'autorisation reçue, l'appareil demande son enregistrement avec son TDD comme paramètre. Le fournisseur

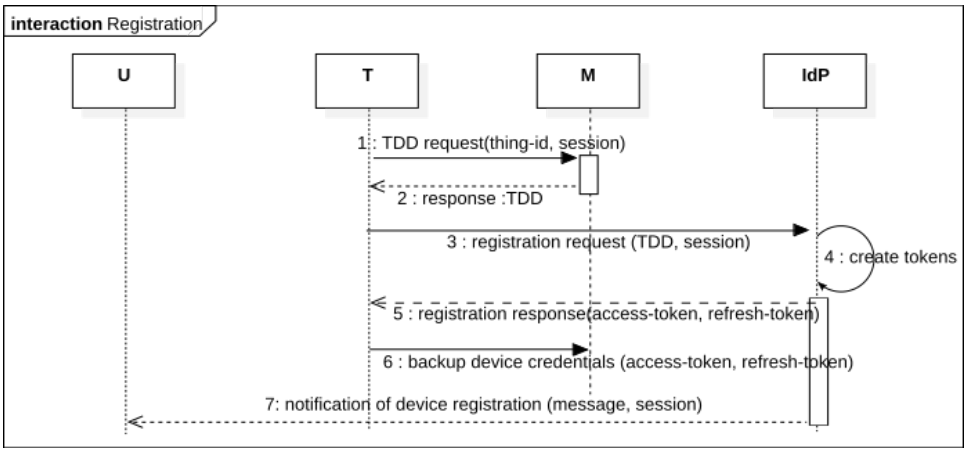


Figure A.3 – Registration sequence diagram

d'identité répond avec les tokens (d'access et de refresh token) correspondant aux services validées par l'utilisateur. Les jetons émis par le fournisseur d'identité sont fabriqués et destinés à être utilisés uniquement par le périphérique. Cela signifie que ces jetons sont associés à un TDD et à un utilisateur dans la base de données du fournisseur d'identité.

A ce stade, le fabricant a deux rôles. Le premier consiste à sauvegarder les informations d'identification du périphérique (tokens) dans une base de données beaucoup plus sécurisée. Grâce à ses privilèges, l'appareil pourra les récupérer en cas de besoin. Le second rôle consiste à reconfigurer le serveur des périphériques et le client en fonction des autorisations accordées par l'utilisateur. À l'issue de cette étape, l'appareil est lié à l'utilisateur et ses actions ont été définies en fonction des droits accordés par l'utilisateur. L'appareil peut alors agir au nom de l'utilisateur avec une identité séparée. Une analyse formelle du protocole DIRE utilisant Event-b et Rodin [Abrial et al. 2010] montre que notre protocole atteint ses objectifs tout en respectant l'intégrité, l'anonymat et la confidentialité comme propriétés de sécurité. Nous avons implémenté ce protocole en utilisant HTTP et MQTT mais de meilleures performances peuvent être obtenues avec quelques changements dans CoAP.

A.3 Améliorations de CoAP pour un protocole plus adapté à l'internet des objets: CoAP 2.0

CoAP est un protocole RESTful qui s'appuit sur UDP et destiné à être utilisé par des appareils connecté à internet et possédant des ressources limitées. C'est généralement le cas des microcontrôleurs à 8 bits avec de petites capacités en ROM et RAM.

Comme défini dans la RFC7252 [Shelby et al. 2014], il est conçu pour les applications Machine-to-Machine et peut être facilement traduit en HTTP pour une intégration simplifiée avec le web des personnes. Il est également décrit comme une version moins verbale de HTTP. Un message CoAP typique est représenté dans la figure A.4. Chaque message CoAP occupe la section de données d'un datagramme UDP. Il se compose d'un en-tête de taille fixe de 4 octets suivi d'un jeton de longueur variable, d'une séquence d'options CoAP et du payload.

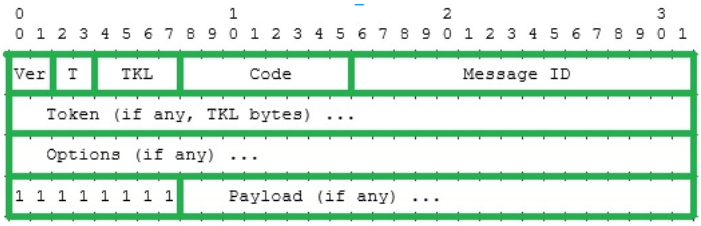


Figure A.4 – CoAP Message Format.

Si CoAP est de destiné à être une version allégée de HTTP, avec des *communications synchrones et asynchrones* efficaces, il reste encore deux autres fonctionnalités à améliorer : la **Découverte** et la **Notification**, pour faire de CoAP, le meilleur protocole pour l'IoT.

A.3.1 La Découverte

L'application d'une politique de contrôle d'accès dynamique nécessite l'identification du périphérique qui fait la requête, non seulement avec ses caractéristiques réseau mais aussi avec ses identifiants. Nous avons donc ajouté, en plus du champ **FILTER**, le champ **QAUTHORIZATION** d'une longueur variable à la requête mDNS pour réaliser le **CoAP2 Discovery Query** représenté dans le tableau A.1. L'autorisation dans notre proposition doit être délivrée par un serveur d'autorisation de type OAuth2.0 et sera vérifiée avant de soumettre toute réponse. Pour parcourir l'ensemble du réseau, le champ QNAME doit avoir la valeur *.local*. Dans ce cas, le client demandeur s'attend à recevoir plusieurs **CoAP2 Discovery Response** avec le champ RDATA rempli de la liste des ressources à découvrir. Cependant, lorsque le champ QNAME a la valeur d'un nom d'hôte spécifique, une seule réponse est attendue.

Le **CoAP2 Discovery Response** a exactement la même structure que le mDNS Resource Record. L'idée est d'effectuer la recherche d'objet et de ressources à l'aide d'une seule requête. Les informations reçues dans la réponse sont :

Table A.1 – CoAP 2 Discovery Query Fields.

Field	Description	Length/Bits
QNAME	The hostname to resolve or just ".local"	variable
QTYPE	Set to 'COAP'	16
U-R	Set to '1'	1
QCLASS	Set to 'IN' for Internet	15
QAUTHZ	The authorization token	Variable
FILTER	list of paths to resolve	Variable

Peut importe la valeur de U-R, si QAUTHZ n'a pas une valeur nulle alors la réponse sera un unicast.

- l'addresse IP de l'objet
- le hostname de l'objet
- les ressources disponible si il y en a.

Nous pouvons par conséquent parcourir le réseau à l'aide d'une seule requête en utilisant la requête sélective. En d'autres termes, dans un réseau de 10 périphériques ou un seul d'entre eux possède la ressource recherchée "lumière". Au lieu d'interroger les 10 périphériques et recevoir 10 adresses IP pour lesquelles il faudra ensuite faire une recherche de ressources, avec CoAP 2.0, seul le périphérique d'intérêt répondra à la diffusion, plutôt que la totalité des appareils.

A.3.2 La Notification

Il y a trois concepts de Pub/Sub : *Topic-based*, *Content-based* et *Type-based*. Ils sont listés du concept le plus simple au plus élaboré. Le système de notification de la version actuelle de CoAP est basé sur le concept Topic-based. Dans CoAP2, nous proposons un système de notification plus intelligent en ajoutant des règles sur la requête d'abonnement (subscription). Passer d'un modèle Topic-based à un modèle Content-based nécessite de modifier : l' *Abonnement*, la *Gestion des Observateurs* et le *Désabonnement*.

L'Abonnement

Avec CoAP, l'abonnement nécessite une méthode GET avec un payload vide. Dans CoAP2, si le champ **Rule** est défini, l'abonné ne recevra que des notifications respectant la règle de d'abonnement. Grâce aux opérateurs de comparaison et des opérateurs logiques (=,<,>,≤,≥,and,or, etc...), les règles peuvent être facilement définies par le client et simplement digérées par le serveur.

La Gestion des Observateurs

Il devrait être possible de souscrire à un chemin avec des règles différentes. L'observateur, le chemin et la règle sont liés par un **numéro d'abonnement**. Cette relation est stockée dans une base de données.

Le Désabonnement

La réponse à une demande d'abonnement réussie est un nombre entier correspondant à un **numéro d'abonnement**. Il est utilisé pour désabonner l'observateur d'un chemin et d'une règle avec moins de données dans la requête. DELETE est utilisé à la place de GET pour la Demande de Désabonnement, afin de supprimer l'entrée correspondant au numéro d'abonnement dans la base de données. La figure **??** montre un exemple d'abonnement et de désinscription à CoAP2.

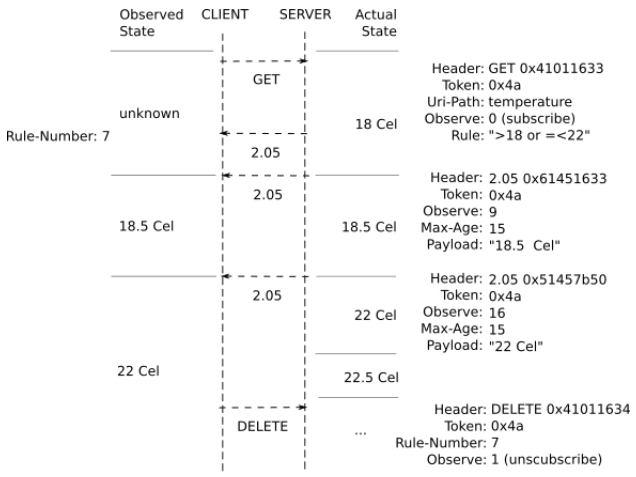


Figure A.5 – A Client Registers and Receives Notifications of the Current State according to the *Rule*: $18 < State \leq 22$

A.4 Un protocole d'échange de clé sans certificat

L'infrastructure à clé publique (PKI) est largement utilisée pour établir un lien entre l'identité de l'utilisateur et sa clé publique. Sachant que l'émission et l'utilisation de certificats sont souvent coûteuses pour l'IdP, il n'y aurait pas d'intérêt à utiliser une PKI dans le contexte de l'IoT, où des milliards de objets connectées devraient rejoindre l'Internet. Cette architecture presenterait plus de difficultés pour la maintenance en plus d'être coûteuse. Nous proposons ensuite une méthode originale, qui permet à deux objets connectés de créer une clé de chiffrement partagée, grâce à l'algorithme de Diffie-Helman et un canal privé. La solution proposée allège les communications cryptées dans l'IoT car elle rend les certificats inutiles. L'analyse des performances de cette solution montre des résultats prometteurs.

Le concept de notre solution est d'utiliser la capacité des objets connectés pour con-

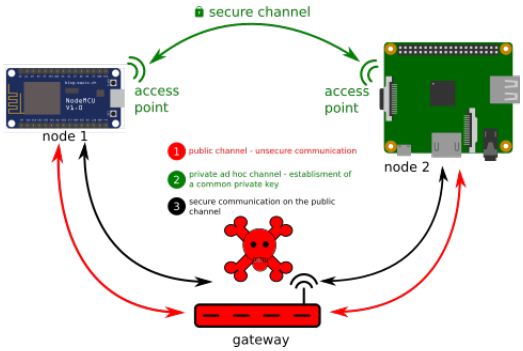


Figure A.6 – Solution overview

struire un point d'accès ad hoc et l'utiliser comme un canal privé. Le canal privé est un point d'accès Wi-Fi protégé par un passphrase WPA (Wi-Fi Protected Access) et configuré pour recevoir un seul appareil connecté à la fois. Le ssid et la passphrase doivent avoir une longueur d'au moins 128 bits pour résister aux attaques par force brute.

Ce nouveau canal éphémère est utilisé pour exécuter une à nouveau l'algorithme de DH de sorte qu'Eve, l'attaquante, ne peut être au courant ni de la nouvelle base ni d'aucun autre élément calculé par Alice et Bob. La figure A.6 donne un aperçu de l'architecture de la solution. Notre solution comporte quatre phases :

L'Authentification. C'est la première étape du protocole et elle a été réalisée sur le canal public. Il s'agit de s'assurer qu'Alice n'engage pas un échange de clés avec Eve, l'attaquant, mais plutôt avec Bob. Alice, qui initie l'échange, doit envoyer une liste des serveurs d'authentification (AS) où l'identité de Bob peut être vérifiée. Bob répond avec un élément de la liste donnée. Vient ensuite l'échange correspon-

dant au hybrid flow OIDC. Nous considérons qu'Alice et Bob sont honnêtes, ce qui signifie que chacun d'entre eux est censé avoir ses références OIDC : (*client_id* et *client_secret*). Après l'authentification de Bob, Alice reçoit un code d'autorisation de l'AS, ce qui prouve l'identité de Bob. Ce code d'autorisation doit être un nonce. De plus, Bob est sûr qu'Alice est digne de confiance parce qu'il a reçu une demande d'authentification du bon AS, juste après avoir initié la communication avec Alice.

Le premier Run. Il s'agit de la première fois que l'algorithme de DH est joué. L'échange se passe sur le canal public, après que chaque acteur se soit authentifié. Le premier message de cette phase, d'Alice à Bob, contient le générateur choisi g en clair, puisqu'il est censé être public. A la fin de ce run, Alice et Bob auront calculé la clé partagée g^{xy}.

Avant de lancer le deuxième run, Alice communiquera à Bob le point d'accès auquel se connecter et un secret crypté avec la clé partagée nouvellement calculée. Alice et Bob devraient être l'un l'autre à portée du Wi-Fi. La portée peut aller jusqu'à 92 mètres pour la bande de 5 GHz.

Le deuxième Run. Cela consiste à faire passer le générateur de g à un nouveau h. La clé partagée finale est alors h^{xy}. Cette étape est toujours effectuée sur le canal privé.

La negotiation du key_id. Chaque acteur est responsable du stockage sécurisé de la clé partagée. Après l'établissement de la clé partagée, Alice et Bob doivent définir un *key_id* commun pour que ni Alice ni Bob n'aient dans leur base de données respective, deux clés avec le même identifiant. Une façon simple de réaliser cette tâche est qu'Alice propose d'abord un numéro et Bob répond avec un numéro différent si celui proposé par Alice correspond à un identifiant dans la base de données.

Une fois la clé partagée établie, il n'est pas nécessaire d'avoir un certificat pour établir la communication. En fait, les communicants n'ont besoin de connaître que le *key_id* correspondant à la clé utilisée pour crypter la communication.

A.5 Conclusion

Notre objectif principal dans cette thèse était de fournir des ponts forts et sécurisés pour lier se la meilleure des manières l'Internet des personnes, des objets et des services. Pour cela, il faut d'abord avoir une définition et une connaissance solides des limites de chaque concept. Il faudrait ensuite considérer la sécurité de la partie la plus faible

du système, dans notre contexte, il s'agit de se placer du point de vue du périphérique contraint.

Nous atteignons notre objectif avec le protocole de découverte et d'enregistrement (DIRE), qui place l'identité et le consentement au cœur de la relation entre les personnes et les objets. Afin d'implémenter pleinement DIRE, nous fournissons le meilleur protocole pour les communications IP dédiées aux appareils contraints : CoAP2. Enfin, nous proposons une solution pour échanger des clés cryptographiques sans avoir recours à une infrastructure à clé publique (PKI). Par ailleurs, dans cette thèse, nous fournissons également une solution pour empêcher l'exfiltration des données des appareils Android ainsi qu'un moyen d'obtenir le consentement de l'utilisateur dans les transactions cashless.

List of Publications

International Conferences

- M. LOBE KOME, M. GRAA, F. Cuppens, N. Cuppens-Boulahia, and V. FREY. DIscovery and REgistration Protocol: For Device and Person Identity Management in IoT - 13th International Conference on Information Systems Security, ICISS 2017, Bombay, India, Dec 16-20, 2017.
- M. LOBE KOME, M. GRAA, F. Cuppens, N. Cuppens-Boulahia, and V. FREY. Detection and response to Data Exfiltration from Internet of Things Android Devices- 24th IFIP World Computer Congress, WCC 2018, Posnan, Poland, Sept 17-21, 2018.
- M. LOBE KOME, F. Cuppens, N. Cuppens-Boulahia, and V. FREY. CoAP Enhancement For a Better IoT Centric Protocol: CoAP2.0. 5th International Conference on Internet of Things: Systems, Management and Security, IoTSMS 2018, Valencia, SPAIN, Oct 15 - 18, 2018
- M. LOBE KOME, F. Cuppens, N. Cuppens-Boulahia, and V. FREY. A certificateless key exchange protocol for IoT. The 13th International Conference on Risks and Security of Internet and Systems, CRISIS 2018, Arcachon, France, Oct 16 - 18, 2018.

Bibliography

[Abrial 2008] J.-R. Abrial. Event-b. 2008. 28

[Abrial et al. 2010] J.-R. Abrial, M. Butler, S. Hallerstede, T. S. Hoang, F. Mehta, and L. Voisin. Rodin: an open toolset for modelling and reasoning in Event-B. *STTT*, 12(6):447–466, 2010. 28, 105

[ACE 2017] ACE. Authentication and authorization for constrained environments (ace). 2017. 40, 51

[Adrian et al. 2015] D. Adrian, K. Bhargavan, Z. Durumeric, P. Gaudry, M. Green, J. A. Halderman, N. Heninger, D. Springall, E. Thomé, L. Valenta, et al. Imperfect forward secrecy: How diffie-hellman fails in practice. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 5–17, 2015. ACM. 67, 68, 130

[Andrew Banks 2014] R. G. Andrew Banks. Mqtt version 3.1.1. 2014. 25, 44, 46

[Apple 2001] Apple. Itunes. 2001. 97

[Apple 2003] Apple Inc. Bonjour printing specification. Technical report, 2003. 44, 45

[Apple 2013] Apple. Multicast dns. 2013. 44, 45

[Apple 2005] S. M. ,Microsoft CorporationApple. Dynamic configuration of ipv4 link-local addresses. 2005. 45

[Armando et al. 2005] A. Armando, D. Basin, Y. Boichut, Y. Chevalier, L. Compagna, J. Cuéllar, P. H. Drielsma, P.-C. Héam, O. Kouchnarenko, J. Mantovani, et al. The avispa tool for the automated validation of internet security protocols and applications. In *International conference on computer aided verification*, pages 281–285, 2005. Springer. 29

[Arzt et al. 2014] S. Arzt, S. Rasthofer, C. Fritz, E. Bodden, A. Bartel, J. Klein, Y. Le Traon, D. Octeau, and P. McDaniel. Flowdroid: Precise context, flow, field, object-sensitive and lifecycle-aware taint analysis for android apps. *Acm Sigplan Notices*, 49(6):259–269, 2014. 79

[Avahi 2010] Avahi. Avahi documention. 2010. 45

[Balfanz 2015] D. Balfanz. Fido u2f implementation considerations. *FIDO Alliance Proposed Standard*, pages 1–5, 2015. 14

[Barki et al. 2016] A. Barki, A. Bouabdallah, S. Gharout, and J. Traoré. M2m security: Challenges and solutions. *IEEE Communications Surveys & Tutorials*, 18(2):1241–1254, 2016. 76

[Berni and Gregg 1973] A. Berni and W. Gregg. On the utility of chirp modulation for digital signaling. *IEEE Transactions on Communications*, 21(6):748–751, 1973. 5

[Bhoyar et al. 2013] R. Bhoyar, M. Ghonge, and S. Gupta. Comparative study on ieee standard of wireless lan/wi-fi 802.11 a/b/g/n. *International Journal of Advanced Research in Electronics and Communication Engineering (IJARECE)*, 2(7):687–691, 2013. 3

[Bigioi et al. 2002] P. Bigioi, P. Corcoran, and G. Susanu. Digital imaging services using ptp (picture transfer protocol). In *Consumer Electronics, 2002. ICCE. 2002 Digest of Technical Papers. International Conference on*, pages 54–55, 2002. IEEE. 80

[Blake-Wilson and Menezes 1998] S. Blake-Wilson and A. Menezes. Authenticated diffe-hellman key agreement protocols. In *International Workshop on Selected Areas in Cryptography*, pages 339–361, 1998. Springer. 75

[Boneh 1998] D. Boneh. The decision diffie-hellman problem. In *International Algorithmic Number Theory Symposium*, pages 48–63, 1998. Springer. 66

[Bormann et al. 2014] C. Bormann, M. Ersue, and A. Keranen. Terminology for constrained-node networks. 2014. 43, 65

[Bradley and Denniss 2017] J. Bradley and W. Denniss. Oauth 2.0 for native apps. 2017. 23

[Bradley et al. 2017] J. Bradley, W. Denniss, H. Tschofenig, and M. Jones. OAuth 2.0 Device Flow for Browserless and Input Constrained Devices. 2017. 41

[Butler and Yadav 2008] M. Butler and D. Yadav. An incremental development of the Mondex system in Event-B. *Formal Aspects of Computing*, 20(1):61–77, 2008. 28

[Cavoukian 2008] A. Cavoukian. Privacy in the clouds. *Identity in the Information Society*, 1(1):89–108, 2008. 17

[Cervesato 2001] I. Cervesato. The Dolev-Yao intruder is the most powerful attacker. In *16th Annual Symposium on Logic in Computer Science LICS*, volume 1, 2001. Citeseer. 32

[Chen et al. 2007] L. Chen, Z. Cheng, and N. P. Smart. Identity-based key agreement protocols from pairings. *International Journal of Information Security*, 6(4):213–241, 2007. 75

[Cheshire and Krochmal 2013] DNS-based service discovery. Technical report, 2013. 22

[Cirani et al. 2015] S. Cirani, M. Picone, P. Gonizzi, L. Veltri, and G. Ferrari. Iot-oas: an oauth-based authorization service architecture for secure services in iot scenarios. *IEEE Sensors Journal*, 15(2):1224–1234, 2015. 41

[Community 2018] C. Community. C.h.i.p community. 2018. 59

[Cox 2012] C. Cox. *An introduction to LTE: LTE, LTE-advanced, SAE and 4G mobile communications*. John Wiley & Sons, 2012. 5

[De Silva et al. 2012] L. C. De Silva, C. Morikawa, and I. M. Petra. State of the art of smart homes. *Engineering Applications of Artificial Intelligence*, 25(7):1313–1321, 2012. 6

[Delsarte 1978] P. Delsarte. Bilinear forms over a finite field, with applications to coding theory. *Journal of Combinatorial Theory, Series A*, 25(3):226–241, 1978. 67

[D.Evans 2011] D.Evans. "the internet of things":how the next evolution of the internet is changing everything. *Whitepaper, Cisco Internet Business Solutions Group (IBSG)*, 2011. 21

[Diffie and Hellman 1976] W. Diffie and M. Hellman. New directions in cryptography. *IEEE transactions on Information Theory*, 22(6):644–654, 1976. 65

[Do et al. 2015] Q. Do, B. Martini, and K.-K. R. Choo. Exfiltrating data from android devices. *Computers & Security*, 48:74–91, 2015. 79, 92

[D'Orazio et al. 2017] C. J. D'Orazio, K.-K. R. Choo, and L. T. Yang. Data exfiltration from internet of things devices: ios devices as case studies. *IEEE Internet of Things Journal*, 4(2):524–535, 2017. 79, 92

[Eastlake 3rd 2013] Domain name system (dns) iana considerations. Technical report, 2013. 46, 53

[Eclipse 2014] Eclipse. eclipse/paho.mqtt.python. 2014. 35

[Ellison and Schneier 2000] C. Ellison and B. Schneier. Ten risks of pki: What you're not being told about public key infrastructure. *Comput Secur J*, 16(1):1–7, 2000. 69, 73

[Eloff et al. 2009] J. Eloff, M. Eloff, M. Dlamini, and M. Zielinski. Internet of people, things and services-the convergence of security, trust and privacy. 2009. 1

[Enck et al. 2014] W. Enck, P. Gilbert, S. Han, V. Tendulkar, B.-G. Chun, L. P. Cox, J. Jung, P. McDaniel, and A. N. Sheth. Taintdroid: an information-flow tracking system for realtime privacy monitoring on smartphones. *ACM Transactions on Computer Systems (TOCS)*, 32(2):5, 2014. 79, 92

[EnOcean 2016a] EnOcean. File type specification. 2016. 3

[EnOcean 2016b] EnOcean. System specification. 2016. 3

[Erdtman 2016] Internet Engineering Task Force. Certificate credentials for ACE framework. Internet-Draft draft-erdtman-ace-certificate-credential-00, April 2016. Work in Progress. 40

[Espressif 2014] Espressif. Espressif systems socs. 2014. 75

[E.U 2018] E.U. General data protection regulation. 2018. 9

[Eugster et al. 2003] P. T. Eugster, P. A. Felber, R. Guerraoui, and A.-M. Kermarrec. The many faces of publish/subscribe. *ACM computing surveys (CSUR)*, 35(2):114–131, 2003. 25, 46

[Evans 2011] D. Evans. The internet of things: How the next evolution of the internet is changing everything. *CISCO white paper*, 1(2011):1–11, 2011. 6

[Flores et al. 2013] A. B. Flores, R. E. Guerra, E. W. Knightly, P. Ecclesine, and S. Pandey. Ieee 802.11 af: A standard for tv white space spectrum sharing. *IEEE Communications Magazine*, 51(10):92–100, 2013. 4

[Fremantle et al. 2014] P. Fremantle, B. Aziz, J. Kopeckỳ, and P. Scott. Federated identity and access management for the internet of things. In *Secure Internet of Things (SIoT), 2014 International Workshop on*, pages 10–17, 2014. IEEE. 13

[Fried 2013] L. Fried. A bill of rights for the internet of things. 2013. 17

[Fuchs et al. 2009] Scandroid: Automated security certification of android. Technical report, 2009. 79

[G. Montenegro 2007] J. H. D. C. ,N. KushalnagarG. Montenegro. Transmission of ipv6 packets over ieee 802.15.4 networks. 2007. 4

[Gaedke et al. 2005] M. Gaedke, J. Meinecke, and M. Nussbaumer. A modeling approach to federated identity and access management. In *Special interest tracks and posters of the 14th international conference on World Wide Web*, pages 1156–1157, 2005. ACM. 13

[Galeev 2006] I. Galeev. Catching the z-wave. October 2006. 2

[Gantz and Reinsel 2012] J. Gantz and D. Reinsel. The digital universe in 2020: Big data, bigger digital shadows, and biggest growth in the far east. *IDC iView: IDC Analyze the future*, 2007(2012):1–16, 2012. 6

[Gartner 2017] Gartner. 8.4 billion connected "things" will be in use in 2017, up 31 percent from 2016. 2017. iv, vi, 6, 102

[George 2014] D. George. Micropython. 2014. 75

[Gerdes et al. 2014] S. Gerdes, O. Bergmann, and C. Bormann. Delegated CoAP Authentication and Authorization Framework (DCAF). 2014. 40

[Gerdes et al. 2016] Internet Engineering Task Force. An architecture for authorization in constrained environments. Internet-Draft draft-ietf-ace-actors-03, March 2016. Work in Progress. 22

[Gomez et al. 2012] C. Gomez, J. Oller, and J. Paradells. Overview and evaluation of bluetooth low energy: An emerging low-power wireless technology. *Sensors*, 12(9):11734–11753, 2012. 3

[Google 2008] Google. Google play store. 2008. 97

[Google 2018] Use network service discovery. Technical report, 2018. 45

[Graa et al. 2014] M. Graa, N. C. Boulahia, F. Cuppens, and A. Cavalliy. Protection against code obfuscation attacks based on control dependencies in android systems. In *Software Security and Reliability-Companion (SERE-C), 2014 IEEE Eighth International Conference on*, pages 149–157, 2014. IEEE. 79

[Graa et al. 2012] M. Graa, N. Cuppens-Boulahia, F. Cuppens, and A. Cavalli. Detecting control flow in smarphones: Combining static and dynamic analyses. In *Cyberspace Safety and Security*, pages 33–47. Springer, 2012. 92

[Graa et al. 2016] M. Graa, N. Cuppens-Boulahia, F. Cuppens, and J.-L. Lanet. Tracking explicit and control flows in Java and native Android apps code. In *ICISSP 2016 : 2nd International Conference on Information Systems Security and Privacy*, volume Proceedings of the 2nd International Conference on Information Systems Security and Privacy, pages 307–316, 2016. 79, 92

[Gubbi et al. 2013a] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami. Internet of things (iot): A vision, architectural elements, and future directions. *Future Generation Computer Systems*, 29(7):1645 – 1660, 2013. Including Special sections: Cyber-enabled Distributed Computing for Ubiquitous Cloud and Network Services & Cloud Computing and Scientific Applications ,Äî Big Data, Scalable Analytics, and Beyond. 21

[Gubbi et al. 2013b] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami. Internet of things (iot): A vision, architectural elements, and future directions. *Future generation computer systems*, 29(7):1645–1660, 2013. 79

[Gupta and Murty 1986] R. Gupta and M. R. Murty. Primitive points on elliptic curves. *Compositio mathematica*, 58(1):13–44, 1986. 66

[Gura et al. 2004] N. Gura, A. Patel, A. Wander, H. Eberle, and S. C. Shantz. Comparing elliptic curve cryptography and rsa on 8-bit cpus. In *International workshop on cryptographic hardware and embedded systems*, pages 119–132, 2004. Springer. 65

[Guri et al. 2016] M. Guri, M. Monitz, and Y. Elovici. Usbee: air-gap covert-channel via electromagnetic emission from usb. In *Privacy, Security and Trust (PST), 2016 14th Annual Conference on*, pages 264–268, 2016. IEEE. 92

[Hardt 2012] D. Hardt. The oauth 2.0 authorization framework. 2012. 10, 22, 51

[Hartke 2015] Observing resources in the constrained application protocol (coap). Technical report, 2015. 44, 49

[Hornyack et al. 2011] P. Hornyack, S. Han, J. Jung, S. Schechter, and D. Wetherall. These aren't the droids you're looking for: Retrofitting android to protect data from imperious applications. In *Proceedings of the 18th ACM conference on Computer and communications security*, pages 639–652, 2011. ACM. 92

[Housley et al. 1998] Internet x. 509 public key infrastructure certificate and crl profile. Technical report, 1998. 68

[Hsu and Lin 2016] C.-L. Hsu and J. C.-C. Lin. An empirical examination of consumer adoption of internet of things services: Network externalities and concern for information privacy perspectives. *Computers in Human Behavior*, 62:516–527, 2016. 79

[Hunt 2016] S. T. E. Hunt. Cyber attack: hackers 'weaponised' everyday devices with malware. *The Guardian*, October 2016. 21, 102

[Hwang et al. 2015] S. Hwang, S. Lee, Y. Kim, and S. Ryu. Bittersweet adb: Attacks and defenses. In *Proceedings of the 10th ACM Symposium on Information, Computer and Communications Security*, pages 579–584, 2015. ACM. 92

[ITU 2005] ITU. Itu strategy and policy unit (spu),the internet of things executive summary. 2005. 1

[Jones 2015] Json web algorithms (jwa). Technical report, 2015. 81

[Jones et al. 2015a] Json web signature (jws). Technical report, 2015. 81

[Jones and Hildebrand 2015] Json web encryption (jwe). Technical report, 2015. 81

[Jones et al. 2015b] M. Jones, E. Wahlstroem, S. Erdtman, and H. Tschofenig. JSON Web Token (JWT). RFC 7519, May 2015. 80, 81, 85

[Jones et al. 2018] M. Jones, E. Wahlstroem, S. Erdtman, and H. Tschofenig. CBOR Web Token (CWT). RFC 8392, 2018. 61

[Joye and Neven 2009] I.-B. C. M. Joye and G. Neven. Identity-based signatures. *Identity-Based Cryptography*, 2:31, 2009. 75

[Kelley 2013] M. B. Kelley. The stuxnet attack on iran,Äôs nuclear plant was ,Äòfar more dangerous,Äô than previously thought. *Business Insider*, 20, 2013. 69, 76

[Khodadadi et al. 2015] F. Khodadadi, A. V. Dastjerdi, and R. Buyya. Simurgh: A framework for effective discovery, programming, and integration of services exposed in IoT. In *Recent Advances in Internet of Things (RIoT), 2015 International Conference on*, pages 1–6, 2015. IEEE. 24

[Khorov et al. 2015] E. Khorov, A. Lyakhov, A. Krotov, and A. Guschin. A survey on ieee 802.11 ah: An enabling networking technology for smart cities. *Computer Communications*, 58:53–69, 2015. 4

[Kobeissi et al. 2017] N. Kobeissi, K. Bhargavan, and B. Blanchet. Automated verification for secure messaging protocols and their implementations: A symbolic and computational approach. In *2nd IEEE European Symposium on Security and Privacy (EuroS&P'17)*, pages 435–450, Paris, France, April 2017. IEEE. 29, 87

[Koblitz 2012] N. I. Koblitz. *Introduction to elliptic curves and modular forms*, volume 97. Springer Science & Business Media, 2012. 66

[Kothmayr et al. 2013] T. Kothmayr, C. Schmitt, W. Hu, M. Brunig, and G. Carle. DTLS based security and two-way authentication for the Internet of Things. *Ad Hoc Networks*, 11(8):2710–2723, 2013. 40

[Krawczyk 2005] H. Krawczyk. Hmqv: A high-performance secure diffie-hellman protocol. In *Annual International Cryptology Conference*, pages 546–566, 2005. Springer. 75

[Lamparter et al. 1992] B. Lamparter, W. Effelsberg, and N. Michl. Mtp: a movie transmission protocol for multimedia applications. *ACM SIGCOMM Computer Communication Review*, 22(3):71–72, 1992. 81

[Langner 2011] R. Langner. Stuxnet: Dissecting a cyberwarfare weapon. *IEEE Security & Privacy*, 9(3):49–51, 2011. 69

[Lau et al. 2013] B. Lau, Y. Jang, C. Song, T. Wang, P. H. Chung, and P. Royal. Mactans: Injecting malware into ios devices via malicious chargers. *Black Hat USA*, 2013. 92

[Lenstra et al. 1993] A. K. Lenstra, H. W. Lenstra, M. S. Manasse, and J. M. Pollard. The number field sieve. In *The development of the number field sieve*, pages 11–42. Springer, 1993. 67

[Li and Liang 2013] Z. Li and Q. Liang. Performance analysis of multiuser selection scheme in dynamic home area networks for smart grid communications. *IEEE Transactions on Smart Grid*, 4(1):13–20, 2013. 7

[Lindsay 2011] J. Lindsay. Pyjwt. 2011. 86

[Liu 2014] F. Liu. Windows malware attempts to infect android devices. 2014. 92

[Lueg 2013a] L. Lueg. Pyrit code source. 2013. 73

[Lueg 2013b] L. Lueg. The twilight of wi-fi protected access. 2013. 73

[Lugo-Cordero et al. 2014] H. M. Lugo-Cordero, R. K. Guha, and E. I. Ortiz-Rivera. An adaptive cognition system for smart grids with context awareness and fault tolerance. *IEEE Transactions on Smart Grid*, 5(3):1246–1253, 2014. 7

[Machani et al. 2014] S. Machani et al. Fido uaf review draft spec set. *FIDO Alliance Proposed Standard*, pages 1–202, 2014. 14

[Maciej Machulak 2018a] J. R. B. E. ,HSBCMaciej Machulak. Federated authorization for user-managed access (uma) 2.0. 2018. 11

[Maciej Machulak 2018b] J. R. B. E. ,HSBCMaciej Machulak. User-managed access (uma) 2.0 grant for oauth 2.0 authorization. 2018. 11

[Maler 2019] E. Maler. Uma implementations page. 2019. 11

[MalwareMustDie 2016] MalwareMustDie. Mirai (malware). 2016. 21, 102

[Manyika et al. 2013] J. Manyika, M. Chui, J. Bughin, R. Dobbs, P. Bisson, and A. Marrs. *Disruptive technologies: Advances that will transform life, business, and the global economy*, volume 180. McKinsey Global Institute San Francisco, CA, 2013. 6

[Matsumoto et al. 1986] T. Matsumoto, Y. Takashima, and H. Imai. On seeking smart public-key-distribution systems. *IEICE TRANSACTIONS (1976-1990)*, 69(2):99–106, 1986. 75

[Mendes et al. 2015] T. Mendes, R. Godina, E. Rodrigues, J. Matias, and J. Catalão. Smart home communication technologies and applications: Wireless protocol assessment for home area network resources. *Energies*, 8(7):7279–7311, 2015. 7

[Microsoft 2004] Microsoft. Microsoft app. store. 2004. 97

[Migault et al. 2016] D. Migault, T. Guggemos, and C. Bormann. Esp header compression and diet-esp. *Internet Engineering Task Force, Internet-Draft draft-mglt-ipsecme-diet-esp-03*, 2016. 76

[Migault et al. 2017] D. Migault, T. Guggemos, S. Killian, M. Laurent, G. Pujolle, and J. P. Wary. Diet-esp: Ip layer security for iot. *Journal of Computer Security*, 25(2):173–203, 2017. 76

[Miller 1985] V. S. Miller. Use of elliptic curves in cryptography. In *Conference on the theory and application of cryptographic techniques*, pages 417–426, 1985. Springer. 66

[Muresan 2016] R. Muresan. Attacks on IoT devices more than doubled in 2015, study shows ‚Äì HOTforSecurity. 2016. 21, 102

[Murphy 2013] B. F. Murphy. Network penetration testing and research. 2013. 73

[N. Kushalnagar 2007] C. S. ,G. MontenegroN. Kushalnagar. Ipv6 over low-power wireless personal area networks (6lowpans): Overview, assumptions, problem statement, and goals. 2007. 4

[Nicanfar et al. 2014] H. Nicanfar, P. Jokar, K. Beznosov, and V. C. Leung. Efficient authentication and key management mechanisms for smart grid communications. *IEEE systems journal*, 8(2):629–640, 2014. 75

[Nicolas 2012] S. Nicolas. Power profiling: Https long polling vs. mqtt with ssl, on android. 2012. 25, 46

[Nordrum 2016] A. Nordrum. Popular Internet of Things Forecast of 50 Billion Devices by 2020 Is Outdated. 2016. 79

[Nyberg and Rueppel 1994] K. Nyberg and R. A. Rueppel. Message recovery for signature schemes based on the discrete logarithm problem. In *Workshop on the Theory and Application of of Cryptographic Techniques*, pages 182–193, 1994. Springer. 65

[O'Leary 2009] N. O'Leary. Arduino Client for MQTT. 2009. 35

[Palmila 2007] P. Palmila. Zeroconf and UPnP techniques. *Helsinki University of Technology, Tech. Rep*, 2007. 45

[Palombini 2018] Internet Engineering Task Force. CoAP Pub-Sub Profile for Authentication and Authorization for Constrained Environments (ACE). Internet-Draft draft-palombini-ace-coap-pubsub-profile-03, June 2018. Work in Progress. 50

[Pedrasa et al. 2010] M. A. A. Pedrasa, T. D. Spooner, and I. F. MacGill. Coordinated scheduling of residential distributed energy resources to optimize smart home energy services. *IEEE Transactions on Smart Grid*, 1(2):134–143, 2010. 6

[Pendragon 1997] Pendragon. the caffeinemark java performance test. 1997. 91

[Raza et al. 2013] S. Raza, H. Shafagh, K. Hewage, R. Hummen, and T. Voigt. Lithe: Lightweight secure CoAP for the internet of things. *IEEE Sensors Journal*, 13(10):3711–3720, 2013. 40, 61

[Raza et al. 2012a] S. Raza, D. Trabalza, and T. Voigt. 6lowpan compressed DTLS for CoAP. In *2012 IEEE 8th International Conference on Distributed Computing in Sensor Systems*, pages 287–289, 2012. IEEE. 4, 40

[Raza et al. 2012b] S. Raza, D. Trabalza, and T. Voigt. 6lowpan compressed DTLS for CoAP. In *2012 IEEE 8th International Conference on Distributed Computing in Sensor Systems*, pages 287–289, 2012. IEEE. 76

[Sakimura et al. 2014] N. Sakimura, J. Bradley, M. Jones, de B. Medeiros, and C. Mortimore. Openid connect core 1.0 incorporating errata set 1. *The OpenID Foundation, specification*, 2014. 70

[Sampath Srinivas 2014] E. T. ,Dirk BalfanzSampath Srinivas. Universal 2nd factor (u2f) overview. 2014. 14

[Schaad 2003] J. Schaad. Use of the advanced encryption standard (aes) encryption algorithm in cryptographic message syntax (cms). 2003. 8

[Schaad 2017] J. Schaad. CBOR Object Signing and Encryption (COSE). RFC 8152, July 2017. 61

[Scott 2002] M. Scott. Authenticated id-based key exchange and remote log-in with simple token and pin number. *IACR Cryptology ePrint Archive*, 2002:164, 2002. 75

[Sharma et al. 2013] P. Sharma, A. Joshi, and T. Finin. Detecting data exfiltration by integrating information across layers. In *Information Reuse and Integration (IRI), 2013 IEEE 14th International Conference on*, pages 309–316, 2013. IEEE. 92

[Shelby 2012] Constrained restful environments (core) link format. Technical report, 2012. 53

[Shelby et al. 2014] Z. Shelby, K. Hartke, and C. Bormann. The Constrained Application Protocol (CoAP). 2014. 44, 48, 76, 106

[Shim 2003] K. Shim. Efficient id-based authenticated key agreement protocol based on weil pairing. *Electronics Letters*, 39(8):653–654, 2003. 75

[Sigfox 2012] Sigfox. Sigfox : Radio technology keypoints. 2012. 5

[Smart 1999] N. P. Smart. The discrete logarithm problem on elliptic curves of trace one. *Journal of cryptology*, 12(3):193–196, 1999. 65

[Smart 2002] N. P. Smart. Identity-based authenticated key agreement protocol based on weil pairing. *Electronics letters*, 38(13):630–632, 2002. 75

[Sourceforge 1999] Sourceforge. Microsoft app. store. 1999. 97

[Spolaor et al. 2017] R. Spolaor, L. Abudahi, V. Moonsamy, M. Conti, and R. Poovendran. No free charge theorem: A covert channel via usb charging cable on mobile devices. In *International Conference on Applied Cryptography and Network Security*, pages 83–102, 2017. Springer. 92

[Taylor 2013] S. Taylor. The next generation of the internet revolutionizing the way we work, live, play, and learn. *CISCO, San Francisco, CA, USA, CISCO Point of View*, 12, 2013. 6

[Thangavel et al. 2014] D. Thangavel, X. Ma, A. Valera, H.-X. Tan, and C. K.-Y. Tan. Performance evaluation of mqtt and coap via a common middleware. In *Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP), 2014 IEEE Ninth International Conference on*, pages 1–6, 2014. IEEE. 44

[thenextweb 2013] thenextweb. Fitbit users are unwittingly sharing details of their sex lives with the world. 2013. 22, 102

[Things 2016] N. Things. The first $9 computer. 2016. 58

[Thuresson 2006] M. Thuresson. Z-wave, zigbee compete to become standard. *NIKKEI ELECTRONICS ASIA*, page 32, 2006. 2

[Tian et al. 2015] D. J. Tian, A. Bates, and K. Butler. Defending against malicious usb firmware with goodusb. In *Proceedings of the 31st Annual Computer Security Applications Conference*, pages 261–270, 2015. ACM. 92

[Torres et al. 2013] J. Torres, M. Nogueira, and G. Pujolle. A survey on identity management for the future network. *IEEE Communications Surveys & Tutorials*, 15(2):787–802, 2013. 95

[Tschofenig 2014] H. Tschofenig. The OAuth 2.0 Internet of Things (IoT) Client Credentials Grant. 2014. 40

[Urien 2016] P. Urien. Three innovative directions based on secure elements for trusted and secured iot platforms. In *New Technologies, Mobility and Security (NTMS), 2016 8th IFIP International Conference on*, pages 1–2, 2016. IEEE. 97

[Velvindron and Baushke 2017] L. Velvindron and M. Baushke. Increase the secure shell minimum recommended diffie-hellman modulus size to 2048 bits. 2017. 65

[Viehböck 2011] S. Viehböck. Brute forcing wi-fi protected setup. when poor design meets poor implementation. 2011. 73

[Wang et al. 2008] S. Wang, Z. Cao, M. A. Strangio, and L. Wang. Cryptanalysis and improvement of an elliptic curve diffie-hellman key agreement protocol. *IEEE Communications Letters*, 12(2), 2008. 75

[Wang et al. 2012] Z. Wang, R. Johnson, R. Murmuria, and A. Stavrou. Exposing security risks for commercial mobile devices. In *International Conference on Mathematical Methods, Models, and Architectures for Computer Network Security*, pages 3–21, 2012. Springer. 92

[Yang 2013] H. Yang. Flask-OAuthlib ,Äî Flask-OAuthlib 0.9.3 documentation. 2013. 35

[Zhang et al. 2013] D. Zhang, N. Shah, and L. G. Papageorgiou. Efficient energy consumption and operation management in a smart building with microgrid. *Energy Conversion and Management*, 74:209–222, 2013. 6

[Zhang et al. 2011] Y. Zhang, L. Wang, W. Sun, R. C. Green II, and M. Alam. Distributed intrusion detection system in a multi-layer network architecture of smart grids. *IEEE Trans. Smart Grid*, 2(4):796–808, 2011. 7

List of Figures

UNIVERSITE BRETAGNE LOIRE / MATHSTIC

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

Titre : Identité et Consentement dans l'Internet des Personnes, des Objets et des Services

Mots clés : IoT, IoPTS, Identité, Consentement, Securité

Résumé : La course à la miniaturisation des appareils informatiques est en train de transformer notre relation avec ces derniers, ainsi que leurs rôles dans notre société. Le nombre d'ordinateurs miniatures contrôlés à distance augmente considérablement et ces objets connectés - comme ils sont communément appelés - sont de plus en plus sollicités pour effectuer des tâches à la place de l'Homme. La tendance actuelle consiste à créer une place dans Internet pour ces objets connectés, autrement dit, à construire des protocoles adaptés à leurs ressources limitées. Cette tendance est connue comme l'Internet des Objets - ou l'acronyme anglais IoT - qui est différent des protocoles destinés à une utilisation exclusivement par des humains dit Internet des Personnes ou IoP en anglais. Avec l'adoption de cette séparation conceptuelle, comment est-ce qu'une personne échangerait ses informations avec des objets sans sacrifier la sécurité ?
Pour aider à réduire cet écart, on a besoin d'un intermédiaire et la mise en réseau de ces intermédiaires amène à construire le concept d'Internet des Services ou IoS en anglais. Les personnes et les objets sont connectés à travers les services. Le réseau dans son ensemble, incluant les personnes, les objets et les services est donc l'Internet des Personnes, des Objets et des Services.
Notre travail se situe à l'intersection de ces trois domaines et notre contribution est double. Premièrement, nous assurons que la liaison entre l'identité d'une personne et de ses objets ne se fasse pas au détriment des propriétés de sécurité telles que l'Intégrité, l'Anonymat et la confidentialité. Et deuxièmement, nous abordons la gestion de la confidentialité des données avec les objets dits connectés.
Dans la quête d'une meilleure intégration des objets connectés à Internet, nous avons contribué à la définition de protocoles autant sur la couche applicative que sur la couche réseau du modèle OSI, avec pour préoccupations principales les contraintes de l'IoT et la sécurité.

Title : Identity and Consent in the Internet of Persons, Things and Services

Keywords : IoT, IoPTS, Identity, Consent, Security

Abstract : The constant efforts of miniaturization of computing machines is transforming our relationships with machines and their role in society. The number of tiny computers remotely controlled is skyrocketing and those connected things are now more and more asked to do things on human behalf. The trend consists in making room for these specific machines into the Internet, in other words, building communication protocols adapted to their limited resources. This trend is commonly known as the Internet of Things (IoT) which consist of appliances and mechanisms different from those meant to be used exclusively by humans, the Internet of Persons (IoP). This conceptual separation being adopted, how would a Person exchange information with Things ?
Sorts of brokers can help bridging that gap. The networking of those brokers led to the concept of Internet of Services (IoS). Persons and Things are connected through Services. This global networking is called the Internet of Persons Things and Services (IoPTS).
Our work is on the edge of these 3 Internet areas and our contributions are twofold. In the first hand, we tackle the secure biding of devices' and persons' identities while preserving the Integrity, Anonymity and Confidentiality security properties. On the other hand, we address the problem of the secrecy of data on constrained Internet-connected devices.
Other mechanisms must be created in order to seamlessly bind these conceptual areas of IoP, IoT and IoS. In this quest for a better integration of Internet connected-devices into the Internet of Persons, our work contributes to the definition of protocols on application and network layers, with IoT concerns and security at heart.