Aalto University
School of Science
Degree Programme in ICT innovation

Relja Paunovic

Contracting Service For Industrial Internet

Re-inventing the Wheel

Master's Thesis
Espoo, June 18, 2011

DRAFT! — August 14, 2017 — DRAFT!

Supervisor: Professor Petri Vuorimaa, Aalto University
Advisor: Instructor?

Aalto University
School of Science

Aalto University
School of Science
Degree Programme in ICT innovation

ABSTRACT OF
MASTER'S THESIS

Author:	Relja Paunovic		
Title:			
Contracting Service For Industrial Internet Re-inventing the Wheel			
Date:	June 18, 2011	**Pages:**	vi + 29
Major:	Hypermedia	**Code:**	T-110
Supervisor:	Professor Petri Vuorimaa		
Advisor:	Instructor?		

A dissertation or thesis is a document submitted in support of candidature for a degree or professional qualification presenting the author's research and findings. In some countries/universities, the word thesis or a cognate is used as part of a bachelor's or master's course, while dissertation is normally applied to a doctorate, whilst, in others, the reverse is true.

!Fixme **Abstract text goes here (and this is an example how to use fixme).** Fixme! Fixme is a command that helps you identify parts of your thesis that still require some work. When compiled in the custom `mydraft` mode, text parts tagged with fixmes are shown in bold and with fixme tags around them. When compiled in normal mode, the fixme-tagged text is shown normally (without special formatting). The draft mode also causes the "Draft" text to appear on the front page, alongside with the document compilation date. The custom `mydraft` mode is selected by the `mydraft` option given for the package `aalto-thesis`, near the top of the `thesis-example.tex` file.

The thesis example file (`thesis-example.tex`), all the chapter content files (`1introduction.tex` and so on), and the Aalto style file (`aalto-thesis.sty`) are commented with explanations on how the Aalto thesis works. The files also contain some examples on how to customize various details of the thesis layout, and of course the example text works as an example in itself. Please read the comments and the example text; that should get you well on your way!

Keywords:	ocean, sea, marine, ocean mammal, marine mammal, whales, cetaceans, dolphins, porpoises
Language:	English

Acknowledgements

I wish to thank all students who use LaTeX for formatting their theses, because theses formatted with LaTeX are just so nice.

Thank you, and keep up the good work!

Espoo, June 18, 2011

Relja Paunovic

Abbreviations and Acronyms

IOT	Internet Of Things
IIOT	Industrial Internet of Things
ETSI	European Telecommunications Standards Institute
M2M	Machine To Machine
GSCM2MTF	Global Standards Collaboration Machine-to-Machine Task Force (GSCM2MTF)
3GPP	3rd Generation Partnership Project
ETSI	European Telecommunications Standards Institute
REST	Representational State Transfer
SCL	RESTful Service Capability Layer
OMA	Open Mobile Alliance
LWM2M	Light Weight Machine-to-Machine
CoAP	Constrained Application Protocol
DoS	Denial of Service
FQDN	Fully Qualified Domain Name
CES	Customer Edge Switching

Contents

Abbreviations and Acronyms **iv**

1 Introduction **1**
1.1 Problem statement 1
1.2 Methodology 1
1.3 Outline 1

2 Literature Review **2**
2.1 Internet of Things 2
2.1.1 Industrial Internet of Things 3
2.1.2 Open Issues 3
2.1.2.1 Standardization 4
2.1.2.2 Security and privacy 5
2.2 Light Weight Machine to Machine (LWM2M) 5
2.3 Policy Based Communications for 5G Mobile with Customer Edge Switching 7

3 Requirements **9**
3.1 Context 9
3.1.1 Smart Traffic 9
3.1.2 Connected Goods 10
3.1.3 Smart Crane 11
3.1.3.1 Maintenance 11
3.1.3.2 Users of the Crane 12
3.1.3.3 KoneCranes 12
3.2 Security 13
3.2.1 Access Control 13
3.2.2 Authentication 13
3.2.3 Authorization? 13
3.3 User Experience – FUNCTIONAL REQUIREMENTS 13

4 Prototype Implementation **14**
4.1 Technologies Used . . . 14
4.1.1 Django Framework . . . 14
4.1.2 Bootstrap . . . 14
4.2 Overview . . . 14
4.2.1 Account Types . . . 15
4.2.2 Device Management . . . 16
4.2.3 Policy Management . . . 17
4.3 Manufacturer . . . 18
4.4 Customer . . . 20
4.5 User . . . 22

5 Evaluation **25**

6 Conclusion **26**

A Appendix A **29**

Chapter 1

Introduction

!Fixme **Contracting Service, IOT, IIOT, hint of use cases(context),...** Fixme!

1.1 Problem statement

1.2 Methodology

1.3 Outline

Chapter 2

Literature Review

!FIXME **connect with introduction, write later** FIXME!

2.1 Internet of Things

Internet of things (IOT) is a relatively new concept gaining momentum since the start of this century. Although, first examples of IOT date back as early as 1983 with an automated inventory system. IOT can be seen as an extension to Internet, with physical devices (such as sensors and actuators) communicating with each other and with humans creating an enormous network.

Nowadays, with the continuous decrease in cost of computational devices we are able to produce powerful devices with communication abilities for a very low price. With the increase in number of devices that needs to be connected (some estimates show up to 75 billion connected devices by year 2025[1]) issues regarding scalability, security, heterogeneity of devices, etc. have emerged.

These issues are holding back the progress of IOT because it forces companies to create their own proprietary systems to fit their needs, which is only an option for large companies with financial capabilities to do so. This leads to large number of systems designed for similar purposes but incompatible with each other (due to use of different protocols or architectures). Temporary solution for this problem is to introduce a middle-ware as proposed by [2] that acts like a bridge between two systems, but this has scalability issues and with the rise of number of connected devices will soon be unacceptable. In order to tackle these issues several standards have been proposed, most

[1] https://www.statista.com/statistics/471264/iot-number-of-connected-devices-worldwide/

notably smartM2M, oneM2M and most recently LightweightM2M (LWM2M) which will be explained in 2.1.2.1. None of these have yet became *de facto* standard.

In the remainder of this section, differences between Industrial Internet of Things (IIOT) and IOT will be introduced, along with a concrete example included in appendix A. Following, open issues of IOT and IIOT will be analyzed in 2.1.2.

2.1.1 Industrial Internet of Things

In IOT, a rough distinction is made between consumer and industrial IOT according to [2]. Consumer IOT applications are aimed to make everyday life easier by saving time and money, such as smart locks, smart homes and wearable hearth monitors. On the other hand, industrial IOT (such as production, automation and intelligent computation systems) focuses on how smart machines, data analytics and networked sensors can improve services in business-to-business domain [11]. As an example, predictive maintenance can generate savings up to 12% over scheduled repairs, leading to a 30% reduction in maintenance cost and a 70% cut in downtime from equipment breakdowns according to Accenture[2] !FIXME **Should I include that research in references? It is not a "standard" research paper, more like slide show** FIXME!.This usually implies extensive machine-to-machine (M2M) communication compared to consumer IOT, where in most cases real time guarantees are not required.

Generally, IIOT have stricter requirements regarding delay, security and general robustness compared to consumer IOT. This is because failures in these devices can have consequences on safety of people and environment. For example, in a factory setting, pressure sensor installed on an indoor crane can cause serious damage by failing to communicate about an obstruction.

2.1.2 Open Issues

!FIXME **You can mention role of midleware here, maybe?** FIXME! As previously mentioned, there are many issues surrounding IOT, most notably, lack of standards and security. These issues will be explained in the remainder of the section.

[2] https://www.accenture.com/us-en/insight-industrial-internet-of-things

2.1.2.1 Standardization

According to Global Standards Collaboration Machine-to-Machine Task Force (GSCM2MTF) there are more than 140 organizations involved in the M2M standardization process worldwide. This is a huge vertical fragmentation of IOT market, and it is a result of long history in Industrial use, starting from seventies with process control systems that continued to be used in todays process automation systems. As already mentioned, these systems are proprietary and are incompatible with each other.

In an attempt to resolve this fragmentation issue three notable initiatives stand out. SmartM2M is an initiative led by European Telecommunications Standards Institute (ETSI), it is based on RESTful Service Capability Layer (SCL) [1] which is available through open interfaces. Resource tree residing on SCL along with procedures for handling them is standardized following Representational State Transfer (REST) principles allowing technology agnostic way of accessing them. SmartM2M also defines security framework including authentication, M2M service bootstrap, key agreement and establishment, and M2M service connection procedures, based on a key hierarchy of the M2M node[6]. Unfortunately, it may have issues with scalability as pointed out by [7], thus, making it unsuitable for IOT and IIOT.

A follow up project was formed in order to resolve issues with SmartM2M, this time with a broader partnership. It is an international project started by seven telecom standards organizations: Association of Radio Industries and Businesses (ARIB) and Telecommunication Technology Committee (TTC), Japan; the Alliance for Telecommunications Industry Solutions (ATIS) and Telecommunications Industry Association (TIA), United States; the China Communications Standards Association (CCSA), China; the European Telecommunications Standards Institute (ETSI), Europe; and the Telecommunications Technology Association (TTA), Korea. This project is based on RESTfull design, same as SmartM2M, resource naming conventions same as SmartM2M and is grounded on horizontal service layer principle. Although, it relaxes the scalability constraints by using hierarchical organization of different actors in the system. With this improvement with respect to SmartM2M, it is a serious contester to became IOT standard, being already adopted by various companies according to [12]. Along with these improvements, it defines a security architecture in three layers: security functions, security environment abstraction and security environment.

Recently, a new standard has emerged from Open Mobile Alliance (OMA) targeting constrained devices named Light Weight Machine-to-Machine (LWM2M). It defines a fast deployable client-server specification while minimizing memory consumption and network overhead making it very appealing to IOT and

IIOT devices. It provides device management and security work-flow for IOT applications in a very light weight manner. Recent results from [13] show that memory footprint overheads on a client side protocol stack are no more than 6-9%. Detailed description of LWM2M will be given in 2.2 since this work is based on it.

2.1.2.2 Security and privacy

Devices in IOT generate, process and exchange vast amount of data that is safety-critical and/or private and they are subject to various attacks. Therefore, it is crucial to assure integrity of the devices code and data from malicious modifications [4]. In IIOT following two requirements are crucial for security according to [15]. Availability is most important requirement, because it can lead to loss in productivity and consequently loss of revenue, this is particularly affected by denial of service (DoS) attacks and preventing any system failure that may result in physical damage or harm to humans, particularly affected by sabotage.

There are many security architectures for embedded IOT devices, although, majority of them are too complex for low-end devices. Solutions for low-end devices usually rely on physical (hardware) isolation of security-critical code and data from other software on same device. Examples of such architectures are SMART [5], SPM [14], SANCTUS [10] and TrustLite [9] but they all have major flaws. SMART does not allow code changes after deployment, SPM and SANCTUS have hardware assisted task isolation but they are non-interruptible which violates real-time guarantees and TrustLite requires all software components to be loaded at boot time which reduces flexibility. TyTAN [3] is the only work that provides secure loading of tasks at runtime, secure inter-process communication, local and remote attestation and real-time guarantees.

2.2 Light Weight Machine to Machine (LWM2M)

OMA has approved first version (V1.0) of LWM2M standard in February 2017, since it is a very recent specification not many implementations exist, although, number has been steadily increasing since then. Most notable implementation by [13] focuses on client side architecture (residing on IOT devices).

LWM2M provides a light and secure communication interface with compact data model to enable device management and service for IOT (and IIOT) devices. It is a client-server architecture named OMA LWM2M Enabler. Enabler consists of LWM2M Server which is a central point that takes care of the devices, assigning security keys, registering device capabilities and more. Other part of Enabler is LWM2M Client which resides on a device and provides necessary information to the Server. Furthermore, it uses Constrained Application Protocol (COAP), instead of HTTP, which has a greatly reduced communication overhead making it ideal for constrained devices (or devices benefiting from efficient communication, as the case in IIOT) along with UDP/SMS transport bindings. Architecture of LWM2M is shown on Figure 2.1.

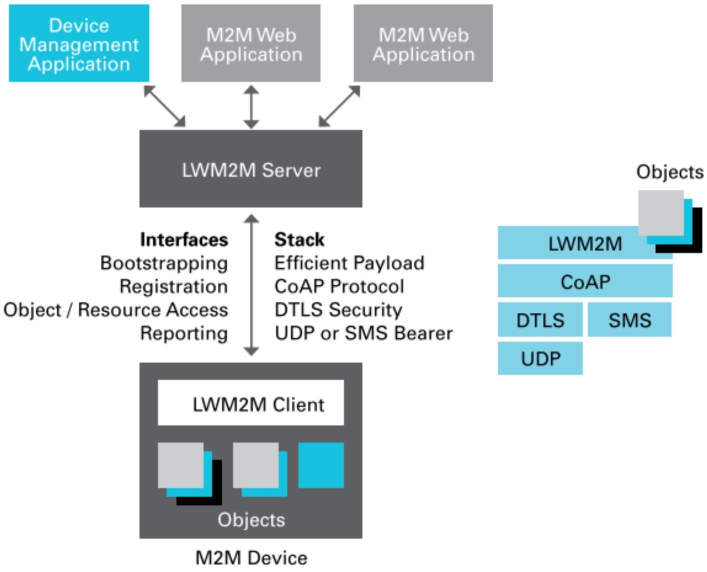


Figure 2.1: LWM2M Architecture

The LWM2M Enabler defines the application layer communication protocol between Server and Client. It is separated into four logical interfaces, namely, Bootstrap, Device Discovery and Registration, Device Management and Service Enablement, and Information Reporting. Diagram showing interfaces and corresponding messages is illustrated in 2.2.

1. Bootstrap: allows LWM2M Bootstrap server to manage keying, access control and to configure device for communication with the Server
2. Device Discovery and Registration: allows Server to discover devices and register their capabilities, e.g. which objects and how to access them does a device have.

3. Device Management and Service Enablement: allows LWM2M Server to manage devices and provide M2M service by sending operations to devices and getting corresponding responses from them
4. Information Reporting: This is a core interface in LWM2M which allows reporting of resource information. It can be triggered periodically, on events or on request

Communication model of LWM2M is based on CoAP methods similar to HTTP verbs GET, POST, PUT, DELETE to manipulate *resources* on devices. Compared to HTTP, CoAP starts with only four bytes of overhead in binary encoded message and is easily translatable to HTTP. Unlike HTTP, CoAP messages are exchanged asynchronously between CoAP end-points over a datagram-oriented transport, in this case UDP.

In LWM2M Enabler, each individually addressable piece of information is called a Resource and groups of Resources are logically organized into Objects. For example, predefined object *Location* contains all resources needed for locating the device. Full list of existing objects can be found at OMA registry [3].

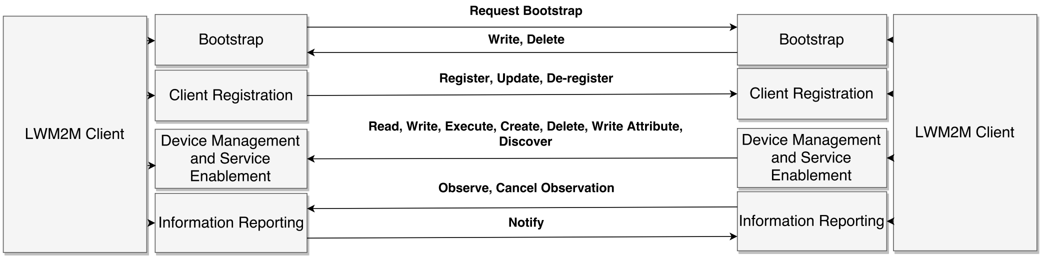

LWM2M Client
Bootstrap
Client Registration
Device Management and Service Enablement
Information Reporting
Request Bootstrap
Write, Delete
Register, Update, De-register
Read, Write, Execute, Create, Delete, Write Attribute, Discover
Observe, Cancel Observation
Notify
Bootstrap
Client Registration
Device Management and Service Enablement
Information Reporting
LWM2M Client

Figure 2.2: LWM2M Interfaces

2.3 Policy Based Communications for 5G Mobile with Customer Edge Switching

Particularly interesting work by [8] proposes a policy based communication with built in security, while also addressing classical weaknesses of Internet, namely, address spoofing, unwanted traffic and DoS attacks. It is based on a principle that before establishing communication between two hosts (or

[3] http://www.openmobilealliance.org/wp/OMNA/LwM2M/LwM2MRegistry.html

networks) they need to negotiate interests, and only if they are matching communication is established. These interests are described with a policy.

This work proposes to replace Network Address Translator (NAT) from the edge of the network with their own Customer Edge Switch (CES) node. This node will act exactly like NAT if the sender, who is behind a CES node, is interacting with a receiver using legacy ip. On the other hand, if the situation is reversed CES node will act as a *realm gateway*. Only if both actors are behind CES node it will act as a cooperative firewall negotiating interests of sender and receiver. Because of this, CES can be applied one network at a time making it suitable for IIOT purposes where vertical fragmentation is a big issue.

Furthermore, CES allows efficient communication by dropping unwanted traffic at the edge and in that way reducing amount of traffic that passes trough network. Also, CES makes use of Domain Name Servers (DNS) to find receivers faster using Fully Qualified Domain Names (FQDN) and MS-ISDN numbers.

In essence, policy database holds all policies and can be accessed through API. Before communication is established, policies of both actors are checked and if they match communication is allowed. Example is shown on 2.3.

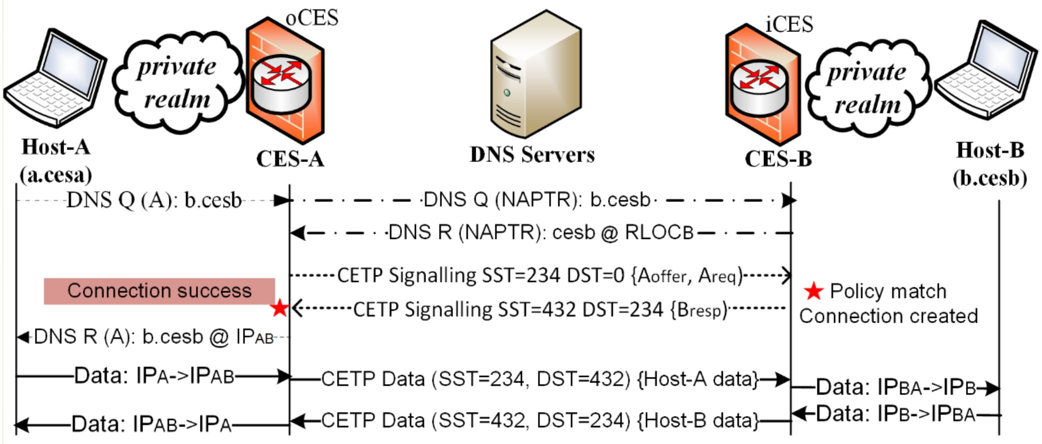


Figure 2.3: CES communication example

Chapter 3

Requirements

To design a solution, environment around the problem needs to be understood, specifically, stakeholders involved and what needs they have. This chapter explains several industrial use cases with aim to familiarize the reader with environment Contracting Service is tailored for, followed by concrete requirements of this solution. Since there are countless number of different use cases, not everything can be covered, especially because new applications are continuously emerging. Therefore, this solution needs to be flexible to cover most of the existing and future use cases.

3.1 Context

This section describes three representative industrial use cases in order to help reader understand the range of applications. All use cases have a common core even though they seem very different from each other as it can be seen from the following text.

3.1.1 Smart Traffic

Recently, smart traffic is emerging as promising trend with the idea to automate transportation of goods and people. Big corporations, such as Google, Tesla, Mercedes and Uber have already joined the race with their models but many technological, legal and business obstacles are holding back its deployment. Integration of these vehicles into regular, non-automated, traffic is a big challenge because it needs to take into account human error and correct it if possible.

One interesting example are smart convoys, driver-less trucks transporting goods in a convoy. For this to be secure, communication between trucks needs to be in real-time so that all, for example, obstructions noticed by sensors on truck in front are conveyed to the rest in time for them to react (slowing down or avoiding obstruction). It is also very important that only authorized people have access to data convoy produces, because if something gets tampered with, human lives are at stake along with structural damages.

For these security reasons not all stakeholders should have access to all the data but only to what is necessary for their operations. Stakeholders involved are: manufacturers; maintenance company; users; and third parties. Firstly, manufacturers should not have access to data that discloses anything that is confidential for truck users, for example, exact location of the convoy in any time because this information can be used to get insight into operations of users. Secondly, manufacturers might not want to give out all information to their users, either because of confidentiality or because it might discredit them. Thirdly, maintenance companies should have only information about the state of the parts of truck. How many times have the breaks been used, level of motor oil, gas and similar which they need in order to schedule maintenance control. Finally, third parties could be any company, or public, that benefits from data these trucks produce and fit into business models of manufacturers or users. For example, traffic control application that provides information about how many convoys are on a particular road so that if the number is too high, traffic jams are expected.

3.1.2 Connected Goods

The servitisation of physical goods will be of strategic importance for the manufacturing industry, where instead of selling parts and machines it will be possible to sell engine hours, kilometers and similar. The vision here is that goods will remember how they were made and produce data throughout whole cycle of their usage, even giving insights into customer satisfaction with those goods. In this case, privacy is a big obstacle, because it is hard to assure customer that data you collected in their home or workspace is not going to be used by anybody they do not want to.

Consider a scenario where all goods in your apartment from carton of milk to air-conditioning are equipped with sensors. Milk carton might posses a heat sensor which alerts when milk is being kept in a warm place for too long, labeling it as spoiled and ordering fresh one from a local marketplace. Same data can be used by for statistical purposes by a third party, to determine, for example, how much milk is being wasted by a nation and use it to adjust size of milk cartons or predict peaks in milk consumption. With some more complex goods, such as air-conditioning, predictive maintenance can be realized with sensors that tell if a particular part is wearing off and alert maintenance service specified by user or manufacturer of that machine. Subset of the data generated by air-conditioning can be used by health organizations to determine whether people are living in unhealthy environments, for example, by checking the ration between how many times air filters have been changed over hours of usage.

Scenarios described in previous paragraph are just few out of many possible ones and it is impossible to tell which ones will be implemented. Although, we have to prepare for the future by designing a flexible system that can withstand rapid changes in the ecosystem.

3.1.3 Smart Crane

KoneCranes have donated a smart crane to Aalto Industrial Campus as a tool for research. Following information was gathered through interviews with researchers and with KoneCranes and it will be used as a representative example for the rest of this work.

This crane has many sensors attached to make it automated and smart as possible. These sensors bring many features, namely **load weighting** (which can also be used to detect whether the crane is stuck somewhere), **remote monitoring** of the position of the crane, **signaling** when some error or warning has occurred, **live video feed** from camera attached above the hook (at the moment, not used for automation but could be supported in the future, for example, image processing to detect obstructions) and more.

There are multiple stakeholders in this ecosystem and all of them require some of the data that Crane generates, although not all information should be available to them, only the bare minimum that is required by their business. These stakeholders are KoneCranes, maintenance service (part of the KoneCranes) and users of the crane. Currently, KoneCranes has access to all the data Crane generates, but in order to increase security, this should not be the case.

3.1.3.1 Maintenance

Maintenance service makes use of usage data (alarms, usage parameters) to monitor and predict maintenance needs of the cranes. They also use the data together with the customer, for example, to review their maintenance spend of the assets, study patterns to reveal relationships between variables and more.

For example, part of the crane that needs to be changed most often are the breaks. Breaks have limited number of uses and this number is approximately known (there is a regulation in place that requires brakes to be changed regularly). Information like, how many times has the Crane been moved from idle state, should be available to maintenance service so they know when do they need to change the breaks (or some other part of the Crane). Crane smart sensors can also detect some irregularities and issue a warning (or error), which is useful information for the maintenance companies.

Information like this should be disclosed to the maintenance service, while restricting the access to, for example, position of the crane in a certain point in time, or any kind of data that can be used to infer the processes inside the factory.

3.1.3.2 Users of the Crane

Inside a factory there are different actors with different privileges. Worker that operates Crane does not need to know the history of the movements of the Crane, or condition of the brakes. Thus, he should only have access to information he needs, e.g. from where to where the load needs to be moved. Manager of those workers on the other hand should be able to monitor all the Cranes in the factory.

In the future, machine to machine communication will be utilized much more, for example, in a setting with two Cranes operating (automated, not by human) in the same room they should be able to signal to each other with their planned path in order to avoid collisions and to optimize the process. In this way, need for a centralized control is eliminated (or at least minimized).

3.1.3.3 KoneCranes

KoneCranes needs access to some of the data that Crane generates, in order to make product improvements, react to possible reliability problems and get better specification for new product generations, adjust warranties, and such. They analyze all types of data that Crane generates: manufacturing data (component lists, manufacturing dates, and more), usage (starts, lifting hours, loads, and more) and sensor data (vibrations, temperature, and more), and maintenance data (maintenance task history, ordered materials, and more).

At the moment KoneCranes has access to all the data Crane generates and their customers are aware of that. For privacy reasons, and in environment with multiple manufacturers, data access should be restricted to only what is needed for a specific role.

3.2 Security

!FIXME **Talk about general security especially of devices** FIXME!

3.2.1 Access Control

3.2.2 Authentication

!FIXME **Dynamic policies should be mentioned - talk about role based access control** FIXME!

3.2.3 Authorization?

3.3 User Experience – FUNCTIONAL REQUIREMENTS

!FIXME **Managing devices for a machine, for a role at same time, Niemen 10?, some stuff like changing user info, checking user info (also for machines and devices),** FIXME!

Chapter 4

Prototype Implementation

For the implementation of contracting service I have used Django framework because it is powerful but also easy to use because it provides a fixed structure to organize the project. Django framework provides built-in modules for most of the common functionalities with a good security architecture. Along with Django, I have used Bootstrap for styling because it is light-weight and simple.

In this chapter I will present a brief description of technologies that I used followed by a brief overview of the whole project, showing how different parts fit together before digging into details.

4.1 Technologies Used

4.1.1 Django Framework

!Fixme **Mention which parts of django do I use, like login(authentication)/logout, admin page, etc** Fixme! !Fixme **Also mention security measures like csrf protection and such** Fixme!

4.1.2 Bootstrap

!Fixme **Mention responsive** Fixme!

4.2 Overview

!Fixme **Mention how it all works (Admin, Manufacturer, Customer, User), CES+LWM2M, add picture from blackboard at end** Fixme!

This solution is a Web platform for managing small devices and controlling who has access to the data they generate. Most challenging part was understanding the users of the platform and their needs. Following section describes account types which correspond to roles they have, followed by the brief description of main parts of the platform.

4.2.1 Account Types

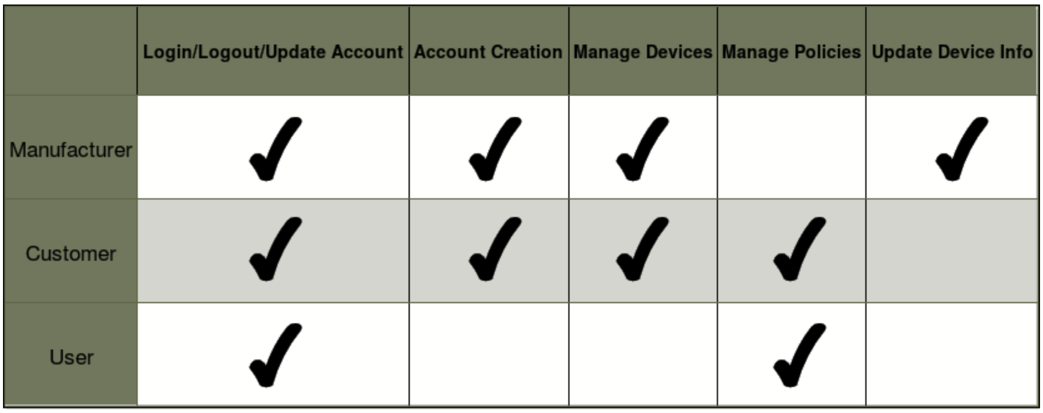
In the Industrial Internet context I have identified four different types of accounts, with their separate views of the platform depending on their role. Views of these accounts have a common core although they are suited for different purposes.

Firstly, there needs to be a central authority that distributes accounts to verified manufacturers of industrial machines. In order to receive manufacturer account you need to contact this central authority, which could be a standardization agency but it can be any impartial actor. When that authority verifies that your company is who they claim to be, several accounts can be issued depending on the number of departments that company has or any other criteria depending on the agreement with the manufacturer company. Since the only responsibility this authority have is creating accounts for manufacturers I have used standard Django admin page (and admin account) for its implementation, stripped down to only basic functionality for managing accounts.

Secondly, manufacturers account is assigned to companies that produce smart machines. Their responsibility is to add devices (and update their information if necessary) to platform and assign them to machines (detailed description of devices will be given in subsection 4.2.2), which are abstract representation of real machine (such is a smart crane) and devices it possess. When the machine has been bought by the customer, manufacturer creates an account for customer (in case they do not possess one) and assigns it to them, giving them full control over who can communicate with that machine. Manufacturer account will be described in detail in section 4.3.

Thirdly, customer account is assigned to owners of the smart machines. Their view is restricted to only devices and machines that they possess and they can manage policies (control access to devices) for those devices only. Along with the possibility of managing policies, customer account can create multiple user accounts (which represents people responsible for single machines or group of them) and assign machines or devices to them. In this way, customer account has a full overview of what policies are in place and who created them while also being able to add or remove faulty ones or general ones (like allowing his work computer to access all the data or opening their data to a statistical agency). By being able to create user accounts customer can !Fixme **pass on** Fixme! the responsibility and fine tuning of policies for certain machines down the hierarchy of the company.

Finally, user account is assigned to people responsible for a subset of machines that a customer company possess. These accounts are restricted to only managing policies of the machines assigned to them without the possibility to create more accounts or manage devices in the system. Overview of accounts and their views are visualized in figure 4.1.



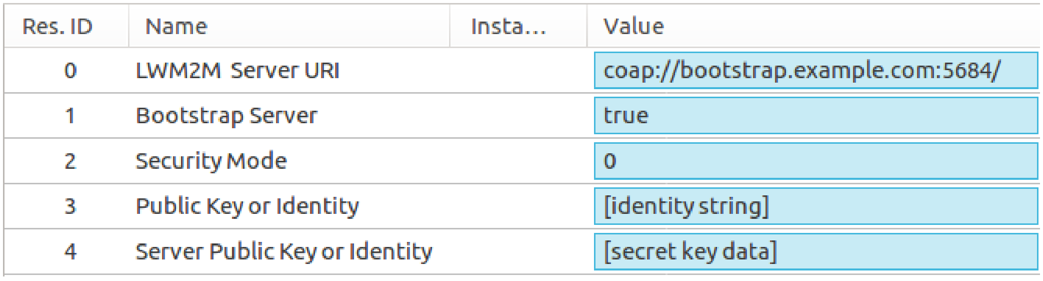
	Login/Logout/Update Account	Account Creation	Manage Devices	Manage Policies	Update Device Info
Manufacturer	✓	✓	✓		✓
Customer	✓	✓	✓	✓	
User	✓			✓	

Figure 4.1: Account types and their views

4.2.2 Device Management

As previously mentioned, manufacturers task is to add devices to the system, filling all necessary information about the device defined by LWM2M standard described in 2.2. This information consists of manufacturer name, model number, serial number, public key or identity of the device along with a descriptive name used to refer to it in the system. Public key or identity according to LWM2M standard could be a certificate, a pre-shared key, raw public key or nothing. Since all these, except last one, are similar and exist in standard only to provide flexibility for the manufacturers only pre-shared key is supported in this solution, although it can easily be extended.

Further following LWM2M standard, which has a client-server architecture, devices act as a client where my solution acts as a bootstrap server. When a device gets connected to a network, it communicates its IP address (in a form of IPv4 or IPv6) and FQDN or MSISDN to a bootstrap server which saves it for use when managing policies. Example of information implanted on a device is shown on figure 4.2.



Res. ID	Name	Insta...	Value
0	LWM2M Server URI		coap://bootstrap.example.com:5684/
1	Bootstrap Server		true
2	Security Mode		0
3	Public Key or Identity		[identity string]
4	Server Public Key or Identity		[secret key data]

Figure 4.2: Example device

Server URI represents the URI of my server which enables a device to locate it. Only after the server has verified identity of the device, using pre-shared key, it can store devices IP address. In an event when a device is moved to another network (changing its IP address), a device communicates a new IP address which replaces the old one and all policies get updated with a new address.

4.2.3 Policy Management

Work described in 2.3 is very extensive and is made with mobile communications in mind. Although, the idea of policy based communications is perfectly suitable for these purposes since allowing communication between a device and some host can be done in one simple HTTP request to Policy Database. Policy Database holds all policies describing who is allowed to communicate with whom. This database could be distributed or in one central place, although that is outside of the scope of this work, but it has one API that hides the way database is arranged and is the only way database can be accessed.

Only four different API requests are used in this work. These requests are for inserting, deleting, retrieving and updating firewall policies using Http Post for inserting and deleting, Http get for retrieving and Http put for updating. Inserting policies require following information and they are sufficient for the CES node to determine whether the communication should be established, should the package be dropped or not at the edge:

1. Target IP address and port - target represents a host wanting to communicate with a device.
2. Source IP address and port - representing a device, extracted from a database in my system.

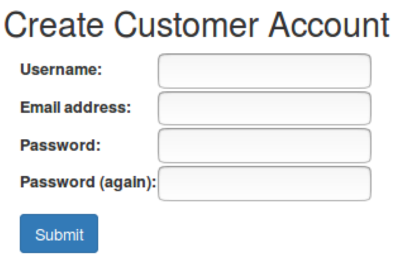
3. Start and end of validity timestamps - allowing scheduling of access to devices.
4. Direction of communication - representing whether only reporting of data from the device to the host is allowed, sending data to a device or both.
5. Transport protocol - in this case CoAP is used, justification is provided in 2.2.
6. FQDN or MSISDN - helps speed up the search for receivers as mentioned in 2.3 and is used for querying policies.

Direction of communication can be bi-directional or uni-directional (from host to device, or device to host). Bi-directional communication is a regular case, when a host sends a request to a device and gets data or acknowledgment in return. Uni-directional communication is useful in cases where the devices just need to send data (for example, temperature readings) to some external host who is subscribed to them, without a possibility of that host controlling the device (for example, sending request for shutdown). Other way around, uni-directional communication from host to a device could be useful in rare cases where a host is allowed to request from a device to do a task, or send data to some other host. Therefore, all three options are supported in this work.

Deleting and retrieving policies is done using only FQDN or MSISDN (depending on which one is available for a device), where updating is done the same way as for inserting only different HTTP verb is used. Graphical interface for managing policies will be described in the remainder of this section where all views from different accounts are explained.

4.3 Manufacturer

When manufacturer creates session by logging-in to the platform he is directed to a home page. Home page for manufacturers contains (along with navigation bar which is always present) only a simple form to create customer account pictured on figure 4.3. This form, along with most of the forms in this solution, is created using Django ModelForm which created a form using database model of a user. When the form is submitted, customer account is made inactive and an automatic email is sent to the provided email address for verification. Automated emails are realized in this prototype using django built-in wrapper for python smtplib module, using a google email account. Only after the customer follows the link in the email, their account



Create Customer Account

Username:

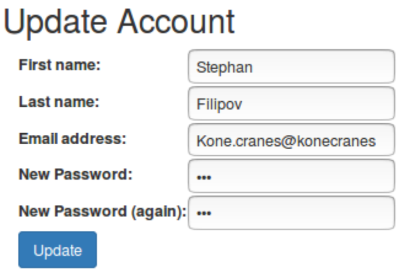
Email address:

Password:

Password (again):

Figure 4.3: Create Customer

is made active and they are prompted to change their password and provide additional information using update form pictured on figure 4.4. Update account form is available to all types of accounts, to encourage password change in order to promote security.



Update Account

First name: Stephan

Last name: Filipov

Email address: Kone.cranes@konecranes

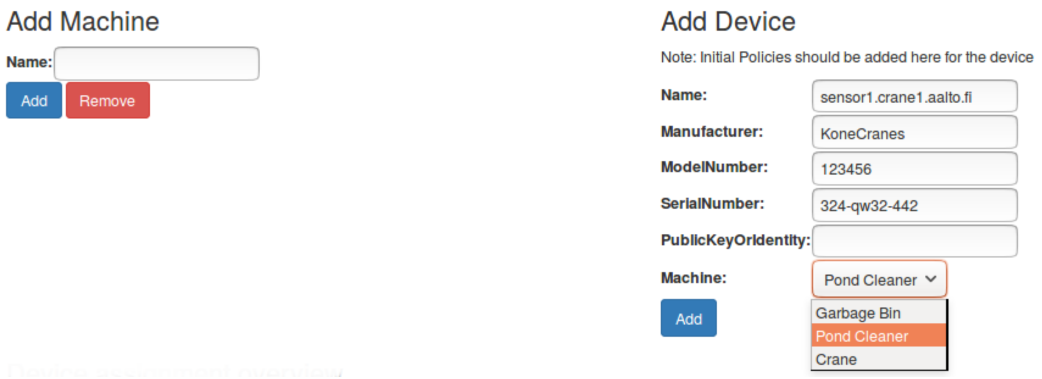
New Password: •••

New Password (again): •••

Figure 4.4: Update Account

As previously mentioned, manufacturers task is to add devices to platform which they can later assign to customers that have bought them. Adding of devices is made available through ”Manage Devices”. For this task, first a machine needs to be created. Machine is an abstract concept that only has a name which is used to organize devices. For example, AaltoCrane could be a machine and all sensors and actuators that are mounted on that crane are assigned to it. In the process of adding a device to the platform, a machine it belongs to needs to be specified. Note that in order to remove a machine from platform, its name needs to be typed in the text box to prevent accidental removal of machines. Two forms needed to complete this task are pictured on figure 4.5.

When the customer has an account and all devices are added to the platform and organized to machines, manufacturer needs to assign these devices to customers. Assigning devices to customers can be done on a device level



Add Machine

Name:

Add Device

Note: Initial Policies should be added here for the device

Name: sensor1.crane1.aalto.fi

Manufacturer: KoneCranes

ModelNumber: 123456

SerialNumber: 324-qw32-442

PublicKeyOrIdentity:

Machine: Pond Cleaner

Figure 4.5: Adding machines and devices

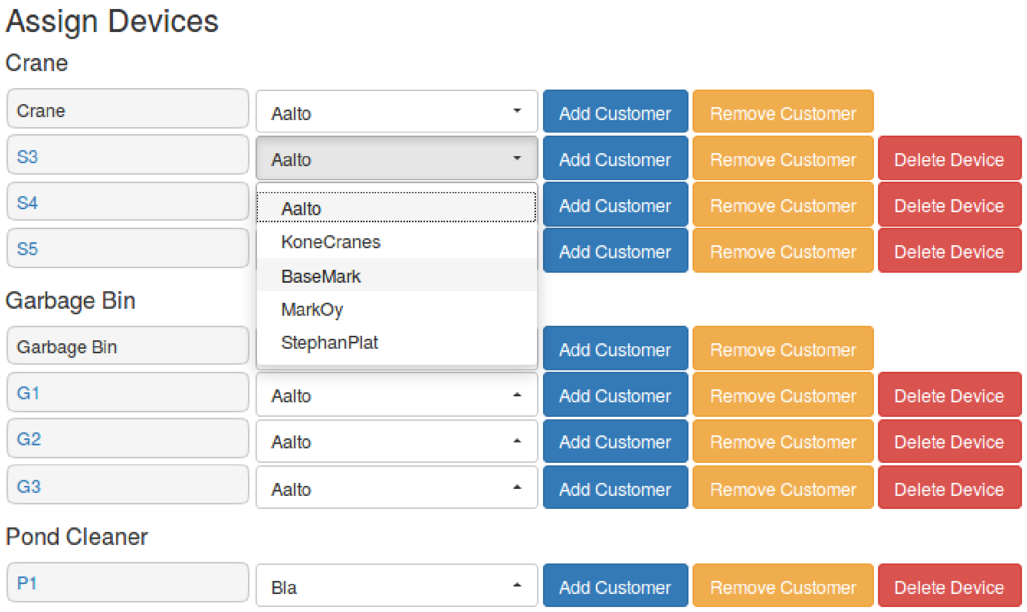
or machine level as pictured on figure 4.6. No matter if the manufacturer is assigning devices or machines (group of devices) the task is done through a simple form and clicking on an appropriate button, either to add or remove a customer. Removing a customer makes sense in case when a machine or device has been rented or a mistake was made while adding. Removing of devices from a platform is also in a same view because it provides a good overview of which devices are attached to which machines.

In the figure 4.6 clicking on a device name navigates to a separate page where details about a device can be inspected, and changed if needed as pictured on figure 4.7. Fields in the form are pre-filled with current information about a device, so that typing mistakes or any small changes can be made easily.

Last element on a Manage Devices page is pictured on figure 4.8. This element only provides an overview of which users are assigned to which devices. It is intended to be used along with element pictured on figure 4.6 in order to check whether any mistakes were made and that everything is how it should be.

4.4 Customer

After logging-in to the platform, customer is directed to a home page where he can create additional user accounts. Creation of user accounts is slightly different from customer accounts as pictured on figure 4.9. In the form, for convenience sake, customer can assign devices to a user while creating the account using a multiple choice select field. In case when the account is created for just one device or a small set of devices, this can greatly reduce



Assign Devices

Crane

Crane — Aalto

S3 — Aalto

S4

S5

Garbage Bin

Garbage Bin

G1 — Aalto

G2 — Aalto

G3 — Aalto

Pond Cleaner

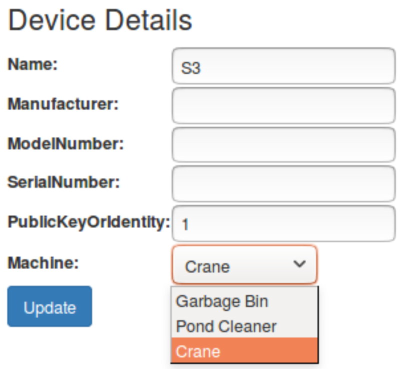
P1 — Bla

Figure 4.6: Device Management For Manufacturer

the number of clicks and queries to the database to complete a task. As with customer accounts, user accounts also need email verification.

On ”Device Management” page, customer can assign devices to users. His view is restricted only to devices that are previously assigned to him by the manufacturer, and choice of users is restricted to the ones he created (which represent users he is responsible for). Interface for doing this resembles one pictured in figure 4.6 with two exceptions: customer is not allowed to change details about devices; and user is not allowed to delete devices from the system. In the case when a machine is no longer used (not needed anymore or replaced with a new one), manufacturer needs to be contacted directly in order to remove the machine from platform. This is because customers can not be allowed to add devices, and if they were allowed to remove devices, they could mistakenly remove the device without the possibility of adding it back. In industrial case, these machines are expensive, large and robust, therefore, they do not need to be removed frequently which is why I opted for this solution.

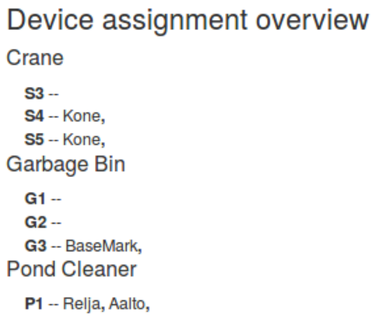
As previously mentioned, only customers and users are allowed to control who can communicate with their devices and they control that through policies. ”Manage Policies” page contains two elements, one for adding policies, other for removing them and to give overview of existing ones. Customer



Device Details

Name:	S3
Manufacturer:	
ModelNumber:	
SerialNumber:	
PublicKeyOrIdentity:	1
Machine:	Crane

Figure 4.7: Update Device Information



Device assignment overview

Crane

- **S3** --
- **S4** -- Kone,
- **S5** -- Kone,

Garbage Bin

- **G1** --
- **G2** --
- **G3** -- BaseMark,

Pond Cleaner

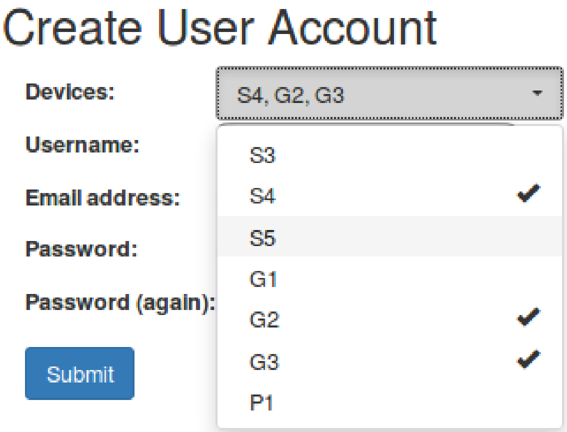
- **P1** -- Relja, Aalto,

Figure 4.8: Device Assignment Overview

view is restricted to only devices that are assigned to him (as in case of managing devices) and he can add policies to Policy database using a group of forms pictured on figure 4.10. Customer only needs to specify IP address (both IPv4 and IPv6 supported) of a host, direction of communication (as explained in 4.2.3), start and end date of policy validity, and click on submit policy. By submitting a policy, customer has allowed a host with specified IP address to interact with a device or all devices that are part of a machine.

Second element on a "Manage Policies" page contains overview of all policies related to devices that a customer has. In the same element, customer can remove policies that are mistakenly made or if the circumstances changed since the policy was put in place.

4.5 User



Create User Account

Devices: S4, G2, G3

Username:

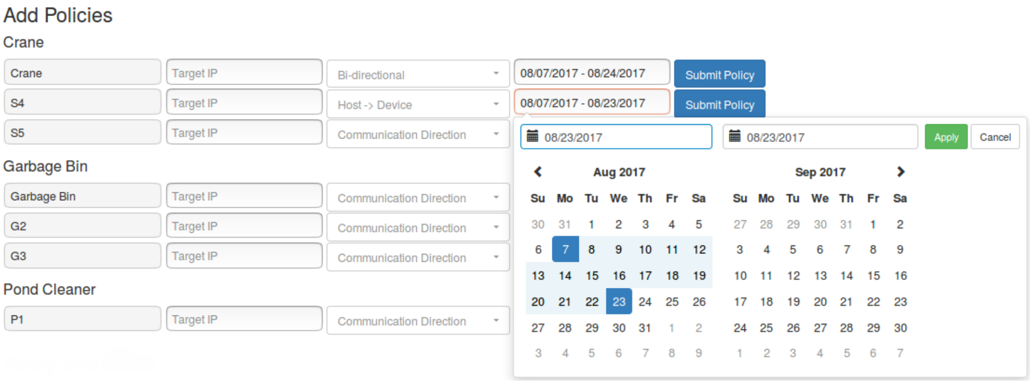
Email address:

Password:

Password (again):

- S3
- S4 ✓
- S5
- G1
- G2 ✓
- G3 ✓
- P1

Figure 4.9: User Account Creation



Add Policies

Crane

Crane	Target IP	Bi-directional	08/07/2017 - 08/24/2017	Submit Policy
S4	Target IP	Host -> Device	08/07/2017 - 08/23/2017	Submit Policy
S5	Target IP	Communication Direction		

Garbage Bin

Garbage Bin	Target IP	Communication Direction
G2	Target IP	Communication Direction
G3	Target IP	Communication Direction

Pond Cleaner

P1	Target IP	Communication Direction

08/23/2017 08/23/2017

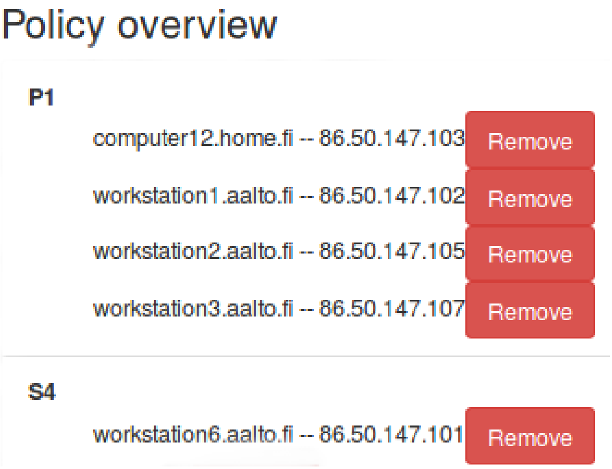
Aug 2017

Su	Mo	Tu	We	Th	Fr	Sa
30	31	1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30	31	1	2
3	4	5	6	7	8	9

Sep 2017

Su	Mo	Tu	We	Th	Fr	Sa
27	28	29	30	31	1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30
1	2	3	4	5	6	7

Figure 4.10: Policy Management



Policy overview

P1

computer12.home.fi -- 86.50.147.103

workstation1.aalto.fi -- 86.50.147.102

workstation2.aalto.fi -- 86.50.147.105

workstation3.aalto.fi -- 86.50.147.107

S4

workstation6.aalto.fi -- 86.50.147.101

Figure 4.11: Overview of policies

Chapter 5

Evaluation

Chapter 6

Conclusion

Bibliography

[1] Alaya, M. B., Banouar, Y., Monteil, T., Chassot, C., and Drira, K. OM2M : Extensible ETSI-compliant M2M service platform with self-configuration capability. *Procedia - Procedia Computer Science 32* (2014), 1079–1086.

[2] Bandyopadhyay, D., and Sen, J. Internet of things: Applications and challenges in technology and standardization. *Wireless Personal Communications 58*, 1 (2011), 49–69.

[3] Brasser, F., Mahjoub, B. E., Sadeghi, A.-r., Wachsmann, C., and Koeberl, P. TyTAN: Tiny Trust Anchor for Tiny Devices.

[4] Control, E., and Security, S. Detecting Industrial Control Malware Using Automated PLC Code Analytics.

[5] Defrawy, K. E., Perito, D., and Tsudik, G. SMART : Secure and Minimal Architecture for (Establishing a Dynamic) Root of Trust.

[6] ETSI. Machine-to-machine communications (m2m); functional architecture, 2013.

[7] Grieco, L. A., Alaya, M. B., Monteil, T., Drira, K., and Bari, P. Architecting Information Centric ETSI-M2M systems. 211–214.

[8] Kantola, R., and Beijar, N. Policy Based Communications for 5G Mobile with Customer Edge Switching.

[9] Koeberl, P., Schulz, S., and Sadeghi, A.-r. TrustLite : A Security Architecture for Tiny Embedded Devices.

[10] Noorman, J., Agten, P., Daniels, W., and Strackx, R. Sancus : Low-cost trustworthy extensible networked devices with a zero-software Trusted Computing Base.

[11] Palattella, M. R., Dohler, M., Grieco, A., Rizzo, G., Torsner, J., Engel, T., and Ladid, L. Internet of Things in the 5G Era: Enablers, Architecture, and Business Models. 510–527.

[12] Park, H., Kim, H., Joo, H., and Song, J. Recent advancements in the Internet-of-Things related standards : A oneM2M perspective. *ICT Express 2*, 3 (2016), 126–129.

[13] Rao, S., Chendanda, D., Deshpande, C., and Lakkundi, V. Implementing LWM2M in Constrained IoT Devices. 52–57.

[14] Raoul Strackx, Frank Piessens, B. P. Efficient isolation of trusted subsystems in embedded systems.

[15] Sadeghi, A., Wachsmann, C., and Waidner, M. Security and privacy challenges in industrial internet of things. *Proceedings of the 52nd* (2015).

Appendix A

Appendix A