MSE | MASTER OF SCIENCE IN ENGINEERING

Hes·so
Haute Ecole Spécialisée de Suisse occidentale
Fachhochschule Westschweiz
University of Applied Sciences and Arts Western Switzerland

Master of Science HES-SO in Engineering

Orientation : Information and Communication Technologies (ICT)

Generic Trust Framework for IoT applications

Realised by

Mohamed Nizar Bouchedakh

Under the supervision of

Prof. Nabil Abdennadher

Prof. Tewfiq El Maliki

At Haute école du paysage, d'ingénierie et d'architecture de Genève

Expert

Mr. Abdenbi Benammour

Solutions architect,
Direction Générale des systèmes d'information

Geneva, HES-SO//Master, 2018

Accepted by HES-SO//Master (Switzerland, Lausanne) on a proposal from

Prof. Nabil Abdennadher
Prof. Tewfiq El Maliki
Advisors

Prof. Nabil Abdennadher
MRU responsible, hepia

Mohamed Nizar Bouchedakh
Geneva, 23 February 2018

Dedication

This work is dedicated to my parents, Younes and Lamia , who have always loved me unconditionally and whose good examples have taught me to work hard for the things that I aspire to achieve.

This work is also dedicated to the people who supported and encouraged me during the challenges of this life.

Table of contents

Acknowledgements

I have taken efforts in this project. However, it would not have been possible without the kind support and help of many individuals and organizations. I would like to extend my sincere thanks to all of them.

I am highly indebted to Pr. Nabil Abdennadher and Pr. Tewfik El Maliki for their guidance and constant supervision as well as for their support in completing this project.

I would like to express my special gratitude and thanks to Mr. Abdenbi Benammour, Dr. Didier Hélal and Dr. Samuel Dubouloz for their support and for giving me such attention and time.

My sincere thanks go to Dr. Royer Philippe and Dr. Paul Royo for their kind cooperation and their support.

My thanks and appreciations also go to my colleagues and people who have willingly helped me out with their abilities.

Abbreviations

6LoWPAN IPv6 Low power Wireless Personal Area Network

A3DB Analyse 3D du Bruit

a.k.a also known as

ABA Anomaly Behavior Analysis

API Application Programming Interface

AWS Amazon Web Services

CAGR Compound Annual Growth Rate

CPU Central Processing Unit

dB Decibels

DDOS Distributed Denial Of Service

DOS Denial Of Service

DWT Discrete Wavelet Transform

ECC Elliptic Curve Cryptography

GPS Global Positioning System

HTTPS HyperText Transfer Protocol Secure

ID Identity

IdM Identity Management

IoT Internet of Things

IP Internet Protocol

IT Information Technology

LTE Long Term Evolution

ML Machine Learning

RRR Reports Reception Ratio

NFC Near Field Communication

PER Packet Error Rate

PVC Privacy Verification Chain

RFID Radio Frequency Identification

RSA Rivest–Shamir–Adleman

SARM Security Adaptation Reference Monitor

SSL Secure Sockets Layer

TCP Transmission Control Protocol

TLS Transport Layer Security

WiFi Wireless Fidelity

WSN Wireless Sensor Network

Abstract

Internet of Things Technology Market is forecast to reach $639.74 billion by 2022 from $176.00 billion in 2016 at a CAGR of 25.1% during 2017-2022. Today, IoT is a very dynamic market with both horizontal and vertical players where applications are diverse and expectations are very high. However, there are some issues that may slow this evolution ; security is one of them.

At its most basic level, security for the Internet of Things depends on the ability to identify devices and protect the data that those devices share. The security aspects related to IoT are tightly linked to the characteristics of the IoT systems: low energy devices, limited computing resources, low cost/quality device, heterogeneity of devices, etc. Conventional IT security policies cannot cope with these challenges and trust techniques are often proposed to overcome these shortcomings.

This work proposes a generic trust framework used by IoT applications to assess trust rates of IoT devices. The developed framework targets non critical IoT applications with no “legal security issues”. The framework assesses the “trustworthiness” of data retrieved from IoT devices, in order to deduce their reliability. The trust score assigned to a data (received from a given IoT device) is used to make necessary “arrangements”: isolate the IoT device, fix the problem, etc.

The proposed generic framework is tested and evaluated in the concrete case of noise devices installed at Carouge (near Geneva). This IoT infrastructure is part of the “Analyse 3D du Bruit” a.k.a A3DB project, jointly conducted by Orbiwise (a Swiss company that develops advanced technologies for the IoT industry), and the “Service de l'air, du bruit et des rayonnements non ionisants” (SABRA). This infrastructure is composed of 1’000 noise sensors. The purpose of the A3DB project is to better understand the sources of noise pollution in Carouge.

Experimental results show that our framework helps detecting misbehaving sensors and assign low trust score to data they are sending.

Keywords IoT, Security, Trust and Reputation systems, Wireless sensors, Noise sensors, Spatial/Temporal correlation.

Introduction

Internet of Things, also referred to as IoT, is an evolution of the internet as we know it nowadays. The dreams of having a self driving car, an intelligent building or a smart city are close to becoming realities thanks to this new way of using the internet. “Things” - e.g. cars, refrigerators and public light bulbs, etc - are getting more and more connected and intelligent. The high potential of IoT and the fast-growing of its market, rushed vendors into producing IoT solutions that meet the users’ dreams. The IoT market is becoming a battlefield. In order to guarantee the biggest market share, vendors are trying to keep a low production cost and a small time to market. The main consequence of this race is the emergence of immature and/or flawed IoT solutions which causes problems at different levels of the IoT ecosystem. Security is one of these levels.

Without secure and reliable solutions, IoT can never reach its full potential. Not a day goes by without hearing about a successful cyber attack targeting Things like IP cameras or Smart TVs. A significant number of IoT solutions handle sensitive and private data. Consequences of security breaches can be catastrophic, especially when dealing with actuators that can cause physical damage. Researchers are working on mechanisms to secure IoT. One of these mechanisms is “Trust and reputation” which is used mostly in the context of non-critical applications. Unlike critical IoT applications, non-critical applications do not have legal obligations and, therefore, can tolerate a “margin of error or malfunction”.

Trust and reputation mechanisms are originally used in large distributed systems with communicating “entities”. Trust and reputation are two coexisting concepts. The later being a result of the first, we often ignore reputation and just talk about trust. Trust mechanisms were first used for classical IT applications to help users or applications detect malicious or misbehaving entities - like fraudulent sellers in e-commerce platforms or bad nodes in grid computing networks. Also, when the processing resources are limited and can not support security overhead, trust can be an alternative. All these factors make trust a promising solution to reinforce security of IoT systems.

This work aims to design a generic trust platform for IoT systems. A generic platform is a platform that can be used by a large set of IoT applications. In this context, trust is used to assess the trustworthiness and the reliability of most kinds of IoT devices.

This document is structured as follows: The first chapter presents IoT systems’ architecture and special features. It gives a glance into the state of the art of security solutions proposed for IoT. The second chapter introduces the trust/reputation concepts. It presents a state of the art of trust and reputation models. The third chapter presents our generic trust and reputation framework. The fourth and final chapter presents an implementation of our trust and reputation model on a large IoT infrastructure of noise sensors deployed on a city scale. Afterwards, this fourth chapter presents results of simulations conducted on simulated data to validate components of our model, and it also presents results of tests conducted on real sensors to validate the overall solution. Finally, results and potential improvements are discussed.

Chapter I Security in IoT

I.1 Introduction

Current conventional IT security policies are often difficult to understand. This issue becomes even more complex in the field of the IoT, where systems can be held in pervasive objects and small interconnected “things”, without displays to provide security information. In fact, what makes IoT special is that it connects the physical world to the digital/virtual world.

Several key IoT features are severely hardening the security problems. Heterogeneity, low energy and the huge number of IoT devices are three illustrative examples of these features.

Furthermore, when we look at the Internet of Things as a system, several actors contribute to the IoT security [1] :

- Hardware developers (devices)
- Middleware developers
- Application developers
- Protocol developers
- Middleware platform operators
- Application services operators

This chapter focuses on the security requirements of IoT systems. It presents the security problems - risks and vulnerabilities - inherent to IoT as well as the solutions proposed by the academic and industrial communities.

This chapter is organised as follows : The first section presents the architecture and the special features of a typical IoT system. The second section gives a glance into the state of the art of security solutions and approaches in the IoT domain. The last section presents a synthesis of the previous sections and the context of the work carried out within this master project.

I.2 IoT : Architecture and Features

The IoT ecosystem is evolving at an incredibly high pace. In fact, new IoT products are introduced every few days which leaves us with a huge and increasing number of IoT solutions. An effort is being made to find a general and unifying model or architecture of IoT systems. Such model can help us better understand the overall system by decomposing it into layers or building blocks.

I.2.1 IoT Architecture

According to [2–4], an IoT system is composed of four layers (see Figure I.1):

1. **Perceptual layer** helps linking the physical world with the digital world through devices with sensing capabilities (e.g. sensors, GPS, RFID tags, etc). It also helps the digital world to

change the physical world via devices with actuating capabilities (e.g. smart door lock, smart bulb, etc).

2. **Network layer** represents the communication part of a device.

3. **Support layer** represents a reliable support platform for the application layer. Cloud computing is an example of a support platform offering computing and storage services.

4. **Application layer** represents the personalised applications and services offering to users interfaces and solutions to interact or manage the devices with sensing and/or actuating capabilities.

As illustrated in Figure I.1 [3], there is a transversal block representing the security management. It means that security mechanisms should be deployed at each layer in a coherent and complementary way to assure security from the application layer to the perceptual layer. Indeed, it is so called cross layering approach, which helps to develop an ab-initio security deployment.

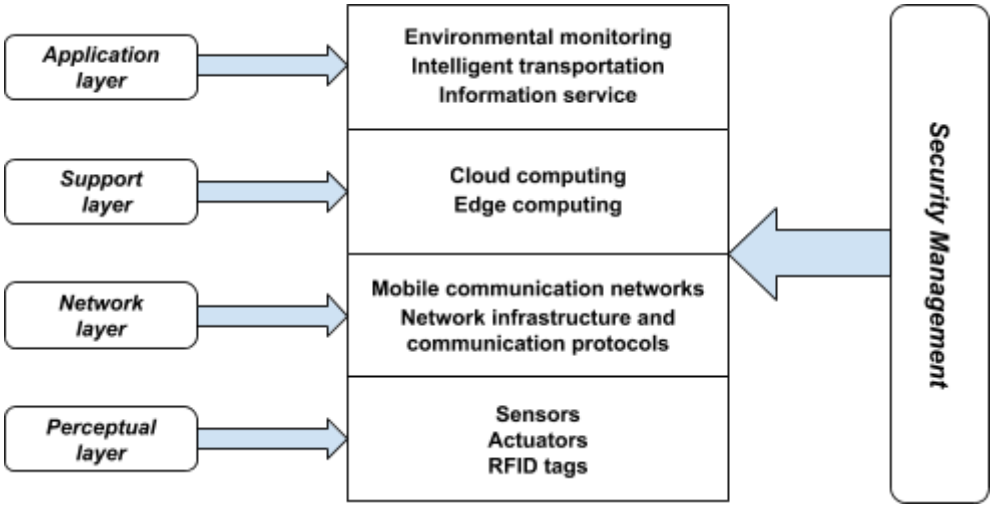

Figure I.1 : Architecture of IoT systems

I.2.2 IoT Features

The next paragraph presents the features of an IoT system at each layer. It is worth reminding here that some of these features could be related to more than one layer of the above-mentioned model.

A. Perceptual layer

IoT devices usually have the following properties:

- Low energy devices: Most of IoT devices are battery powered and should operate in an energy saving way.
- Limited computing resources: The majority of IoT devices have minimal processing and storage capabilities so they can only run basic and simple softwares or services.
- Low cost/quality device: To keep the prices of IoT devices low, vendors often use low quality hardware. Thereby, sensors' measurements or actuators' responses may lack accuracy.

- Physically exposed devices: Devices used in smart buildings or smart cities' infrastructures are deployed in a public domain, so they are physically exposed to tampering.
- Heterogeneity of devices : Devices are heterogeneous due to the fact that they have different purposes and capabilities : presence sensors, cameras, home automation actuators, etc.

B. Network layer

The properties related to the network layer are the following:

- Heterogeneity of protocols : The communication protocols and technologies are numerous and very heterogeneous.
- Dynamic networks with mobile devices : The dynamicity of the IoT networks is due to the fact that these networks are mostly wireless, with devices continuously joining and leaving the network, especially when dealing with mobile devices.
- Low latency requirements : One of the features of IoT data is its "freshness". Some applications (e.g. Telemedicine) require low latency between the end user application and the devices to allow real time and accurate decision making.
- Low energy: Communication protocols used should have energy saving features to help achieve a long and reasonable battery lifetime.
- Limited computing resources : These limitations rule out the use of popular communication protocols (e.g. TCP/IP) and cryptographic solutions (e.g SSL/TLS).
- Increase of IoT devices' deployments : The number of IoT devices (sensors, actuators, cameras, etc) that are being deployed is exponentially increasing.

C. Support layer

The Support layer allows a virtual/digitised representation of the real world which helps hiding heterogeneity of lower layers. IoT Support platforms are often based on Cloud Computing. Some of the most popular platforms are : AWS IoT [5], ThingWorx [6], Microsoft Azure IoT Suite [7], IoT lab [8], FIWARE [9]. These platforms offer frameworks and solutions that help user applications to connect and to manage IoT devices. Such platform need to have certain properties :

- Unified data model : Data can be exchanged between platforms only by using a unified data model. Otherwise, user's data will get trapped inside a single platform.

- Semantic interoperability : Data need to be exchanged between different platforms with unambiguous, shared meaning.

- Huge data storage : IoT produces huge amounts of data. To better trade-off storage cost and data access speed, platforms should offer multi-temperature data storage (hot, warm and cold storage).

- (Near) real-time data processing / Batch processing: Some IoT applications demand real-time data processing (e.g. home security). Other applications don't and platforms can use batch processing to process their data on free processing cycles.

D. Application layer

IoT applications have the following features:

- Manage personal and sensitive data: Data gathered and used by IoT applications can be personal and sensitive (e.g. a person's body temperature, location, etc.). This data is confidential and should be carefully managed.
- Real world Impact: The fact that applications can command devices with actuating capabilities means that a malfunction of applications could have many effects and percussions on the physical world.
- Data Brokering: Applications can sell public or private user data to organisations. A monetisation of the data should involve the users and respect legislations.
- Knowledge inference: IoT applications are treating data coming from different sources. They are expected to point out correlations between sources and produce new information about them.

I.3 IoT Security

The security aspects related to IoT are tightly linked to the characteristics of the IoT systems described in the previous section.

Conventional IT security requirements cover 5 aspects : Confidentiality, Integrity, Availability, Authentication and non-repudiation.

In The IoT context, non repudiation can be assured through proper authentication methods by verifying and proving the provenance of data. Therefore, non repudiation aspects will be discussed as part of the authentication aspects.

I.3.1 Security requirements for IoT

The next section presents the security requirements associated to each layer:

a) Perceptual Layer : Three levels of security are considered in this layer: authentication, confidentiality and integrity. Authentication is necessary to prevent illegal access to data generated by IoT devices. Confidentiality is needed in order to protect data on the IoT device and during transmissions of the data generated by the IoT devices. This requirement (confidentiality) also affects the network layer. Finally, integrity is required to detect illegal alterations of a device's data or software.

As an illustration, let's take the example of a LoRa sensor which measures temperature. The usage of an "application key" restricts data access to only legitimate applications and users. This guarantees authentication and confidentiality. Confidentiality can also be enhanced by encrypting memory so that data can still be confidential even if a hacker has physical access to the device and its memory.

To guarantee the software's integrity, the sensor can have secure boot feature to detect software alteration or introduction of malicious codes.

b) Network Layer : This layer has four security requirements : authentication, confidentiality, integrity and availability.

Authentication mechanisms prevent illegal access to the network. Confidentiality mechanisms protect data from eavesdropping attacks during the transmission. Data alteration during transmission, caused by transmission failure or malicious attacks, could be detected and even corrected using integrity mechanisms. Today's cryptographic mechanisms are strong and can assure identity authentication, data confidentiality and integrity. The problem is that they demand computing resources that IoT devices can't afford. To solve this problem, lightweight encryption technology is needed.

Finally, the availability of the network must be insured to avoid Denial of service attacks, especially after the recent experiments showing that attacks like distributed denial of service (DDoS) and jamming can be easily performed on some IoT networks, especially wireless ones.

As an illustration, let's take the LoRa example: Before transmission, the LoRa sensor encrypts packets using a "network key" to help authenticate the network. These cryptographic mechanisms guarantee authentication and confidentiality. Like all communication protocols, LoRa checks the integrity of transmitted data through a "Message Integrity Check" field in the packet.

c) Support Layer : The Support layer needs a lot of the application security architecture such as cloud computing with secure multiparty computation, multitenancy, antivirus, strong encryption algorithms and encryption protocols. All these mechanisms can be used to assure confidentiality, integrity and authenticity of gathered data. Trust mechanisms can also be used to evaluate the trustworthiness of devices or applications which helps detect misbehaviour and extract useful information from doubtful data. Trust is detailed in the next section.

d) Application Layer : The Application layer has conventional security requirements which are: authentication, confidentiality, integrity and availability. These requirements could be satisfied by conventional security solutions and mechanisms.

In summary, security requirements in the IoT domain are very important and full of challenges, mostly because of the IoT features or constraints that make the implementation of existing security solutions impossible. On the other hand, laws and regulations requirements in terms of data security and privacy reduce the flexibility margin.

The Table I.1 summarises the security requirements for each layer:

Layer	Security aspects	Semantic/explanation	Examples
Perceptual Layer	**Authentication**	Prevents illegal access to data generated by IoT devices	• LoRa "Application key" • Lightweight cryptography

	Confidentiality	Prevents illegal access to data generated by IoT devices	• Encrypting device's memory
	Integrity	Detects illegal alterations of a device's data or software	• Secure boot
	Availability	Data or service is available any time	• Many sensors deployment
Network layer	**Authentication**	Prevents illegal access to the network	• LoRa "Network key" • Lightweight cryptography
	Confidentiality	Protects data from eavesdropping during the transmission	• LoRa "Network key" and "Application key" • Lightweight cryptography
	Integrity	Prevents data alteration during transmission	• "Message Integrity Check" field in the LoRa packets
	Availability	Protects from DOS attacks	• Bluetooth Frequency hopping technique
Support layer	**Authentication**	Similar to conventional authentication mechanisms	• TLS/SSL (HTTPS)
	Confidentiality	Similar to conventional confidentiality mechanisms	• TLS/SSL (HTTPS) • Multitenancy
	Integrity	Trust	• Evaluation and reputation based systems (eBay's evaluation system for users)
	Availability	Similar to conventional solutions	• Cloud providers with multiple regions
Applications layer	**Authentication**	Similar to conventional mechanisms	• IdM (Shibboleth)
	Confidentiality		• Encrypt database • End-to-end encryption

	Integrity		• Digital signature
	Availability		• Redundant architecture

Table I.1 : Summary of security requirements at each layer

I.3.2 Security for IoT

In the next section, a state of the art of IoT security is presented at each layer of the IoT architecture.

a) Perceptual Layer

Without integrity of a device, the integrity of the whole IoT system can't be guaranteed. G. Hernandez et al. [10] proved that IoT devices can easily be tampered with and that countermeasures should be applied. Authors conducted an experiment on a "Nest" thermostat (see Figure I.2) which is a smart home automation device manufactured by the "Nest" company, acquired by Google in 2011 with 3.2B $.

Figure I.2 : Nest Thermostat (credit: Nest)

The authors were able to boot the device from USB and execute a malicious code. They proved through this experiment that the device must authenticate the code it runs. The memory should also be encrypted because attackers can introduce malicious codes into devices and can even extract potentially sensitive data like measurements and identity credentials. This can lead to the failure of the whole "Chain of trust" built from the perceptual layer (e.g. device) up to the application layer. An alternative suggested by the authors deals with two of the security issues addressed by the perceptual layer.

1. integrity of devices, based on the authentication of the software running on the device.
2. confidentiality of data, guaranteed by the encryption of the memory.

These solutions cannot be applied to limited resources and low energy IoT devices.

R. Tahir et al. [11] and X. Zhu et al. [12] propose two different ways of generating digital signatures and encryption keys from the internal behavioural characteristics of a device. The two solutions consist on generating signatures using the device's characteristics. These characteristics are repeatable, stable and difficult to capture or replicate by an attacker. R. Tahir et al. [11] generate signatures called "ICMetric" through a mathematical and statistical process. X. Zhu et al. [12]

generate signatures using Discrete Wavelet Transform (DWT) of a signal that uniquely identifies the device, like electric consumption over time.

The advantage of this approach is that the encryption key is not stored on the system, so an attacker cannot jeopardise it. Instead, it is regenerated from the signatures when needed. This enforces the confidentiality and the authentication of the device. The fact that the key is generated through the device's internal features gives assurance about the identity of the device and protects from device cloning. These solutions help authenticate devices and encrypt data by generating secure and unique identities and cryptographic keys based only on the device's features. The problem yet to be solved is : how to share these keys with the rightful users? X. Zhu et al. [12] propose a solution for this identity and keys exchange using a Blockchain-based Identity management framework. This solution will be discussed in the "Application layer" part.

In Addition to signature and key generation, X. Zhu et al. [12] propose correlation between devices (things) and their owners by including the owner's signature in the thing's identity to prove ownership. The Table I.2 shows the structure of an owner's and a thing's (device) identity. Each owner and each thing have : an identifier, credentials (key pair or password) and attributes. As you can see (in red), DWT signature can be used as credentials to authenticate a device. A user's signature can also be used to mark a device and prove the ownership.

Owner's Identity	Thing's Identity
Identifier: 160 bits hash value **Credentials:** Key Pairs **Attributes:** Since: xx/yy/zz Until: xx/yy/zz ...	**Identifier:** 160 bits hash value **Credentials:** Key Pairs Thing's DWT Signature **Attributes:** Since: xx/yy/zz Until: xx/yy/zz Owner's Signature ...

Table I.2 : Identities example of an owner and a thing

b) Network Layer

Mendez et al. [13] present several networking protocols used by IoT systems. The paper explains how popular and already deployed networking systems are adopting new mechanisms and standards in order to meet IoT systems requirements: cellular networks (LTE/LTE-Advanced), Local Area Networks (WiFi) and Personal Area networks (Bluetooth). The paper mentions new networking solutions specially conceived for IoT systems like NFC, 6LoWPAN, Zigbee, Zwave, LoRa, etc. Each one of these solutions offers mechanisms to guarantee all or a subset of the security requirements at the network layer: authentication, confidentiality, integrity and availability. For example, LoRa uses cryptographic mechanisms to guarantee authentication and confidentiality of data during transmission. LoRa also checks the integrity of transmitted data through a "Message Integrity Check" field in the packet. But the availability is not guaranteed by LoRa because it is very vulnerable to DOS attacks like jamming. These communication protocols help finding low energy networks operating

with limited resources. However, we find ourselves with a growing number of technologies supporting redundant features which worsens the heterogeneity of the IoT ecosystem.

S. Sicari et al. [14] mention the efforts being put to come up with lightweight encryption methods like lightweight ciphering algorithms and lightweight key generation/exchange techniques. Such encryption methods could help to guarantee authentication, integrity and confidentiality in a way that saves energy and computing resources. As an example of a lightweight method, authors mentioned Elliptic Curve Cryptography (ECC), which is an asymmetrical cryptography method (public/private key) that uses smaller keys but guarantees the same level of security as the widely used RSA method.

T. El Maliki et al. [15] suggest "Security Adaptation Reference Monitor" (SARM) which is an autonomous framework that deals with extremely dynamic security conditions (wireless networks, mobile endpoints, publicly accessible networks, etc). Through a context monitoring module and a feedback loop, SARM can correct and adapt the security means of the system in order to counter the effect of changes in the environment (see Figure I.3). The authors implemented their framework on smartphones and WSN. They tested it through a simulation of a city-wide WiFi network with multiple Access points. They simulated the presence of malicious nodes with security compromising and energy wasting behaviour (sinkhole behaviour). The results showed a convergence of the system after a period of time where user nodes could detect malicious nodes and adapt their security mechanisms to avoid them. This framework offers an adaptive and scalable model, which is well suited for dynamic networks. Nodes implementing this framework require a minimum amount of computing resources. This solution addresses the availability by detecting and eliminating perturbing elements connected to the network. It also addresses confidentiality issues by adapting security policies to the status of the system: for example, adapting the encryption strength to the context of the network.

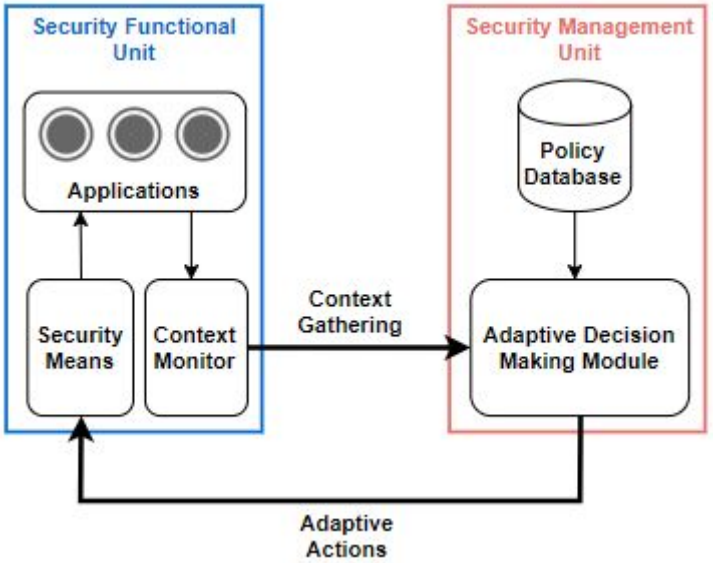

Figure I.3 : Adaptive Security Framework Architecture

By trying to use the minimal security means that could guarantee an acceptable security and performance levels, this approach guarantees a trade-off between energy consumption and system performance.

c) Support Layer

The Support layer is usually associated with Cloud infrastructures. As specified earlier, this support layer aims to provide a virtual/digitised representation of the real world. It often centralises all retrieved data in one “place”. In IoT, a common strategy consists on managing the data centrally. This oversimplified approach has a number of shortcomings, when scaled up:

- IoT applications require exchanging/processing data close to sensors and actuators.
- Data production is exploding, making the centralised cloud model ill-equipped to cope with the expected digital deluge.

To solve this problem, one of the promising solutions consists on deploying smart edge devices between the IoT devices and the Cloud: this is what is commonly called “edge computing”. In this case, the support layer is split between Edge and Cloud and security issues are also "distributed" between Cloud and Edge. Security issues linked to the support layer are similar to the ones known in conventional IT systems.

One of the security issues of IoT that can be solved at the support layer is trust and reputation. Trust based systems can help detect malicious or misbehaving devices and extract useful information from wrong or twisted data. A proposed solution is to estimate a device’s trust based on the received data. An estimation of a device's trustworthiness can be based on its behaviour in the perceptual (sensing/actuating) and network layers. The concept of trust and its mechanisms will be further detailed in Chapter 2.

N. Foukia et al. [16] present the "Privacy Verification Chain" (PVC) which is a framework that allows the exchange of data between participating entities (applications, users) based on digital contracts. PVC uses a blockchain-based solution to keep all the contracts available to all participating entities. So every entity has a copy of this database of contracts a.k.a "privacy ledger". Using the PVC, a user can verify which entities are using his data and under which terms/conditions (anonymity, price, etc.). Applications and service providers can prove that they are entitled to hold private information by presenting the contract between them and the data owner. This solution ensures privacy and confidentiality of data at an application level.

d) Application Layer

Besides the signature and key generation solution mentioned previously in the perceptual layer, X. Zhu et al. [12] present a solution for identity management in dynamic IoT networks. The authors suggest a distributed identity management (IdM) architecture using Blockchain technology to avoid a centralised solution. The centralised architecture requires a trusted third party to manage identity proof and credentials distribution (public keys). If used in IoT, such party will have millions or even billions of identities to check and will present a single point of failure. The reality is that IoT devices are deployed on public networks with no trust authority. This solution addresses the authentication by managing identities and distributing public keys, and the availability by offering a copy of the identity records at each node of the blockchain network. By facilitating the distribution of

asymmetric cryptographic keys, this solution enables the use of conventional cryptographic mechanisms to guarantee confidentiality and integrity requirements on network, support and application layers. This solution enables identity authentication, management of personal/sensitive data, multi-users applications in publicly exposed and dynamic networks. On the other hand, blockchain technology demands memory and processing resources to store the copy of whole blockchain and mine blocks which does not go well with IoT constraints.

[17,18] present some solutions to guarantee the confidentiality and the integrity of users' applications and data on multi cloud architectures, that can also be used if data or applications are being deployed on public or doubtful infrastructures :

- Replication of application: to deploy the same application on different processing nodes and then compare results to guarantee their integrity.

- Partition of application into tiers: no processing node has the complete application which guarantees the confidentiality of the application's logic.

- Partition or encryption of data: In case of data partition, no edge node has the whole data. Data is partitioned in a way that guarantees the confidentiality of sensitive/personal information of the clients. Another solution presented in [17], is the use of homomorphic encryption where processing nodes can operate directly on the encrypted data and produce the result in encrypted form. The processing node will produce a result without seeing the data in clear. This technology is very promising because it guarantees ultimate confidentiality and users could process their sensitive data in public computing infrastructures without fear of information disclosure. The problem is that the algorithms are far from being practical and make the solution unreal for the foreseeable future.

I.4 Synthesis

Section I.3 presented different alternatives to solve security problems related to IoT applications and platforms. These solutions, although inspired from the conventional security strategies, are taking into account the features and characteristics of IoT systems.

Some of these alternatives are hard to implement. This difficulty is not technical, but tightly linked to market, regulation and societal forces.

At the level of the first two layers (perceptual and networking), the contribution and involvement of manufacturers is inescapable. As specified by A. Robachevsky [1] "*... device vendors do not provide strong security because they do not bear the costs of security exploits. :"Consumers have no way to assess the security of the IoT system as a whole, thus diminishing motivation for the vendors to deliver secure solutions. Vendors are under intense competitive pressures to get their products to market as quickly and cheaply as possible, and to iterate with new versions rapidly. Security by design, done properly, costs money, requires skilled staff or consultants, and slows down the process. It cannot be "bolted on" as an afterthought – but that is how it is treated by many vendors, if they give it any attention at all.*"

Security solutions implemented on Application layer are similar to the the ones known in classical IT platforms and infrastructures.

Trust and reputation alternatives are deployed at the level of the Support layer. They attempt to establish a trust rate for each IoT device. The trust value assigned to a data (received from a given IoT device) is used to make necessary “arrangements”: isolate the IoT device, fix the problem, etc. These rates are often based on temporal and spatial criteria. Trust solutions will be detailed in Chapter 2.

I.5 The Context

In this work, we aim to contribute to the development of a toolset to “secure” IoT applications against suspicious exploitation. Securing IoT applications does not mean individually securing IoT devices used by the IoT application. We will target specific IoT applications which do not require securing individual IoT devices. These “non critical” applications do not have any “legal security” issues.

Security vs. trust

Device vendors do not provide strong security because they do not bear the costs of security exploits. This factor can limit the growth of the IoT ecosystem.

Despite the importance of this problem, this work does not address the problem of security at the level of the IoT device (perceptual layer). Our goal is not to guarantee the consistent running of all individual IoT devices, but to provide mechanisms that detect suspicious behaviours of IoT devices and warn the user to take appropriate actions. For this purpose, we will rely on the concept of “trust” which will be introduced in the next chapter.

Chapitre II Trust and Reputation Systems

II.1 Introduction

Conventional security methods, like authentication and integrity mechanisms, can resolve the untrusted network problem but cannot help with the untrusted devices problem. Most of IoT devices are often low cost, publically deployed and physically accessible. The resources, available on these devices, are very limited and do not enable them to support strong security mechanisms. These features make them exposed to malfunctioning or physical tampering. Such problems can generate corrupted and malfunctioning devices which make the data and networks unreliable. The previous chapter mentioned some of the solutions proposed by researchers, like trust-based systems.

When the processing resources are limited and can not support security overhead, trust and reputation could be an alternative. Trust and reputation are particularly interesting in large distributed systems with communicating “entities”. IoT systems are a good example of such environments because of the interactions and communications among the huge number of IoT devices. The main idea behind the “Trust and reputation” alternative is based on two attributes found in human relationships: Trust and Reputation [19].

This chapter presents the concept of trust and reputation as well as some of the architectures and solutions proposed, by the academic and industrial communities, in the domains of IT and IoT. The chapter is organised as follows : The first section presents the concept of trust and reputation. The second section details some trust and reputation models used within classical IT applications. The last section discusses trust and reputation models adapted to IoT domain.

II.2 Trust and Reputation

Although we rely on trust and reputation in our everyday life, it is difficult to define them accurately. The literature on trust and reputation is quite confusing since the definitions are numerous and different. Depending on the context, trust and reputation can be seen as moral or physical notions that help boost cooperativeness in communities or affect the stability of systems. In this paper, we adopt the following definitions :

Trust : Gambetta [20]: "trust (or, symmetrically, distrust) is a particular level of the subjective probability with which an agent assesses that another agent or group of agents will perform a particular action, both before he can monitor such action (or independently of his capacity ever to be able to monitor it) and in a context in which it affects his own action (...)". As a result of this definition, trustworthiness can be described as the probability of an entity to behave as expected.

Reputation : Reputation in general is an estimation of how an agent will behave in the future based on observations of its past behaviour. Reputation can either be the

accumulation of several observations from different communicating agents or it can be based only on the experience a single agent has made in the past [19].

Even after fixing these definitions, trust and reputation still seem very similar as they are both presented as estimations of the quality of future interactions. Misztal [21] helps distinguishing reputation from trust by presenting trust as a prediction of the outcome of future interactions, based upon a set of criterions. On the other hand, the same author presents reputation as a publicly held opinion, based on knowledge of the past, that is open to manipulation. Therefore, trust is a larger notion that can be backed by reputation but can ignore it and use other criterions or metrics.

II.3 Trust and reputation-based systems

Trust systems, which are based on reputation, can work in two ways: The first with a centralised unit that manages information to estimate trust between entities. The second is a distributed manner where each entity stores its own information about other entities.

Trust is a multidimensional metric. [22] distinguishes 4 dimensions of trust:

- **Device Trust :** refers to the need to interact with reliable devices such as sensors and actuators.
- **Processing Trust :** expresses the need to deal with correct and meaningful data. This demands accurate data gathering and suitable data analytics.
- **Connection Trust :** This is achieved by using cryptography techniques to assure confidentiality, authenticity and non-repudiation.
- **System Trust :** refers to the system's overall trust. This kind of trust is achieved by providing transparency of processes and by passing security and conformity certifications.

In the context of our work, we are interested in the device, processing and connection dimensions of trust.

Trust systems include a method of calculating the degree of trustworthiness. Indeed, the calculation of the trust must make it possible to take a decision. For example, a high degree of trust towards an entity makes it possible to judge this entity as reliable and to make the decision to trust it.

[23] claim that all Trust and reputation-based systems follow five generic steps :

- **Gathering Information :** The first step consists on collecting informations about the behaviour of the entities.
- **Scoring and Ranking :** Reputation and/or trust scores, of each entity, are calculated based on gathered information.
- **Entity Selection :** Selection of target entity based on reputation and trust scores.
- **Transaction :** A transaction takes place with the selected entity.
- **Reward and Punish :** In this last step, an evaluation of the transaction is made.

In what follows, a state of the art of trust and reputation-based models applied in classical IT systems is presented.

II.3.1 Trust and reputation-based models

Trust models have been proposed to describe and estimate trustworthiness in systems. According to [24], there are five categories of trust and reputation-based models. In each of the five categories, we will present the concept of the model and we will give an implementation example.

A. Bayesian networks

A Bayesian network is a way of representing a system as a directed acyclic graph, where nodes are variables and arcs are relationships between nodes. A node's state depends statistically on states of other nodes. For example, if we consider the weather as our system, we can represent this system as a graph of environmental variables (Snow, Pressure, Temperature, etc). The state of the variable "Snow" depends on states of variables "Pressure" and "Temperature". These dependencies or relationships between variables are probabilistic and are represented through statistical and probabilistic methods like Bayes' rules. If we use the previous example, the probability for snowfall is, statistically, very low knowing that the "Temperature" is high and "Pressure" is low.

Some Trust models are based on this theory. They perceive the trust of one entity towards another entity as a probability that an event takes place. This probability is estimated using some variables like history of good interactions or the performance of the target entity.

The Jøsang model [25] is a Bayesian approach which expresses trust towards an entity as the reputation of that entity. This approach expresses reputation of an entity as the probability that a future interaction, with this entity, will be positive. Jøsang uses the record of previous interactions to calculate this reputation. The reputation score of node x towards node y is calculated as follows :

$$R_x^y = \frac{\alpha+1}{\alpha+\beta+2}$$

α is the number of positive interactions of node x with the target node y, β the number of negative interactions of node x with the target node y.

This formula gives a measure of reputation and estimates how an entity is expected to behave in the future. The reputation value is in [0,1[, where 0 means that an entity has a bad reputation (untrustworthy) and 1 means that it has a good reputation (trustworthy). It would be more intuitive to make the reputation score in [-1,1[to have the neutral point at zero. Thereby, the reputation score is calculated as follows:

$$R'^y_x = (R_x^y - 0.5) \times 2 = \frac{\alpha-\beta}{\alpha+\beta+2}$$

We notice that this model expresses trust/reputation as a probability. This approach presents trust and reputation as a single value expressing the expectation toward future interactions/experiences based on previously conducted ones.

Following a different approach but ending up with a similar formula, eBay's reputation model calculates reputation of an entity as the percentage of its good transactions in the past. The formula is as follows :

$$R = \frac{\alpha}{\alpha+\beta}$$

α : good transactions; β : bad transactions

B. Discrete values

[26] rejects defining trust as a probability, because for them a probability only makes sense for similar repeated events. In their model, authors define trust as a subjective measure (belief) of personal experience in a particular context. For example, Alice can trust her neighbour Bob to fix her plumbing problem but cannot trust him to babysit her son. This subjective measure is propagated in the system under the term reputation.

The goal of this model is to obtain a measure of the semantic distance between the recommendation of a node B to the node A regarding node C, and the direct personal experience that node A really perceives. The belief of an agent towards another agent in a specific context is represented by 4 values: very reliable, reliable, doubtful, very doubtful. For example, if Alice asks Bob how much she can trust Cara and Bob tells her that Cara is "very reliable" and Alice finally realises that Cara is not "very reliable" but just "reliable", Alice realises that Bob's new recommendations should be lowered by one level. This semantic distance is called "trust recommendation value".

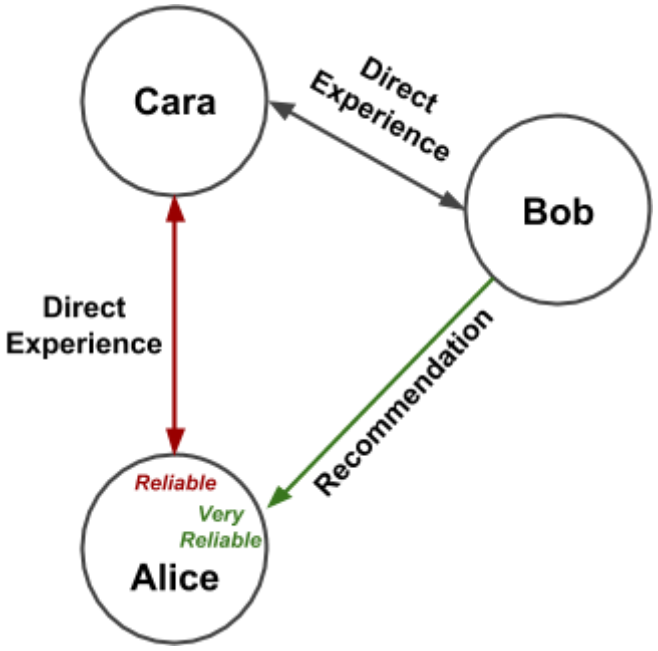

Figure II.1 : Semantic distance between recommendation and direct experience

In the same way, based on discrete values of reputation, [27] proposes the “TidalTrust” algorithm to calculate a movie’s reputation based on users’ ratings. Users rate a movie on a scale from half a star to four stars. Reputation is then calculated as follows:

- Retrieve proposed ratings of other users
- Use trust scores of users to weight their evaluations and calculate the reputation of a given movie. The calculation of users’ trust scores is not described in the paper.

Unlike our understanding of reputation (being a criterion in trust estimation), this approach deduces reputation from trust.

C. Fuzzy Models

Fuzzy models represent trust and reputation as fuzzy linguistic concepts. They describe how reliable an agent can be by formalizing rules based on a fuzzy approach.

In the REGRET e-commerce system [28], trust is a multi-dimensional concept. It has individual, social and ontological dimensions.

In their environment, agents can cooperate, compete or trade. Each agent can estimate trust value towards another agent (called target agent) using three values :

- **Direct-trust value:** This value is similar to the one used in previous models. It reflects individual dimension of trust estimation. Each agent evaluates transactions conducted with the other agents and stores evaluations locally.
- **Indirect-trust value:** Like the previous model, fuzzy models use recommendations of other agents. This value, called Indirect-trust, reflects social dimension of the trust estimation. Shared direct-trust values toward a certain target agent form its indirect-trust value a.k.a reputation value.
- **System-trust value:** This value reflects the target agent's trust based on its role in the system. It is similar to a recommendation given by the system.

By combining these three values, REGRET estimates a more complex trust value towards a target agent. As explained at first, fuzzy trust systems describe how reliable an agent can be by formalizing rules based on a fuzzy approach. For example, if a target has : "Reliable" as direct-trust value, "Cooperative" as indirect-trust value and "Neutral agent" as system-trust value, using fuzzy rules, we could calculate the complex trust value. An example of a fuzzy rule : IF "Reliable" + "Cooperative" + "Neutral agent" THEN trust value is "Good agent". The fuzzy rules are fixed by the user based on his domain knowledge. The fact that we could put the agent in a new category that is a combination of two categories (e.g. "good but selfish agent") reflects the ontological dimension of trust estimation using this approach.

D. Flow Models

Flow models calculate trust through transitive iterations and long loops. In some models, total trust is constant throughout the community, so increasing the trust score of one entity is done at the expense of others trust.

EigenTrust is a distributed algorithm that represents a Trust and Reputation security mechanism for peer-to-peer file sharing networks [29]. Each agent (peer) rates transactions that it had with other target agents. A transaction can be positive (satisfying) or negative (unsatisfying). A local trust score, for each target agent, is then calculated using the record of transactions conducted with it.

After normalisation, local trust scores are shared between agents by exchange of queries. Agents aggregate their local trust scores with exchanged ones to generate a global trust

score for each target agent. This long transitive chain of repeated and iterative calculations is conducted until global trust scores of agents converge.

E. Marsh's General Model

Marsh [30] defines three types of trust: The "basic trust" which is the general aptitude of an agent X to trust, the "general trust" is the trust of an agent X towards another agent and without taking into account a particular situation. "Situational trust" which is the trust of an agent in another agent in a given situation: $Tx(y, \alpha) \in [-1, 1[$; where y is the target agent and α is the situation.

Marsh proposes a method for calculating the "Situational trust" that takes into account the "general trust" that an agent has towards another agent, as well as the importance and usefulness of the situation: $Tx(y, \alpha) = Ux(\alpha) * Ix(\alpha) * \widehat{Tx}(y)$

With: $Ux(\alpha)$: the utility of the situation α for agent X; $Ix(\alpha)$: the importance of the situation α for agent X, that is to say, is there a good chance that the agent will obtain benefits? ; $\widehat{Tx}(y)$: general trust that takes into account all possible situations, if the agent is rational, he will calculate an average of $Tx(y, \alpha)$.

The following table summarises the previously mentioned trust and reputation models :

Model	Trust value (type)	components of trust calculation/estimation	Trust and Reputation estimation method	Example of an implementation
Bayesian Networks	Probability of a positive future interaction $R \in [-1, 1]$	• Record of interactions with target agent	$R_x^y = \frac{\alpha-\beta}{\alpha+\beta+2}$ α : Positive interaction β : Negative interaction	Jøsang Model: béta réputation
Discrete Values	Discrete values (numbers or states)	• Interactions with target agents • Recommendations of other agents	Aggregation of recommendations with regard to semantical distance	TidalTrust algorithm
Fuzzy Model	Fuzzy category	• Direct-trust value • Indirect-trust value • System-trust value	Combination of the three component through a fuzzy rule	REGRET e-commerce system
Flow Model	Number	• Local trust score • Normalised local trust scores from other agents	Aggregation of local trust score with exchanged trust scores to generate a global trust	EigenTrust algorithm

Marsh's General Model	Number $\in [-1, 1[$	• $Ux(\alpha)$: the utility of the situation α for x • $Ix(\alpha)$: the importance o the situation α for x • $\widehat{Tx}(y)$: the general trust that takes into account all possible situations (average)	$Tx(y, \alpha) = Ux(\alpha) * Ix(\alpha) * \widehat{Tx}(y)$	

Table II.1 : Summary of trust and reputation models

II.3.2 Characteristics of Trust Models

Although these trust models have some points in common, they have differences that form each model's strengths and weaknesses. The following table presents advantages (pros) and disadvantages (cons) of these previously mentioned models :

Model	Pros	Cons
Bayesian Networks	• Trust and reputation are simple to calculate	• Only takes into account direct transactions
Discrete Values	• Takes into account direct transactions and recommendations of others • Trust values have semantic meaning	• Excessive use of network to get recommendations of others
Fuzzy Model	• Takes into account different dimensions of trust : direct, indirect and system trust • Fuzzy rules are simple to calculate • Trust values have semantic meaning	• Excessive use of network to get recommendations of others to calculate indirect trust
Flow Model	• Takes into account direct transactions and recommendations of others • Each agent has a uniform trust value network wide	• Resource consuming due to continuous iterative calculations • Excessive use of network to get recommendations of others
Marsh's General Model	• A Generalised model that could be applied in a variety of domains • Context dependant trust	• Utility and importance are difficult to assess • Trust calculation not described

Table II.2 : Pros and cons of trust and reputation models

II.4 Trust and reputation systems for IoT

Previously mentioned models have proved their effectiveness in many domains such as e-commerce and grid computing. IoT systems come with a number of challenges and constraints that make the application of such models unrealistic. The next section presents some trust and reputation based systems developed by researchers for IoT systems.

There are two families of Trust models : centralised models and decentralised models. In centralised models, data used to calculate the trust of an IoT devices is gathered and centralised on the same server. In decentralised models, data used to calculate the trust of a given IoT device is collected from neighbouring IoT devices. This section presents some of the models in each family.

A. Centralised models

[19] presents an adaptation of the eBay model, originally used to evaluate the trusts of the sellers/buyers. In the original model, after each transaction, buyers and sellers evaluate each other and send ratings to a central server. The central server collects and accumulates all these feedbacks to calculate a so called Trust Score as an indicator of the trustworthiness of each actor. This Trust Score corresponds to the Reputation of this actor within the community/network. Before each transaction, an actor (buyer or seller) requests the Trust score from the central server.

To adapt this model to IoT, authors replace buyers and sellers by IoT devices.

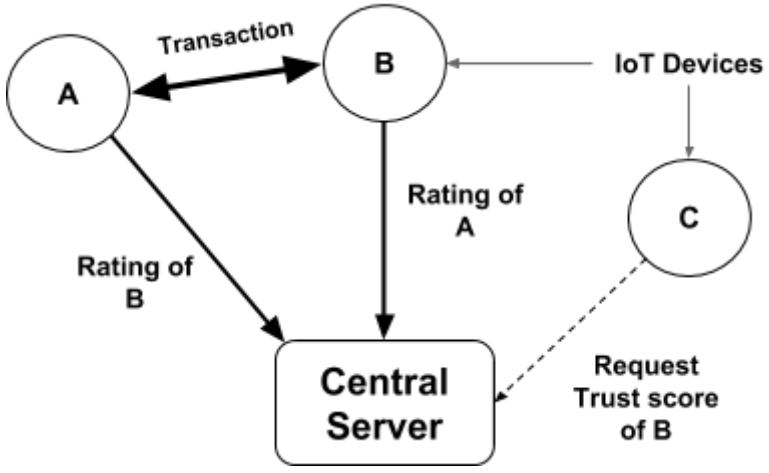

Figure II.2 : Centralised architecture applied to IoT

This centralised approach reduces processing and storage needs at a "Thing" level by delegating all the trust processing to the central server which has enough computing resources. There is also the fact that reputation is no longer calculated at each node but it is centralised. All the nodes have to do is request the reputation value from the central server. This central server has a global view of the system, so it can detect malicious nodes and inform other nodes about the situation. On the other hand, the central server presents a single point of failure and a bottle-neck which limits the availability and the scalability of the model.

[31] presents a new approach for trust estimation based on "frequent patterns". The idea behind this approach is the fact that, in many cases, sensors' measurements are timely and/or spatially

correlated: road traffic can be correlated with noise and air quality, temperature of a room can be correlated with the presence of a large number of people or with the closed and/or opened windows, humidity in a kitchen is correlated with temperature and physical presence, yesterday's temperature is close to today's temperature, etc. This approach is well adapted to centralised architectures since it requires a global view on sensors values.

Furthermore, if we use the history of the sensor's measurements, we can deduce a pattern that characterises a measure of a given IoT device collected in a given context. Here, the context is the set of measures collected at the same time (or slot of time) as the measure of the concerned IoT device. In Figure II.3, we aim at assessing the trust of the humidity sensor at time t. The context is the set: {<Temperature, 0>, <Luminance, 1>, <Motion, 1>}. For the sake of simplicity, we suppose that sensors give two possible values : 1 if the measured parameter is high and 0 if not.

A pattern is the set of <IoT Device, Value> pairs that frequently appear in the context history. A pattern may not contain all the <IoT Device, Value> pairs. In Figure II.3, the set {<Temperature, 0>, <Motion, 1>} is considered as the pattern of the blue context.

Patterns are generated as follows:

1. We retrieve the measure of the IoT device to evaluate (V). In Figure II.3, the IoT device to evaluate is the Humidity sensor and V=1.
2. We extract all the past contexts associated to the same value V: red circles in Figure II.3.
3. We identify the pattern by calculating the most frequently set of <IoT Device, Value> pairs that appear in the set of contexts of point 2. In Figure II.3, for the blue context, the set {<Temperature, 0>, <Motion, 1>} appears 2 times: t-4 and t-2.

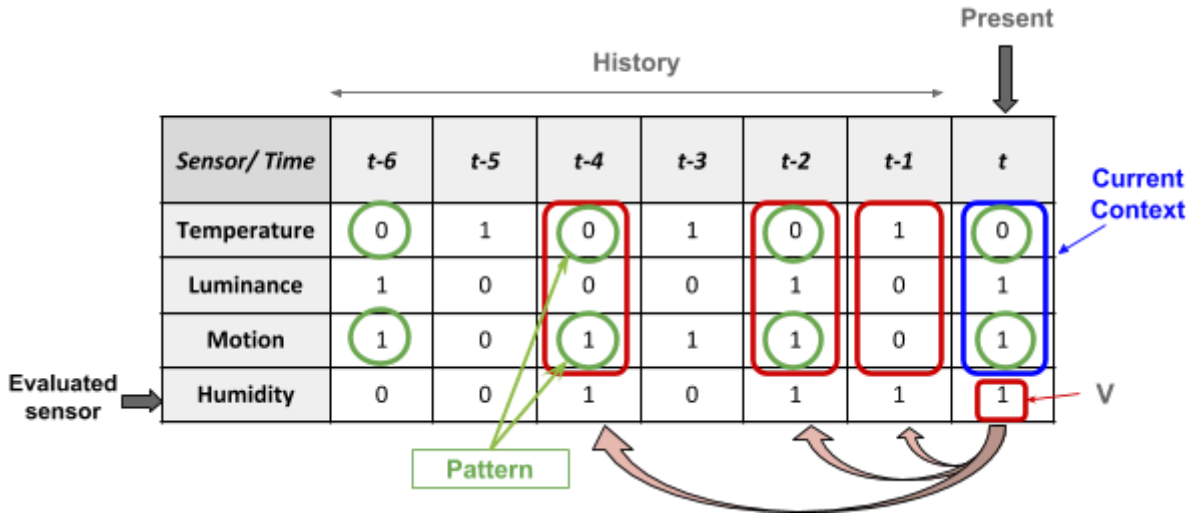

Figure II.3 : Pattern generation process

The trust is calculated on the basis of three parameters:

- Frequency of appearance of the pattern in the history of measurements (x1). In Figure II.3, the pattern occurred 3 out of 6 times (t-6, t-4, t-2). therefore, x1 = ½ .
- Strength of the correlation between the pattern and the value "V" of the evaluated sensor (x2). x2 is the ratio of the number of times the value "V" matches with the selected pattern and the number of times the pattern appears. In Figure II.3, the pattern appears 2 times with the value "V" (t-4, t-2). Therefore, x2 = ⅔.

- Size of the pattern (x3). In Figure II.3, the pattern is composed of two IoT devices : Temperature and Motion.

Trust is calculated by combining these three parameters as follows :

$$T = \alpha \times x_1 + \beta \times x_2 + \gamma \times x_3 + \delta$$

where $\alpha, \beta, \gamma, \delta$ are coefficients fixed based on user evaluation expressing the importance of each parameter in trust estimation.

To have a score in [-1, 1], authors use the previous formula as a parameter in a logistic regression function. Trust score is then calculated as follows :

$$T = \frac{1}{1+e^{-(\alpha * x_1 + \beta * x_2 + \gamma * x_3 + \delta)}}$$

Pattern detection algorithms can be greedy in term of computing resources. The complexity of such algorithms increases exponentially with the size of the sensors network, which makes this model hardly scalable.

[32] presents a solution to detect misbehaviour based on Anomaly Behavior Analysis (ABA). ABA is an intrusion detection technique that does not use databases with attack signatures or patterns like classical systems. Instead, it uses a baseline model representing the normal behaviour. This baseline model defines, for example, the normal consumption of computational resources like memory and CPU or any other feature that could help detect abnormal behaviour.

Such technique can be used to evaluate received data (good or bad) or to detect a misbehaviour of IoT devices and/or IoT applications. ABA can help estimate the IoT devices' integrity and trustworthiness which are very important, especially when using publicly deployed devices that are physically exposed to tampering.

The problem with this approach is that it uses machine learning techniques to construct the baseline model for normal behaviour, which can be demanding in term of computing resources.

B. Decentralised models

"TRM-IoT" [33] is a trust and reputation model designed to assess trust between nodes in a wireless sensor networks (WSN) based on their communication and routing performances. In the proposed solution, a node in the WSN operates in promiscuous mode, which means that it can see all packets within its reception range, even the ones that are not destined to it. As shown in the Figure II.4, node A observes its neighbour node B (a.k.a target node) and follows its forwarding behaviour. If node B receives a packet that is destined for node Y and it drops the packet instead of forwarding it, node A considers that node B has a bad behaviour. Node A can also retrieve the energy consumption of node B. Knowing the energy consumed during a routing and forwarding process, energy

consumption of node B helps node A find out whether node B is working hard enough to route and forward packets or not. These informations, form the direct trust of node A towards node B.

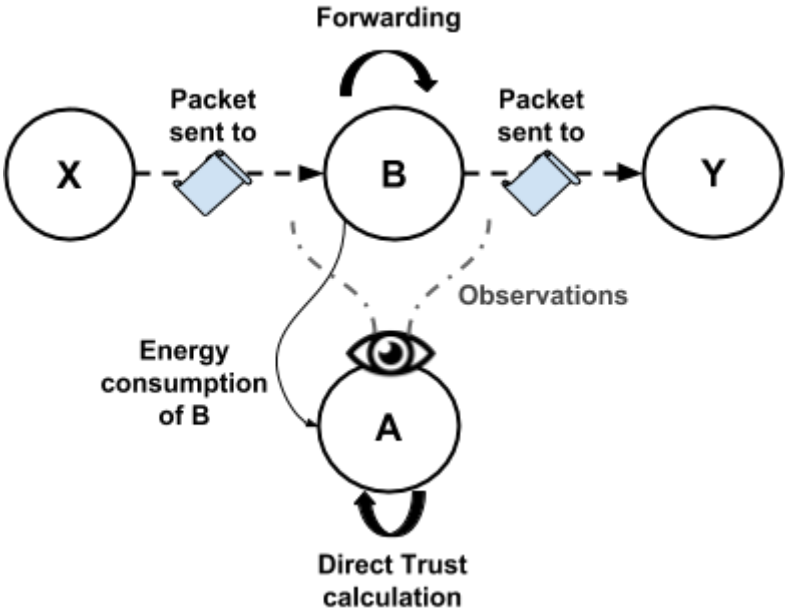

Figure II.4 : Direct trust calculation process

This model also uses recommendations of other nodes in the network. As shown in Figure II.5, Nodes C and D can give recommendations to Node A about Node B. A recommendation can be positive (Node B is good) or negative (Node B is Bad). Based on these recommendations, Node A deduces its indirect trust towards Node B.

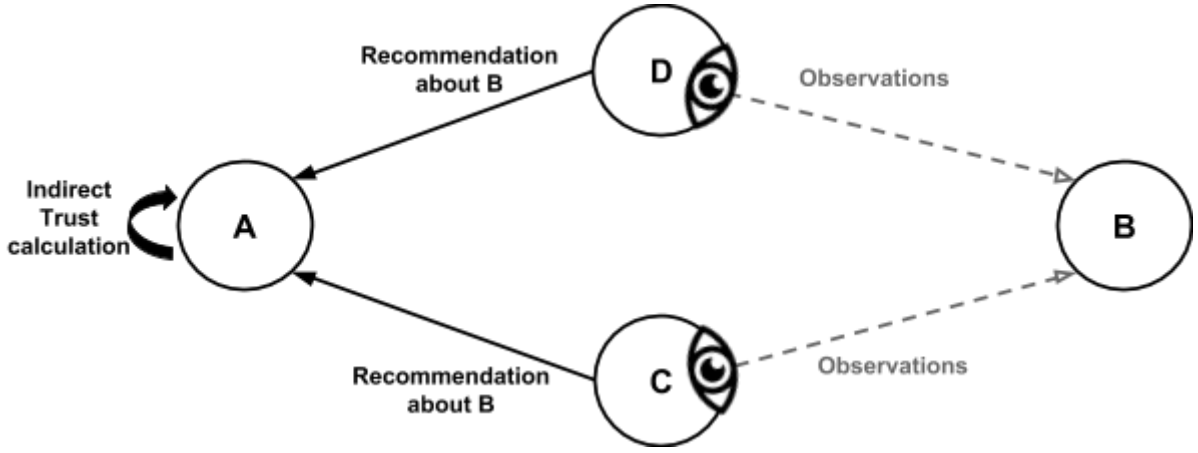

Figure II.5 : Indirect trust calculation process

By combining direct and indirect trust, Node A calculates the global trust towards Node B.

[19] presents a very similar model, called “Travos” model, that calculates direct trust from direct observations and indirect trust from recommendations of other nodes. But “Travos” adds a third

parameter which is the trust in the recommendations of other nodes. This parameter helps filtering recommendations that are originated by untrustworthy nodes.

The "TRM-IoT" and "Travos" models use a record of positive and negative observations/recommendations to calculate both direct and indirect trust. The decentralised architecture of the system makes it very scalable. On the other hand, this decentralisation makes identity management, nodes discovery and authentication of transactions difficult. Furthermore, nodes must have similar computing capabilities and energy resources in order to process and gather trust data in similar ways, otherwise powerful nodes will be privileged in trust estimations.

[34] proposes a model for distributed WSNs to help assess a node's cooperativeness and trustworthiness at the network layer. In a WSN, a node *n* receives messages from its neighbours. Each message contains the recipient's address. If this address is not *n*, the message is then forwarded to a neighbour node closer to the recipient node. Messages sent and received by a node *n* are also read by neighbouring nodes (within a given range). This means that a given node can "observe" its neighbours to check if they are forwarding received messages that are not intended for them. In other words, a node *n* observes its neighbour *m* if *n* can read messages received by *m*. The observation is "satisfying" (S) if *m* behaves as expected and forwards the messages. Otherwise, the observation is unsatisfied (U).

The error rate is thus calculated as follows: r = U / (U + S). This error rate is time-weighted: the more recent the error (unsatisfied observation), the more it affects the trust. The number of unsatisfied and satisfied observations are collected by node *n* during a given time frame (t), and the error rate is calculated : r_t. These time-weighted error rates express the weight or gravity of the misbehaviour.

A malicious node can misbehave by dropping few packets at timely-separated occasions in a way that produces a small weight of misbehaviour. This makes the malicious node undetectable. For this reason, the authors introduced another parameter called frequency of misbehaviour. As shown in Figure II.6, a data frame (t) can be divided into time slots (Δ). For each time slot, we calculate the error rate. If the error rate exceeds a fixed threshold, the time slot is considered a bad one meaning that a misbehaviour is detected during this period. The smaller the threshold is, the more we can detect small misbehaviours. The frequency of misbehaviour in a time frame (t) is the ratio of bad time slots to the total number of time slots calculated as follows:

$$f_t = \frac{Number\ of\ Bad\ time\ slots}{Number\ of\ all\ time\ slots}$$

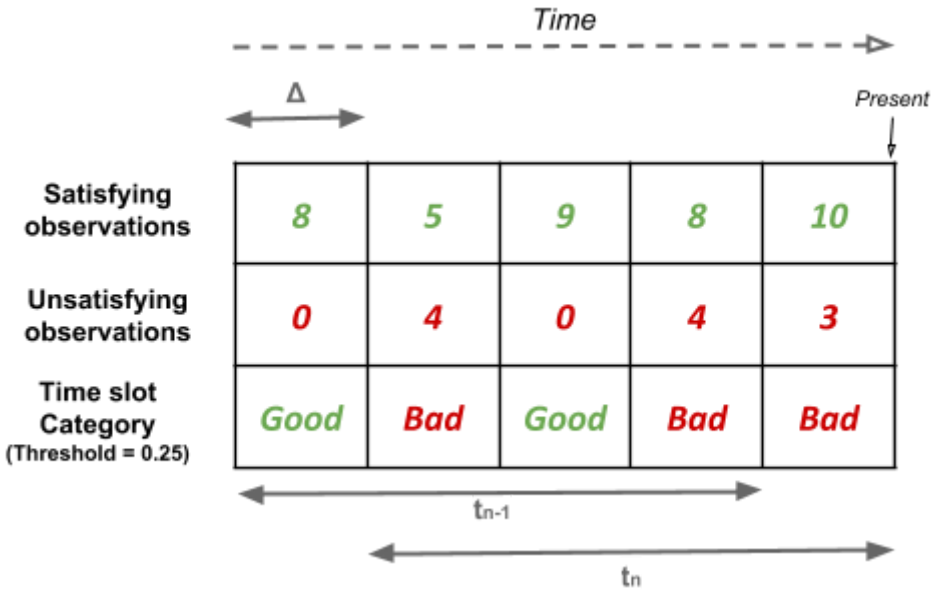

Figure II.6 : Frequency of misbehaviour calculation

If we take the example shown in Figure II.6, in time frame t_n, the first time slot (second from the left) is considered as a bad time slot because the error rate $(\frac{4}{9})$ exceeds the threshold (0.25). The frequency of misbehaviour is $\frac{3}{4}$ (3 bad time slots among 4 time slots).

The trust value of a node during a time frame (t) is calculated as follows:

$$T_t = \begin{cases} 1 - w_t \quad if\ w_t > f_t \\ \beta \times (1 - f_t) + (1 - \beta) \times (1 - w_t)\ otherwise \end{cases}$$

where w_t and f_t are, respectively, the weight and the frequency of the misbehaviour during the time frame (t) and β is the weight of the frequency of misbehaviour in trust score.

As you can see in the trust formula, frequency is used once it exceeds the weight.

II.5 Conclusion

Each of the previously mentioned models proposes a solution that helps applying trust and reputation models in IoT systems. Some of these solutions are : fixing metrics or parameters to evaluate transactions/observations, use simple calculations to estimate trust, respect communication architectures of the IoT system, store simple records containing only positive and negative record of transactions, use time-window mechanism to limit record's size, etc. These solutions can help us come up with our own trust and reputation model that fits the context of our project.

Chapter III Design of trust and reputation model

III.1 Introduction

The previous chapter presented trust and reputation models used within IoT systems. Such models enhance the security and the performance of IoT applications. IoT applications could be classified into two categories:

- Critical applications: Applications that can not tolerate errors and need a guarantee of integrity and proper functioning of devices. Therefore, critical applications need devices with high quality and security levels. These requirements can only be found in sophisticated devices that use heavy cryptographic mechanisms for authentication, integrity checks and other security aspects. Such devices are very expensive and power consuming. This makes them unsuitable for large scale IoT deployments.

  An example of critical applications is video surveillance. Surveillance cameras and the video streams they generate need to be highly secured through confidentiality and authentication mechanisms in order to protect privacy of persons. Another example of critical applications is power consumption monitoring. Power companies are beginning to use smart power meters that help remotely collect the power consumption of their clients. To avoid tampering attempts, energy meters are secured : physically by keeping them locked in electric cabinets and software-wise by using strong security means like secure boot and authentication mechanisms.

- Non-critical applications: Applications that tolerate a certain margin of error in devices' performance or data integrity. This type of applications can make use of mid/low quality devices with normal security levels, which allows the usage of low cost devices. Non-critical applications can be used with large scale IoT deployments, like smart cities and smart buildings solutions, that usually use low cost IoT devices.

  An example of non-critical applications is air quality monitoring. This application usually gets air quality measurements from different sensors that are deployed in different locations. First of all, this kind of sensors is not an important target of malicious entities. Because of the sporadic nature of wind movement, sensors will report different values which makes the application relatively tolerable to such divergent data. The same goes for noise measurement applications that use noise sensors to quantify noise levels at different locations.

In our work, we focus on securing non-critical applications. Instead of using strong security means, like critical applications, non-critical applications can use "Trust" mechanisms to assess the reliability and trustworthiness of IoT devices and their data. We propose a generic trust and reputation model

that estimates the trustworthiness of data and the reputation of IoT devices based on devices behaviour and data analysis. Our goal is to help users and applications detect and avoid misbehaving devices. Misbehaviour can result from the poor quality/performance of the device or from external disturbances (cyber attacks, physical attacks, honest physical obstructions, etc). Furthermore, by associating trust scores to data retrieved from IoT devices, we help applications take the necessary arrangements like data filtering or pretreatment, which improves the applications performance. Reputation scores associated to IoT devices help IoT applications estimate how devices would behave in the future.

This chapter presents a generic trust and reputation model. It is organised as follows : The first section presents our trust model and its components. This trust model is the core of our work. The second section presents an extension of our trust model that we call reputation model.

III.2 Proposed Trust model

To evaluate the behaviour, Trust models follow three steps :

1. “Interact” with the IoT device. Interactions with IoT devices could take many forms : asking for measurements when dealing with sensors, sending commands when dealing with actuators, getting video streams when dealing with cameras, etc.
2. Evaluate the quality and reliability of the interaction.
3. Use this information to deduce the trust score of the interaction and, therefore, the IoT device.

In this section, we present a new trust model which uses the three steps mentioned above. It starts by evaluating interactions with IoT devices and then use these evaluations to deduce a trust score for every interaction. A trust score reflects the quality and the reliability of the IoT device and the interaction itself.

III.2.1 Generic Trust Model

We suggest a “generic” trust model. What is meant by “generic” is that our trust model should support several types of IoT devices : sensors, actuators, cameras, virtual sensors, etc. Also, this model should not be linked to a specific IoT protocol or architecture. Therefore, we use a specific set of criterions for each type of IoT devices, that we call “parameters”. Every parameter should express a characteristic of the IoT device that could affect its trustworthiness or its efficiency. The nature and number of parameters depend on the device’s type. An interaction with the IoT device has an evaluation score relative to every parameter. We call this score a “parameter score”.

A “generic” trust model also means that it can assess trust for most (if not all) kinds of applications. As defined in the previous chapter, trust is a subjective measure. This means that two applications using the same IoT device (at the same period of time) could have different trust scores for the device and its interactions.

Therefore, we calculate a trust score as a combination of device-specific parameters and application-specific coefficients. A device’s parameter score “P” expresses the performance of the device related to a specific parameter that affects the trustworthiness and/or the quality of its interactions. An application coefficient “C” expresses the importance/relevance of a parameter to

the application. The trust score of an interaction with device “D” for an application “A” is calculated as follows:

$$Trust\,(Device\ D,\ Application\ A\,) = C_1^A \times P_1^D + C_2^A \times P_2^D + ... + C_n^A \times P_n^D$$

$$= \sum_{i=1}^{n} C_i^A \times P_i^D$$

where P_i^D is the parameter score of the interaction with device “D” related to parameter “i” and C_i^A is the coefficient of application “A” related to the parameter “i”.

The Figure III.1 illustrates the architecture and the workflow of our model. An interaction with an IoT device (For example, a measurement coming from a sensor) goes through “n” processes represented by the boxes : P_1, P_2, ... , P_n. Each of these boxes represents a parameter and will evaluate the interaction based on this parameter. This results in producing “n” parameter scores. By combining the scores of all the parameters, using the application coefficients and the trust formula mentioned earlier, a trust score of the interaction is calculated.

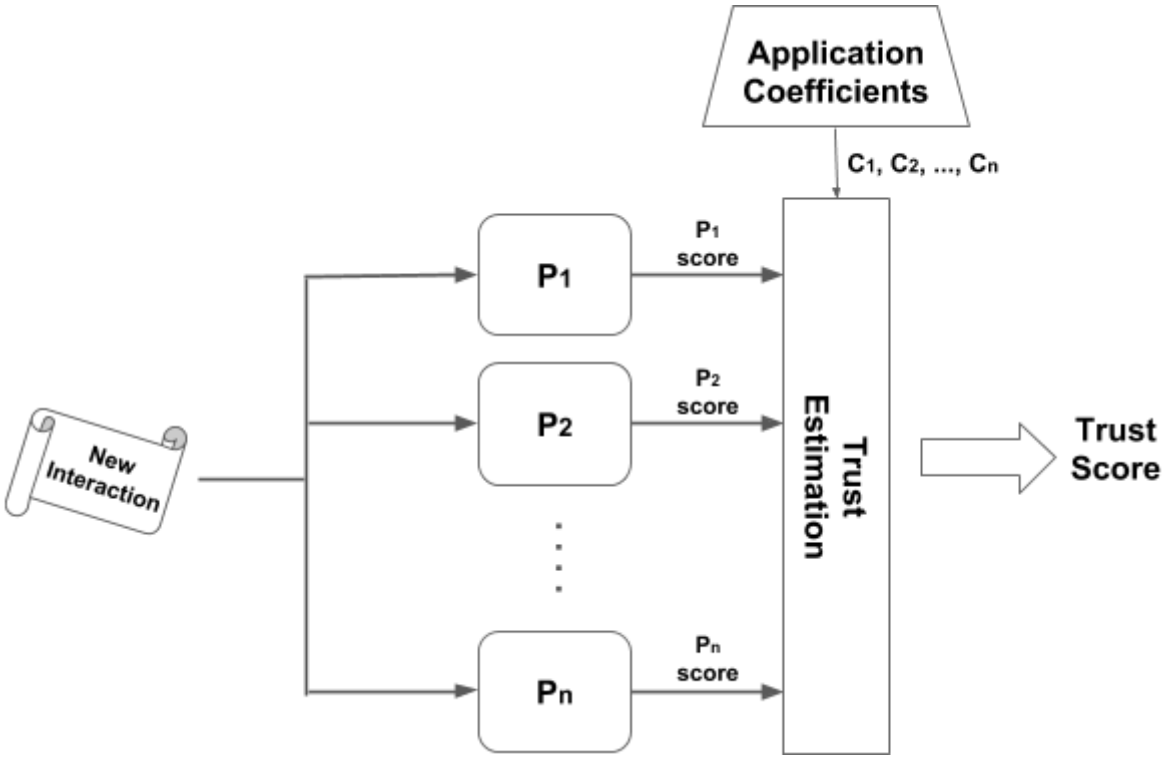

Figure III.1 : Architecture and workflow of the trust model

III.2.2 Parameters of an IoT device

To further explain the concept of parameters, let’s take the example of a wireless temperature sensor. This IoT device sends its measurements over a wireless network. To evaluate the

trustworthiness and quality of the measurements, the end user application could use parameters such as :

- **Sensitivity :** This parameter expresses the capability of our sensor to detect small variations in temperature levels. It is normally represented by the smallest variation that a sensor could detect. This parameter indicates the trust that an application could have in the sensor's measurements.

- **Packet Error Rate (PER) :** This parameter is the ratio of erroneously received packets to the total number of packets received. This parameter indicates the quality of the transmission channel between the sensor and the gateway.

For a camera device, these parameters could be:

- **Image resolution :** This parameter expresses the number of pixels in the captured image. This parameter indicates the quality of the image received.

- **Zoom range :** This parameter expresses the focal length range of our camera's lens. This parameter indicates the capability of our camera to get tight photos of a distant object or person.

III.2.3 Application coefficients

To further explain the concept of coefficients, let's take the example of a traffic monitoring camera. This camera is used by two applications, the application "A" is using the video stream to count big trucks and the application "B" is detecting license plate numbers. It is obvious that, compared to application "A", application "B" needs a video stream with a higher image resolution. So, if we use "image resolution" as a parameter for trust score calculation, trust score of the same video stream will be higher for application "A".

The difference in trust scores between the two applications is not due to difference in term of parameters, as they are using the same one which is "image resolution". It is due to the difference in importance or relevance of the parameter to the application needs. This importance could be expressed using a weight that we are calling "coefficient".

A coefficient is related to an application and a parameter. It expresses the importance of the parameter to the application. The question now is : how do we fix these coefficients ?

There are two ways to fix coefficients :

- The first is by going through a training phase, during which application results are evaluated at different levels of each parameter. This would help determine the impact of each parameter on the application's performance. At the end of the process, we end up with coefficients for each parameter.

- The second is based on domain knowledge. An application user can estimate the importance of each parameter. Let's take the previous example of application "A" that uses a traffic camera video stream to count big trucks. The camera has two parameters : "image

resolution" and "zoom range". Mike, a user, will fix the acceptable scores of each parameter at different trust levels. For example, if trust score is in the interval [0,1] (0 : totally untrustworthy, 1: totally trustworthy), Mike considers that trust score is higher than 0.7 (Trust > 0.7) if "Image resolution" is higher than 3 Megapixel and "zoom range" is higher than 35mm (3X zoom). Using the trust formula, we have the following inequality:

$$0.7 < C_1 \times 3 + C_2 \times 35 = Trust$$

With one similar inequality, we will end up with a system with two unknowns (coefficients C1 and C2) and two inequalities. The solution of this system will give us ranges from which we could choose the two coefficients.

This method can lead to a dead end if our inequalities system does not have a solution. In this case, a user can give coefficients based on his personal assessment.

For normalisation purposes, we suppose that parameter scores are in the range [0, 1] : 0 being a bad evaluation and 1 being a good one. Based on these assumptions, we know for sure that coefficients can only be positive numbers (≥ 0) because trust score will increase if one of the parameter scores increases and vice versa. A normalised version of the trust calculation formula would be :

$$Trust\,(Device\ D,\ Application\ A) = \frac{\sum_{i=1}^{n} C_i^A \times P_i^D}{\sum_{i=1}^{n} C_i^A} \qquad \text{(Formula I)}$$

This normalised version of the formula will help compare trust scores of all devices by keeping the trust scores in the range [0, 1].

III.3 Reputation of IoT device

By applying the trust model mentioned previously, we can have a record of parameter scores and, eventually, trust scores relative to each interaction with the IoT device. This record indicates the quality and trustworthiness of each interaction.

Using this record, we can deduce an indicator that expresses the overall quality and trustworthiness of the IoT device in the past. We call this indicator a "reputation score". This reputation score helps an application, or a user, predict how an IoT device will behave in the future based on the evaluation of its past behaviour. Therefore, this reputation score will help the application, or user, decide whether or not to engage in an interaction with the device.

In the previous chapter, we presented some of the reputation models used in classical IT systems like e-commerce platforms and grid computing networks. The same models could be applied In IoT systems. We simply need to replace entities, like buyers and sellers or grid computing nodes, by IoT devices.

In our work, we propose using two different reputation models :

- The first is an intuitive approach that calculates reputation score as the average of trust scores of all past interactions.

$$R = \frac{\sum Trust\ scores}{N} \in [0, 1] \qquad \text{(Formula II)}$$

N is the number of past interactions.

- The second using the Bayesian approach (see chapter II), where a reputation score is the ratio of good interactions to the total number of interactions. The formula is as follows:

$$R = \frac{\alpha}{\alpha+\beta} \in [0, 1] \qquad \text{(Formula III)}$$

α : good transactions; β : bad transactions

To apply this approach, first we need to classify the past interactions as good or bad. To do so, we propose that an application, or a user, has to fix a threshold for each of the parameters. As shown in Figure III.2, if a past interaction has at least one of the parameter scores below the parameter's threshold, we consider the interaction as a bad interaction. Otherwise, it is considered as a good one.

Figure III.2 : Evaluation of past interactions

The reputation formula (Formula III) can not give a reputation score to a new IoT device with no record of interactions $(\alpha = \beta = 0)$. Therefore, we need to change the formula to allow a new device to have an initial reputation score. This initial score needs to be low because a known attack on reputation systems is the identity spoofing. In fact, a node, with bad reputation score (for example, 0.3), could leave and then rejoin the network with a new identity. The reputation model will consider the node as a new one and gives it an initial reputation score. If the initial reputation score is high (for example 0.7), this bad node will keep his misbehaviour and use this technique periodically to stay undetected by our

reputation model. We propose an initial reputation score of 0. Therefore, the reputation formula becomes :

$$R = \frac{\alpha}{\alpha+\beta+1} \quad \in [0, 1[\qquad \text{(Formula IV)}$$

As you can see, a new device with no record of interactions $(\alpha = \beta = 0)$, will have a score of 0.

III.4 Conclusion

This chapter presented a generic trust framework used to evaluate interactions with IoT devices. It also presented a reputation model that helps an application, or a user, decide whether or not to engage in an interaction with the IoT device. The next chapter presents an implementation of our models destined to assess the trustworthiness and reputation of devices in an IoT infrastructure deployed at a city scale.

Chapter IV Implementation and Tests

IV.1 Introduction

The goal of this chapter is to test and evaluate the generic trust framework, detailed in the previous chapter, in the real concrete case of an IoT application. This application:

- collects noise measurements from a city-wide or district-wide network of IoT devices and,
- builds the "noise map" of a neighborhood or a city.

The chapter is organised as follows: the first section presents the noise application and details the purpose of the application, its architecture and the used IoT infrastructure (IoT devices and network). The second section presents the implementation details of the trust framework. The third section presents the experimental results. The last section discusses the results and suggests few improvements.

IV.2 IoT system : A3DB project

The Canton of Geneva is setting up a generic platform for IoT applications. This initiative, called Smart-Canton, is driven by applications: So far (January 2018), at least 15 use-cases are already identified. The platform must be fully operational by 2019. One of these use-cases, is the noise application (A3DB) detailed in this section.

The “Analyse 3D du Bruit” (3D Analysis of Noise) - A3DB - project, is part of the "Smart canton" program. The aim of A3DB project is to create a 3D map of urban noise levels in order to help better understand the urban noise phenomenon and identify its causes. This map will make it possible to target with greater efficiency the remediation measures that can be implemented to reduce noise.

Over a 1000 connected sensors measuring noise levels are planned to be installed in the municipality of Carouge (see Figure IV.1). The installation of the first devices on the facades (gutter, balconies, windows, etc.) began in the spring of 2016 [35]. At the time of writing (January 2018), there are around 700 sensors installed. Some of them started sending measurements since the beginning of 2017.

Figure IV.1 : Deployed noise sensor [35]

Measuring less than 15 cm, the sensors record ambient noise in the form of sound levels. These sensors are configured to send measurements to a central IT infrastructure every 15 minutes. Measurements are sent via a wireless communication LoRa network (Figure IV.2).

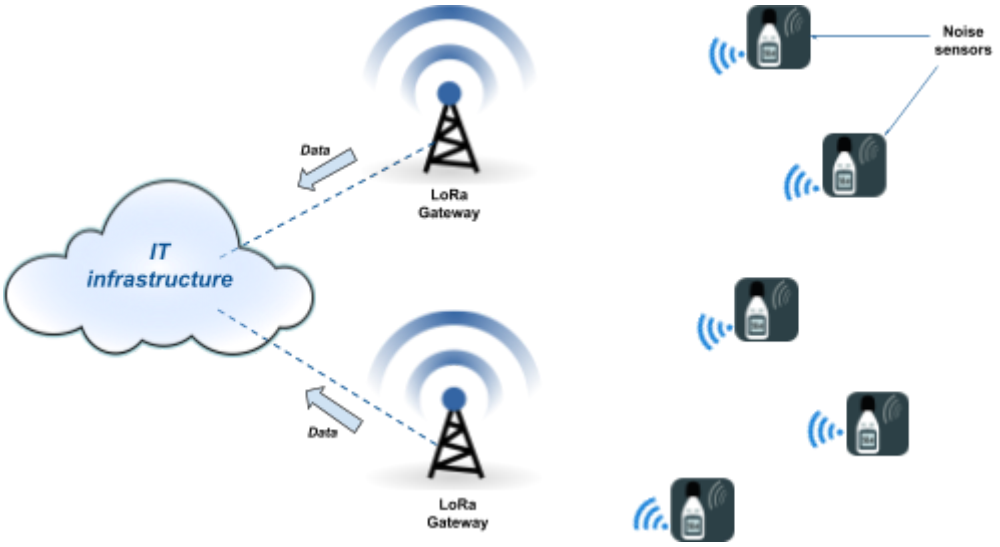

Figure IV.2 : A LoRa infrastructure is used to collect noise data

A noise sensor reads the noise level each second. These sample measurements are used to produce a summary report which is sent to the IT central infrastructure every 15 minutes. Here is a description of the content of a report sent by a noise sensor :

- **"idx"**: Report identifier, incremented by the sensor before sending any report. "idx" is used detect reports losses caused by transmission errors or sensor malfunctions.
- **"T"**: The timestamp indicating when the sensor began his 15 minutes noise recording period. For example, if a sensor sends his report with this timestamp : "2017-12-15T13:30:02.000Z", this means that the report corresponds to a recording period that started on 15.12.2018 at 13:30:02. The duration of the recording period is specified in the Duration value (see next “D”).
- **"D"**: The duration of noise recording period expressed in minutes. Usually, a sensor records during a period of 15 minutes.
- **"Leq"**: The average noise level during the noise recording period. The unit is decibels (dB).
- **"Lmin"**: The minimum noise level detected during the noise recording period. The unit is decibels (dB).
- **"Lmax"**: The maximum noise Level detected during the noise recording period. The unit is decibels (dB).
- **"L10":** The noise level above which 10% of the sample measurements are found during the recording period. L10 is expressed in decibels (dB).

- **"L50"**: The noise level above which 50% of the sample measurements are found during the recording period. L50 is expressed in decibels (dB).
- **"L90"**: The noise level above which 90% of the sample measurements are found during the recording period. L90 is expressed in decibels (dB).
- **"L95"**: The noise level above which 95% of the sample measurements are found during the recording period. L95 is expressed in decibels (dB).
- **"nMeas"**: Number of sample noise measurements conducted by the sensor during the noise recording period. Usually, we have a measurement every second. Hence, nMeas must be equal to 900.
- **"Temp"**: Temperature level expressed in Celsius.
- **"Vbat"**: Battery voltage indicating the battery level.

IV.3 Implementation of trust and reputation model

IV.3.1 Implementation of Trust model

The trust function defined in chapter III is used as follows:

1. "Interact" with the IoT device.
2. Evaluate the quality and reliability of the interaction. The quality and reliability will be modeled by parameters as defined in chapter III.
3. Use this information to calculate the trust score of the interaction and, therefore, the IoT device.

It's worth reminding here that the parameters are deduced from the measures retrieved from the IoT device.

IV.3.1.1 Defining an interaction

Following these steps, we must start by defining the interactions that we are having with our IoT devices. Because we are dealing with sensors, It is obvious that interactions are reports received from the noise sensors. But, a sensor's report contains a lot of information : Leq, Lmin, L10, T, etc. Choosing a subset or only one of these values to express trust depend on the application requirements and the capacity of this value to reflect the behaviour of the device.

Since the parameters: temperature and battery level are not related to the noise measurement, they will not be used by our model. Duration "D" and "nMeas" are always at constant levels (15 minutes and 900 sample measurements). So far, "D" and "nMeas" do not bring new information which makes them useless for validating our model. From all the noise level values, we think that "Lmax" and "Lmin" express better the behaviour of sensors because they represent instant measurements. On the other hand, "Leq" and percentile levels, like "L10" and "L90", present average and statistical

values that could hide misbehaviour manifestations. Obviously, “idx” could be very useful in detecting loss of reports and timestamp “T” is necessary to help place reports in time.

This leaves us with the four values : timestamp “T”, report identifier “idx”, “Lmin” and “Lmax”. To keep our use case simple, we are only using one of the noise level values. We choose “Lmin” because it helps assess sensitivity of sensors. Therefore, our interaction with a noise sensor is defined as: a received report containing a timestamp “T”, and identifier “idx” and a minimum noise level “Lmin”.

As described in our trust model (see Figure III.1), we need to define parameters that help evaluate the quality and reliability of sensors reports. By associating a coefficient to each of these parameters that expresses the importance of the parameter to the application, we can deduce a trust score of a report as shown in Formula I :

$$Trust\,(Device\ D\ ,\ Application\ A\ ) = \frac{\sum_{i=1}^{n} C_i^A \times P_i^D}{\sum_{i=1}^{n} C_i^A}$$

In the next sections, we present the steps followed to choose the parameters and the coefficients and the approaches used to evaluate a sensor report according to each parameter.

IV.3.1.2 Trust Function parameters

[36] presents a set of standard parameters that are used to evaluate a sensor’s measurement in general. Here are some of the presented parameters :

- **Sensitivity :** The sensitivity of the sensor is defined as the lowest variation (in our case, variation in noise level) that the sensor can detect.
- **Range :** The range of a sensor is the maximum and minimum values that can be measured.
- **Accuracy :** The accuracy of the sensor is the difference between the actual value (collected by an accurate sensor) and the indicated value sent by the sensor.
- **Response time :** The response time can be defined as the time required for a sensor output to change from its previous state to a final settled value.
- **Hysteresis :** A sensor should be capable to follow the input changes regardless of which direction the change is made; hysteresis is the measure of this property.

Although the majority of the presented parameters in [36] are not relevant to our noise sensors. We chose two parameters that could help assess the noise reports: Accuracy and Sensitivity.

In addition, based on the observations carried out, we can deduce two more parameters :

- **Availability :** The availability is the percentage of reports sent by a given a noise device. In our case, a noise device is assumed to send a report each 15 minutes. If the LoRa gateway receives 1 report during 30 minutes, this means that availability is 0.5.

- **Reports Reception Ratio (RRR) :** RRR of a sensor is the ratio between the "properly" received reports and the total number of reports sent by the sensor. This parameter expresses the quality of the transmission channel.

We also thought of other parameters that could help assess the quality and reliability of sensors :

- **Origin :** The source of the data can be a factor in reliability assessment. In this work, we deal with a trustworthy measurements provider which is "Orbiwise". But in some Smart City platforms, we find measurements that are provided by open communities or anonymous parties. This affects the credibility and reliability of data coming from such sources.

- **Location and height :** Sensors that are installed in public and risky locations or at low heights are more exposed to physical tampering than sensors installed in secure locations or at high heights.

For the trust function we are applying to our noise sensors, the selected parameters are : Availability, RRR, Accuracy and Sensitivity. The rest of this section details these parameters and presents the evaluation mechanism for each one.

A. Availability

We know that a good noise sensor sends one report every 15 minutes. To detect sensors that do not respect this constraint, we define the parameter "Availability" as the number of reports sent by a sensor every 15 minutes during a period of time :

$$Availability\ (SD,\ ED) = \frac{\#\ reports\ received\ between\ SD\ and\ ED}{ED{-}SD\ (in\ minutes)} \times 15$$

where SD and ED are respectively the start date and the end date of the availability assessment period. Availability score is in the range [0,1] because a perfect sensor sends a report every 15 minutes will get a score of 1. A sensor that did not send any report will get a score of 0.

B. Reports Reception Ratio (RRR)

Ratio between received reports and total sent reports. Since the "idx" value (report identity) is automatically incremented by the noise sensor, it is very straightforward to calculate the number of missing reports :

$$\#\ missing\ reports\ =\ idx\ of\ the\ last\ received\ report\ -\ number\ of\ received\ reports.$$

Reports Reception Ratio is calculated as follows :

$$RRR\ =\ \frac{\#\ reports\ received}{\#\ reports\ received + \#\ reports\ missing}$$

RRR score is also in range [0, 1]. A perfect sensor will have no reports missing and will get a score of 1. A sensor that had all his reports missing will get a score of 0.

RRR is similar to Availability but it's different in two ways :

- Availability is calculated between two fixed dates, but RRR is calculated from the first report received from a sensor (normally since its deployment) until the last report sent.
- More importantly, the fact that we detect a gap between the identifiers of two consecutive reports, discharges the sensor from the responsibility of the reports loss. Having a gap means that the sensor generated and sent the lost reports, they just didn't get through to the IT infrastructure. The problem is then limited to a poor transmission channel. Hence, We can say that RRR expresses the quality of the transmission channel.

C. Accuracy

Accuracy of a sensor is the difference between the actual value (a.k.a true value) and the value received from the sensor. To calculate the accuracy, we need to know the actual value.

Accuracy could be guaranteed by using high quality sensors. Nonetheless, this solution is very costly. In our context (A3DB sensors), the goal is to deploy "off-the-shelf" noise sensors and use "smart techniques" to assess accuracy. Our model is based on two types of corrections :

- Spatial correlation: neighbour sensors should report similar measurements.
- Temporal correlation: noise levels follow a pattern, for example noise is high at rush hours and low around midnight.

The idea is therefore to deduce the actual value of the noise from these two correlations. We implemented three techniques to estimate the actual value. The first two use the spatial correlation : spatial covariance and spatial average. The third uses the temporal correlation.

1. **Spatial covariance:** This method is presented in [37], it is used when dealing with statistically correlated sensors. In fact, noise propagates through the air. This means that neighbouring sensors should report similar values. Using this correlation, we can estimate the value that a sensor should report based on the reported value of its neighbours. We consider the estimated value as the actual value. The estimated value is calculated as follows:

$$\widehat{S_i^j} = m_i + \frac{Cov_{ij}}{Var_j} \times (S_j - m_j)$$

where $\widehat{S_i^j}$ is estimated value of sensor "i" based on value of neighbour "j", S_j is the reported value of neighbour "j", m_i and m_j are the average of values reported respectively by sensor "i" and sensor "j", Cov_{ij} is covariance of sensors "i" and "j" and Var_j is the variance of sensor "j". $\frac{Cov_{ij}}{Var_j}$ represents the spatial correlation factor.

The estimated value using the spatial covariance method is calculated as the average of estimated values based on all the neighbours:

$$\widehat{S_i}^{c} = \frac{\sum_{j \in N} \widehat{S_i^j}}{|N|}$$

where $\widehat{S_i^c}$ is estimated value of sensor "i", $\widehat{S_i^j}$ is the estimated value of sensor "i" based on value of neighbour "j", N is a group of sensors containing "i" and all its neighbours and $|N|$ is the cardinality of the group.

2. **Spatial average:** Using spatial correlation but in a simpler way, this method estimates the actual value as the average of all the reported values of neighbouring sensors.

$$\widehat{S_i}^{a} = \frac{\sum_{j \in N} S_j}{|N|}$$

where $\widehat{S_i^a}$ is estimated value of sensor "i", S_j is the reported value of neighbouring sensor "j", N is a group of sensors containing "i" and all its neighbours and $|N|$ is the cardinality of the group.

In these spatial methods: "Spatial average" and "Spatial covariance", the choice of the neighbours is crucial. We recommend choosing sensors which are close to each other and have the same context (within the same street, following the same tram rail, no obstacles between sensors like walls, etc). This guarantees that neighbours detect the same noise events with similar intensity.

3. **Temporal correlation:** There are different methods that estimate values based on temporal correlation. Some of these methods use machine learning techniques to model the pattern of data over time. We will be using a much simpler method that uses the recent values of a sensor and estimates the actual value using a linear regression technique. The idea is basically to deduce a linear function that fits the most the trend of recent reports, as shown in Figure IV.3. The estimated value is the projection of this line at present time.

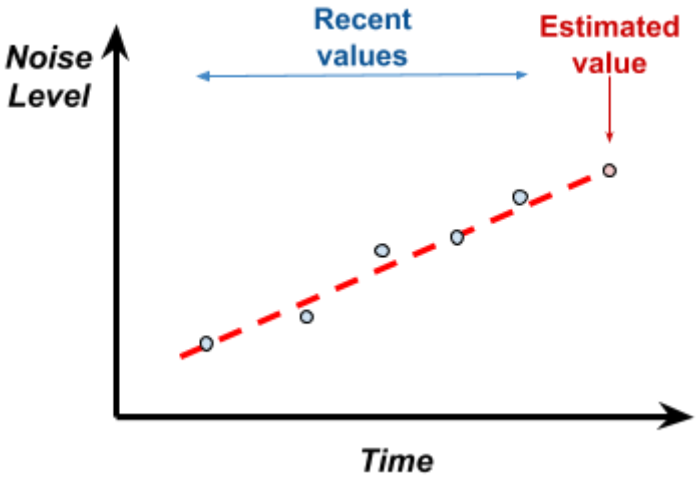

Figure IV.3 : Temporal estimation

Each of the mentioned estimation methods gives an estimation of the actual value. Our final actual value is a weighted average of the estimated values of the three methods :

$$\widehat{S_i} = \alpha \times \widehat{S_i^c} + \beta \times \widehat{S_i^a} + \gamma \times \widehat{S_i^t}$$

where $\widehat{S_i}$ is the final actual value of sensor "i", $\widehat{S_i^c}$, $\widehat{S_i^a}$ and $\widehat{S_i^t}$ are the estimated values of sensor "i" using respectively the spatial covariance, spatial average and temporal correlation methods. α, β and γ are the weights of the estimations of respectively the spatial covariance, spatial average and temporal correlation methods. After testing the estimation methods with different types of errors (see Annexe, section B), we realised that each method is best at detecting and estimating a specific type of errors/misbehaviours. We deduced the following weights : $\alpha = 0.4$, $\beta = 0.4$ and $\gamma = 0.2$.

After deducing the actual value, we evaluate the accuracy of the measurement by calculating the difference between the value received by the sensor S_i and the actual (estimated) value $\widehat{S_i}$. To have a score in range [0, 1], we present the accuracy formula as follows:

$$Accuracy = 1 - \frac{|\widehat{S_i} - S_i|}{S_i}$$

A perfectly precise measurement will have an accuracy score of 1. A measurement where $\frac{\widehat{S_i}}{2} \geq S_i$ or $2\widehat{S_i} \leq S_i$ has an accuracy score $Accuracy \leq 0$. In case the accuracy score is negative, we bring it to 0.

D. Sensitivity

Sensitivity of a sensor is its ability to detect small variations. Usually, sensitivity tests are conducted in controlled environments like laboratories. We aim to assess the sensitivity of a sensor by using its received reports. Our method consists on calculating the variation between consecutive measurements. Then, we calculate the number of variations that are in small intervals using a histogram as shown in Figure IV.4.

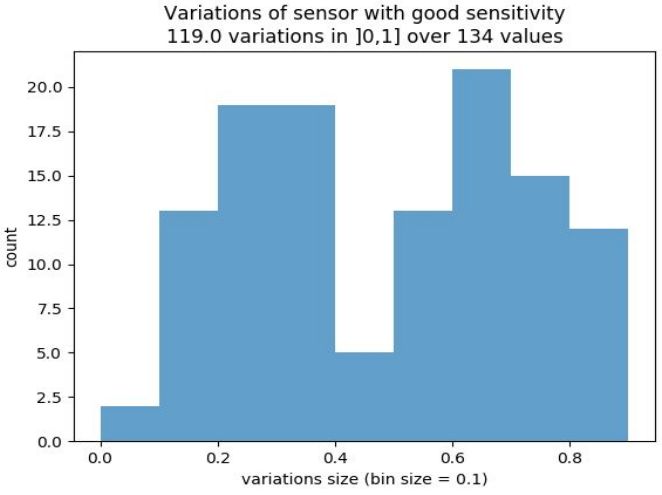

Figure IV.4 : Histogram of variations in noise levels

We calculate sensitivity as the ratio of the variations in the range]0,1] to the total number of variations. To improve our formula, variations close to 0 are given more weight than the ones close to 1. The sensitivity of a sensor is calculated as follows:

$$Sensitivity = \frac{\alpha_1 N_0^{0,1} + \alpha_2 N_{0,1}^{0,2} + ... + \alpha_{10} N_{0,9}^{1}}{Total\ number\ of\ variations \times \alpha_1}$$

where N_a^b is the number of variations in interval]a, b] and α_x are weights of the intervals with $\alpha_1 < \alpha_2 < .. < \alpha_{10}$ to give more weight to small variations.

Sensitivity parameter only makes sense when it is used to compare neighbouring sensors. Neighbours should report similar measurements, so they should have similar sensitivity scores. We should stress out that the choice of the neighbours is crucial for the good performance of the sensitivity method. We recommend choosing sensors who are close and have the same context.

IV.3.1.3 Application coefficients

The purpose of the massive deployment of noise sensors in Carouge is to create a map of noise levels in an attempt to locate the sources of noise pollution. If we consider this use case, we realise that accuracy and availability are the most important parameters. There is no need for a good transmission channel or a highly sensitive sensor (that detects small variations) to locate the big noise sources. Therefore, we decided to fix the coefficients as follows :

Parameter	**Accuracy**	**Sensitivity**	**Availability**	**RRR**
Coefficient	*20*	*2*	*5*	*0*

Table IV.1 : Fixed application coefficients

IV.3.1.4 Report Trust score

After fixing the evaluation parameters and their respective coefficients, we can calculate trust of received reports. As shown in Figure IV.5, a report goes through four evaluation processes, one for each parameter : accuracy, sensitivity, availability and RRR. The report gets a score for each of those parameters. These scores are weighted by the coefficients and then summed to form the trust score of the report, as presented by the formula :

$$Trust = \frac{C_{ac} \times Acc + C_{sen} \times Sen + C_{av} \times Av + C_{rrr} \times RRR}{C_{ac} + C_{sen} + C_{av} + C_{rrr}}$$

where C_{ac}, C_{sen}, C_{av} and C_{rrr} are respectively accuracy, sensitivity, availability and RRR coefficients. Acc, Sen, Av and RRR are respectively accuracy, sensitivity, availability and RRR scores.

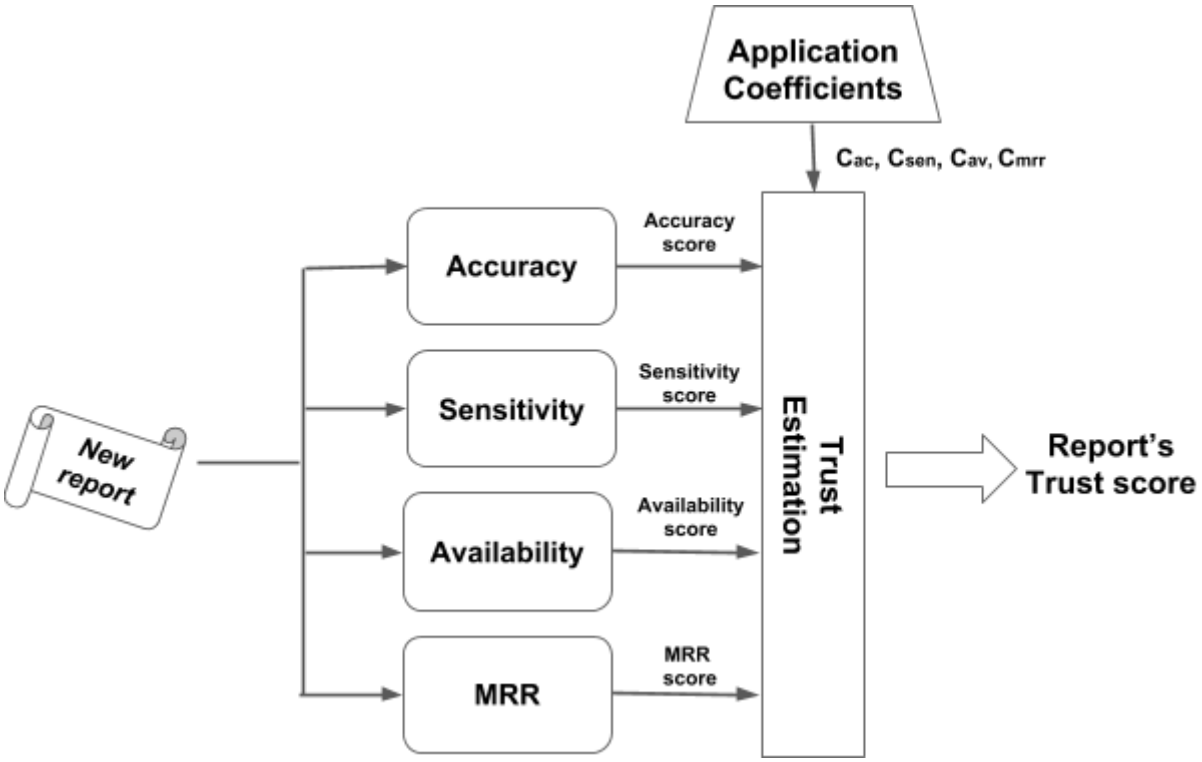

Figure IV.5 : Implemented trust model

Having trust scores for reports helps applications assess the quality and conformity of data to their needs or requirements. For example, an application can ignore reports with low trust scores or make use of them but with precaution.

IV.3.2 Reputation of sensors

After elaborating our trust model, we were inspired by some trust solutions that generate reputation scores for IoT devices. A reputation score expresses the performance and the reliability of an IoT device based on its past behaviour. Such scores help users and applications decide whether or not to engage in interactions with IoT devices. In our case, a reputation score will help an application decide whether or not to ask a sensor for measurements, based on trust scores of its past reports.

In our current system, retrieving sensor reports is free of charge. So an application can ask for measurements and then calculates trust scores and chooses if data is reliable or not. This is not an optimal approach for two reasons :

- Calculating trust for each report can be expensive in term of computational resources. Knowing that a sensor have always been giving good reports (so have a good reputation), can save us the trouble of calculating trust scores of his reports, at least in the near future.
- Data is not always free. A retrieval of measurements that have expected poor quality or unsuited for our needs, can be a waste of money and resources.

For these reasons, we propose our reputation model that is an extension of our trust model. As explained in the previous chapter, our reputation model calculates two reputation scores:

- The first reputation score is the average of all the trust scores of past reports received from the sensor. This score can be used by applications that are interested in the overall behaviour of sensors.

$$R_1 = \frac{\sum Trust\ scores}{N}$$

where N is the number of past reports

- The second reputation score is calculated as the ratio of bad reports to the total number of reports received from the sensor. A bad report is a report that has one of its parameter scores below a certain threshold (see chapter III). This score can be used by applications that have relatively strict requirements towards some or all of the parameter scores of a report.

$$R_2 = \frac{\alpha}{\alpha+\beta+1}$$

α : good transactions; β : bad transactions

As you can see, trusts scores are needed to calculate reputation scores. Therefore, we suggest to calculate reputation scores at the IT infrastructure to save applications the trouble and the cost of reports retrieval and trust calculation.

IV.4 Experimental results

In this section, we present the results of our trust and reputation model. The first part presents results of simulations that were used to validate our parameters and trust scores evaluation methods. The second part presents the results of our solution applied on real sensors.

IV.4.1 Simulations

In order to validate our model, we chose to start by testing it in perfect situations where neighbouring sensors are perfectly correlated. To have this perfect correlation, we simulated four sensors by generating four sets of measurements that are perfectly correlated. This will assure a good performance of our accuracy and sensitivity evaluation methods. To test each of the parameter evaluation methods, we introduced different types of errors in one of the simulated sensors. Then, we observed the capability of our model to detect the introduced errors.

IV.4.1.1 Simulated data generation

To generate our four datasets simulating four perfectly correlated sensors, we took a dataset of a real sensor and we created three copies. For each one of the copies, we introduced a slight offset and small fluctuations to simulate the distance effect between sensors. Figure IV.6 shows an example of simulated sensors' datasets.

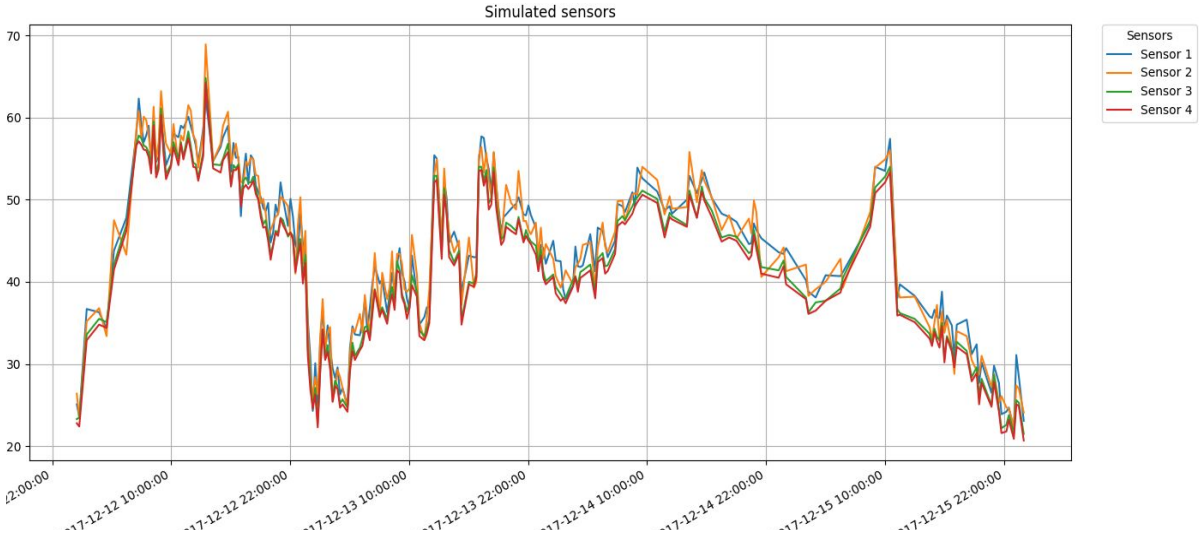

Figure IV.6 : Datasets representing simulated sensors

IV.4.1.2 Simulated errors and performance of evaluation methods

To test our evaluation methods, we introduce a specific type of error on one of the sensors and we see the performance of our methods in detecting and evaluating the errors.

A. Accuracy

As mentioned earlier, accuracy is the difference between the actual value (a.k.a true value) and the value received from the sensor. The challenge in accuracy evaluation is to estimate the actual value. To do so, we combine three estimation methods : spatial covariance, spatial average and temporal correlation.

As shown in Figure IV.7, one of the errors a sensor can show is an important offset in its measurements. This type of error can be due to an obstacle (snow, bird nest, etc) blocking the sensor's microphone and attenuating the real noise level, or can simply be due to a malfunction of the sensor. The red line presents the erroneous sensor's measurements (sensor 4) and the purple line presents the sensor measurements before introducing the error (what an accurate sensor should send).

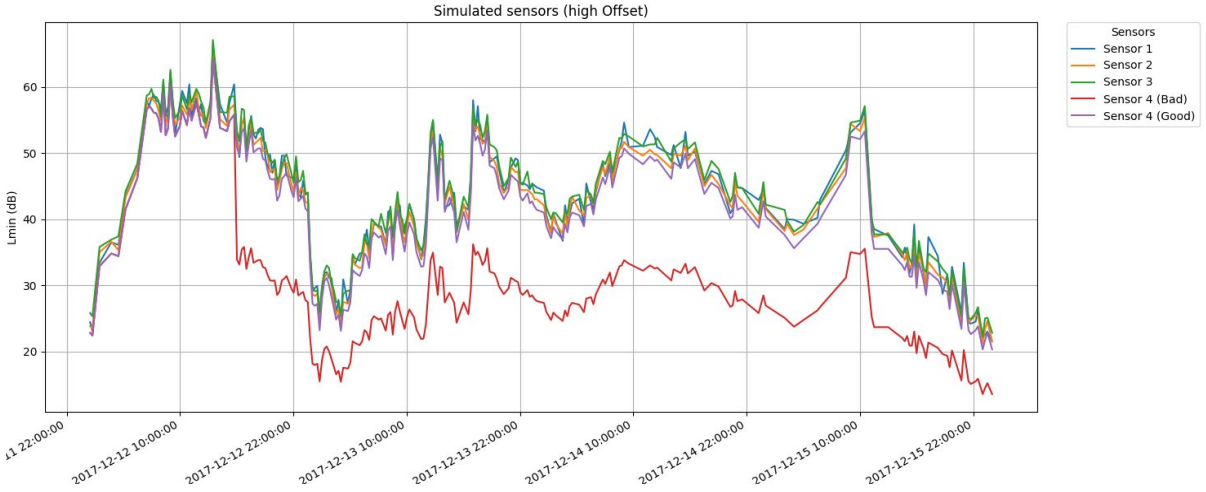

Figure IV.7 : Datasets representing simulated sensors with one erroneous sensor

The purpose of the estimation methods is to find the actual values which are normally unknown. So, the closer the estimated values of a method are to the unknown actual values, the more accurate they are. Having a good estimation method helps finding the actual values, which helps detecting erroneous values.

In our simulations, we have the actual value (purple line in Figure IV.7) presenting the values before introducing errors. Therefore, we can evaluate our estimation methods by comparing their estimated values to the actual values. Figure IV.8 presents the estimation errors of each of the estimation methods. The estimation error of a method is the difference between the method's estimated value and the actual value (purple line in Figure IV.7).

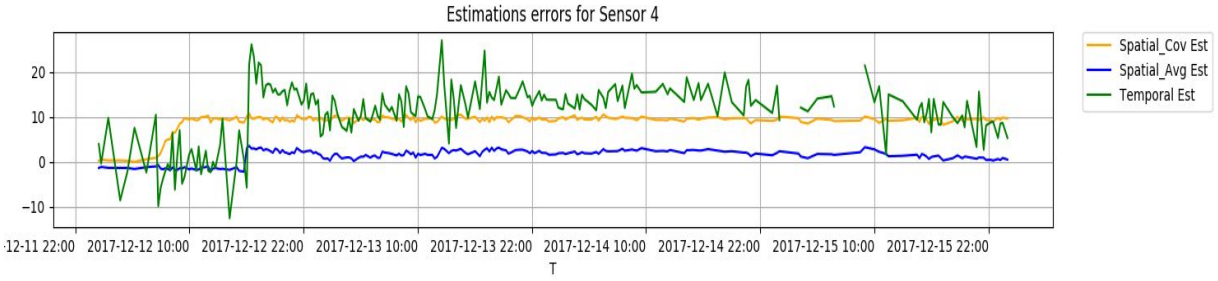

Figure IV.8 : Errors in estimations of actual values

We can see that spatial average method (blue line in Figure IV.8) gives the smallest estimation errors. So, the spatial average method is the best method to use to detect and evaluate the offset errors.

After testing the estimation methods with different types of errors (see Annexe, section B). We realised that each method is best at detecting and estimating a specific type of error. So, we decided to combine them as explained in section IV.3.1.2.C .

We calculated the trust score based only on the accuracy score. The results are shown in Figure IV.9.

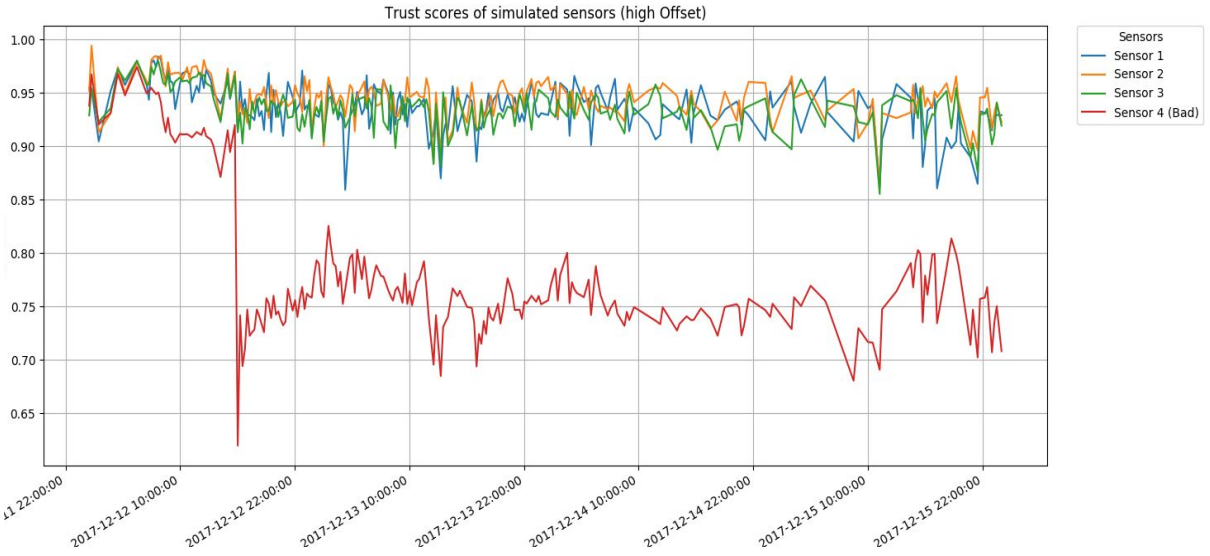

Figure IV.9 : Trust scores of simulated sensors measurements using only accuracy

As we can see, our accuracy evaluation method isolates the erroneous sensor by affecting worst trust scores to its measurements. Therefore, our method helps detect and evaluate bad measurements in the case of offset errors.

Other types of errors have been tested : measurements with high variance (fluctuations), offset+high variance, constant values, etc. Result of these types of errors are presented in Annexe, section B.

B. Sensitivity

Sensitivity of a sensor is its ability to detect small variations. Our method consists on calculating the variation between consecutive measurements. Then we calculate sensitivity as the ratio of the variations in the range]0,1] to the total number of variations. By giving more weight to variations that are closer to 0, we aim to improve our method's performance.

To simulate the situation of having two sensors with different sensitivity levels, we took the dataset of a sensor and we rounded up 70% of its measurements to 0.5 which minimises the number of small variations in the range]0, 0.5[. The dataset, before the rounding up, represents a reasonably sensitive sensor. The dataset, after the rounding up, represents a sensor with a bad sensitivity. The histograms of the number of variations in intervals in]0, 1] for the two sensors are presented in Figure IV.10.

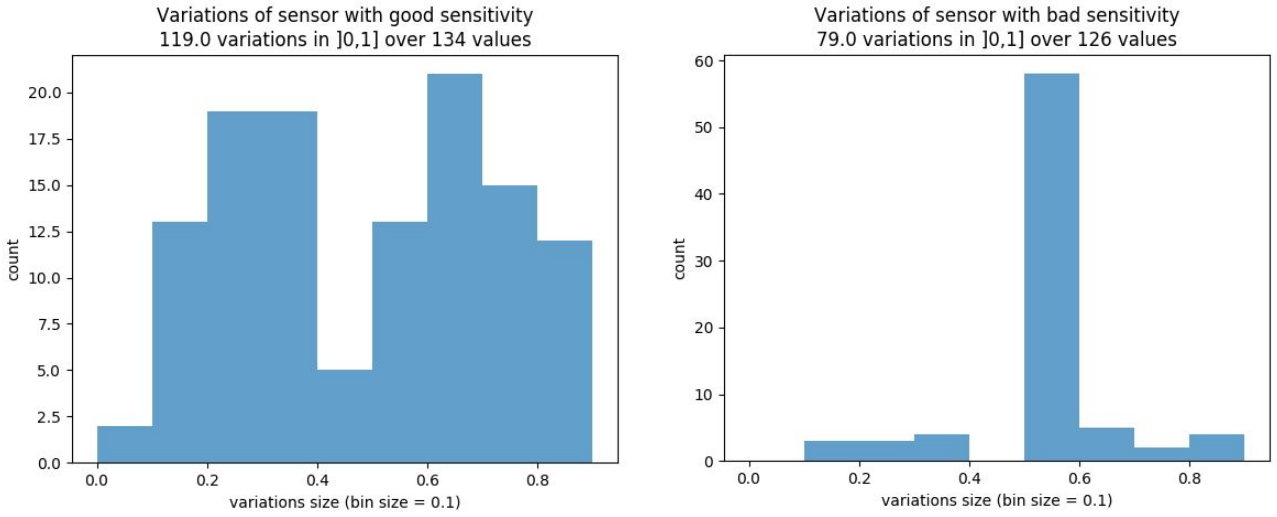

Figure IV.10 : Variations of reasonably sensitive sensor (Left) and a sensor with a bad sensitivity (right)

We notice that the good sensor (on the left) detects an important number of small variations within a variety of intervals (bins). On the other hand, the bad sensor detects mostly variations of 0.5 dB which is expected.

To test our sensitivity evaluation process, we calculated the trust score based only on the sensitivity score. The results are shown in Figure IV.11.

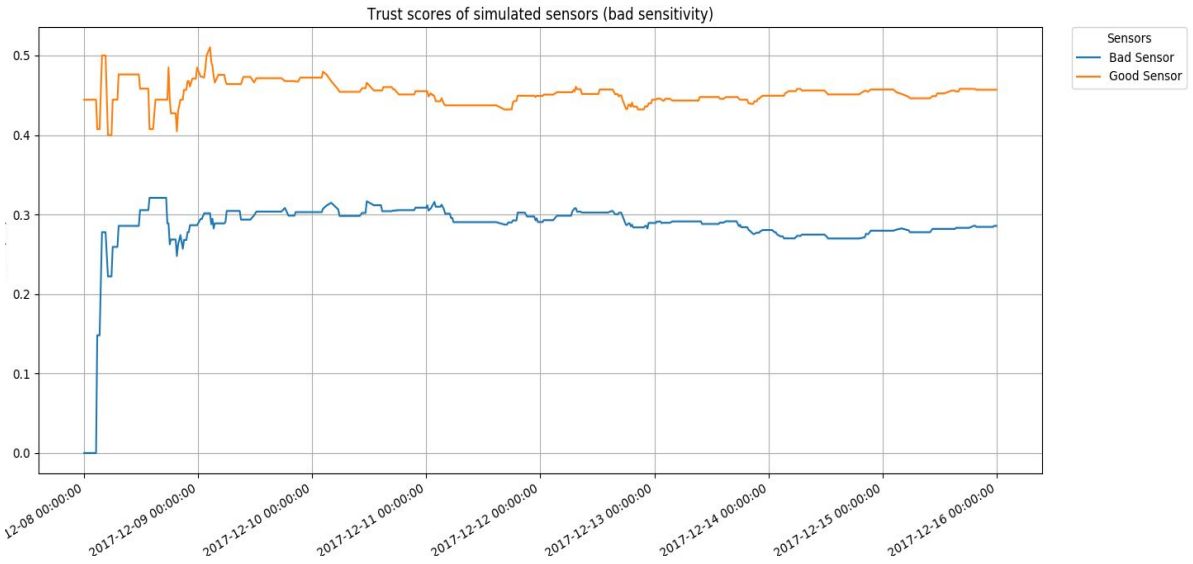

Figure IV.11 : Trust scores of simulated sensors measurements using only sensitivity

As we can see, our sensitivity evaluation method isolates the bad sensor by affecting worst trust scores to its measurements.

C. Availability

Availability score is the number of reports sent by a sensor every 15 minutes. To simulate the situation of having two sensors with different availability levels, we took the dataset of a sensor and we randomly deleted 70% of its reports. The original dataset represents a reasonably available sensor. The dataset, after deletions, represents a sensor with a bad availability. The two sensors are presented in Figure IV.12.

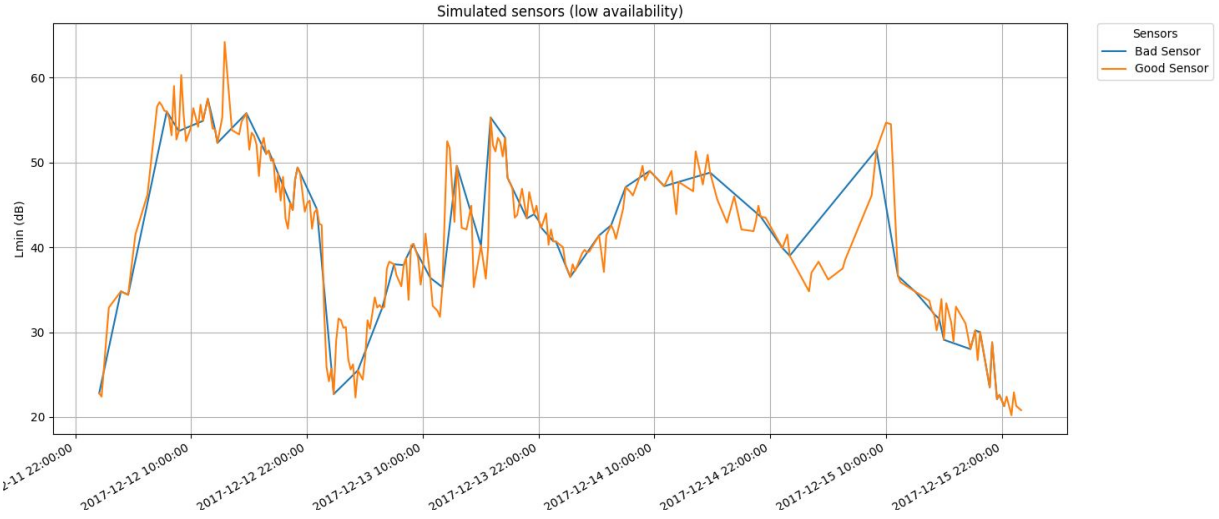

Figure IV.12 : Datasets of a reasonably available sensor and a sensor with a bad availability

To test our availability evaluation process, we calculated the trust score based only on the availability score. The results are shown in Figure IV.13.

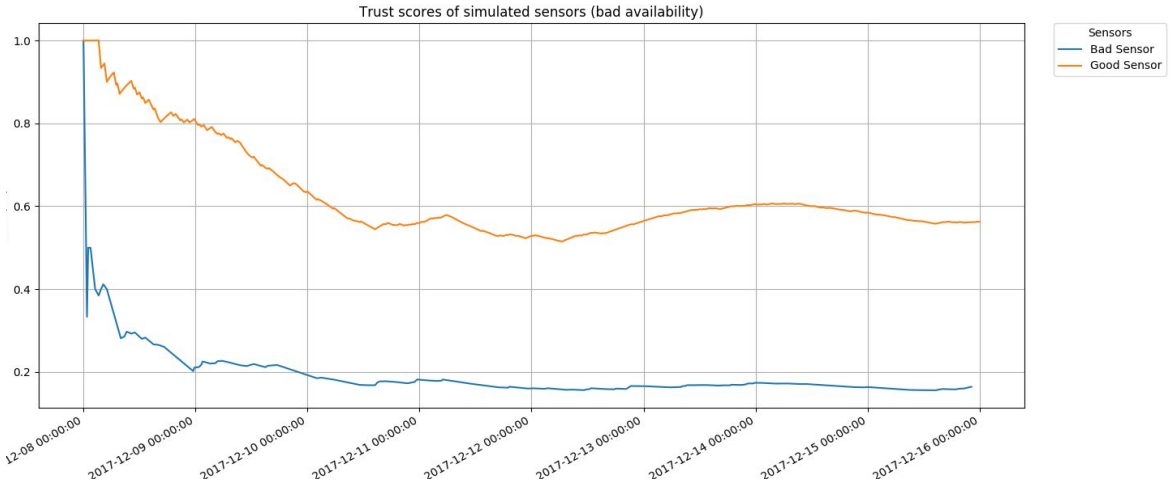

Figure IV.13 : Trust scores of simulated sensors measurements using only availability

D. Reports Reception Ratio (RRR)

RRR is calculated using the number of reports missing and the number of reports received. These numbers are retrieved from the RESTful API of the IT infrastructure. Simulating this parameter seamed unnecessary because its efficiency is considered valid.

IV.4.2 Experimental results

After validating the evaluation parameters with simulated data, we tested the trust framework on real sensors. First, we started by selecting a group of real sensors that have correlated measurements. To assure this correlation, we chose four sensors that are close to each other and have the same context (follow the same road). The Figure IV.14 shows the measurements received by each sensor.

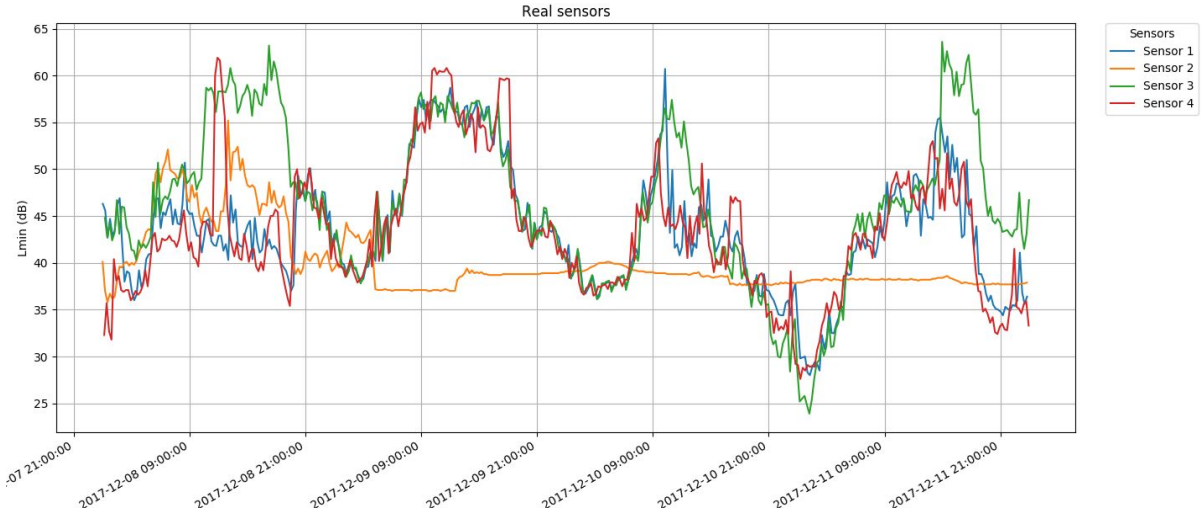

Figure IV.14 : Datasets of a real group of sensors

You can see that, as of 09.12.2017, “Sensor 2” is behaving strangely. The same thing goes for “Sensor 3” at 10.12.2017 but for brief periods of time. We calculated trust scores of all the sensors using the four parameters : accuracy, sensitivity, availability and RRR. The coefficients used are the ones fixed in section IV.3.1.3. Figure IV.15 shows the trust scores of sensors’ measurements over time.

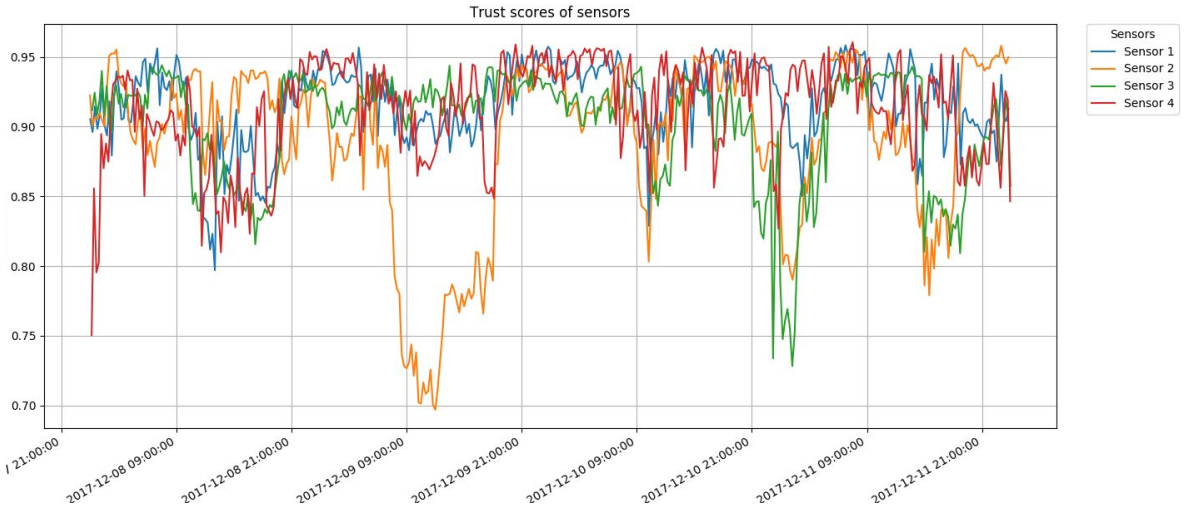

Figure IV.15 : Trust scores of the real sensors’ measurements

We notice in Figure IV.15 that our system gives worst trust scores to measurements of “Sensor 2” especially in the proportion of time where it behaves differently from its neighbours. Measurements of “Sensor 3” also had some bad trust scores but for brief periods of time.

Table IV.2 gives a global view of the parameters and trust scores for each sensor.

Sensor ID	Average Accuracy score	Average Sensitivity score	Average Availability score	Average RRR score	Average Trust score
Sensor 1	0.932560	0.495789	0.999984	0.667621	0.912695
Sensor 2	0.882854	0.649591	0.999997	0.637629	0.887270
Sensor 3	0.918126	0.458882	1.000000	1.000000	0.899297
Sensor 4	0.927456	0.482293	1.000000	0.635510	0.908186

Table IV.2 : Overview of average parameters and trust scores of sensors

We can deduce that, compared to its neighbours, “Sensor 2” has a bad average accuracy and, strangely, a high sensitivity. We can fairly say that “Sensor 2” is misbehaving. These results show that out trust model can help detect misbehaving sensors.

To test our reputation model, we had two ways of calculating reputation scores. The first using average trust scores. The second using parameters thresholds (see chapter III). For the second, we fixed the following parameter thresholds :

$$Accuracy = 0.8 \; ; \; Sensitivity = 0.1 \; ; \; Availability = 0.1 \; ; \; RRR = 0.45$$

Here are the results of our reputation scores calculation:

Sensor ID	Sensor 1	Sensor 2	Sensor 3	Sensor 4
Reputation score 1 (Average Trust)	0.912695	0.887270	0.899297	0.908186
Reputation score 2 (Thresholds)	0.991684	0.835759	0.977131	0.983402

Table IV.3 : Reputation scores of sensors

We can deduce that reputation scores calculated using the second method (Thresholds) reflect the behaviour of the sensors better than the first method (Average Trust). We can clearly see that "Sensor 2" should be avoided because it has a lower reputation score compared to other sensors.

IV.5 Discussion

IV.5.1 Deployment

After presenting and validating our trust and reputation model, it is time to discuss the deployment options of our solution. We believe that parameter scores calculation needs to be conducted on the IT infrastructure for these reasons :

- The IT infrastructure has a global view on sensors and is more suited to calculate parameters scores, especially the ones that use spatial correlation.
- The parameter scores only depend on devices and their measurements and don't change from an application to another. Only coefficients change. Hence, parameter scores need to be injected into the reports as metadata.

Following this deployment approach, an application gets a report with metadata containing parameter scores. Then, the application calculates trust score of the report using its own coefficients. Finally, the application uses the trust score as an element that helps decide what to do with the report : use it as it is, pretreat it, reject it, etc.

To calculate the reputation score, we need the whole record of the sensor reports. Therefore, it is recommended that reputation scores get calculated on the IT infrastructure. An application can send its thresholds for each of the sensor parameters and the IT infrastructure will response with the sensor's reputation score. Based on this score, the application decides whether on not to use the sensor.

It is worth mentioning that the developed algorithms can be executed on small edge nodes. Thus, a Raspberry Pi can be used as a pretreatment node that calculates parameters, trust and reputation scores of sensors and their measurements.

IV.5.2 Faced issues

During this work, we faced some issues. The first was that an important number of sensors have gaps in measurements records. Some sensors were not sending measurements for days. This issue slowed our validation process.

The second issue was that sensors are not perfectly synchronised. They don't send measurements of the same recording periods which forced us to make assumptions and consider that delayed measurements concern the same period. This had some effect on the correlation between sensors measurements.

IV.5.3 Critics

The trust framework proposed in this document assumes that the majority of the sensors, within a group of neighbouring sensors, behave correctly and safely. However, if the number of "bad" sensors becomes significant, parameters that are evaluated using spatial correlation methods can "backfire" and penalise good sensors.

The sensitivity evaluation method can mislead our judgement because noise gets rapidly attenuated through the atmosphere and small noise variations can vanish. So, a sensor "A", that is close to a small noise source, will report a small noise variation. A sensor "B", that is farther from the noise source, won't detect this variation, not because "B" is not as sensitive as "A" but because the small noise variation got attenuated. Our assumption that sensors within the same group should have the same small variations is not realistic.

Another weakness in our model is that we are supposing that our parameters are linearly dependent. For example, if coefficients of accuracy and sensitivity parameters are respectively 0.3 and 0.1, this means that an increase by 0.3 of accuracy score has the same effect on trust score as an increase by 0.1 of sensitivity score. This is not always true.

IV.6 Conclusion

This chapter presented an IoT infrastructure used as a backbone to assess noise pollution at a city scale. This infrastructure is used as a use case to test and validate the trust framework proposed in Chapter III. We explained how the trust model is implemented. We applied our model on simulated data to validate parameters evaluations. Then, we tested the trust model directly on real sensors. The results showed that our trust model helps assessing trustworthiness of measurements and detecting misbehaving sensors. At the end of this chapter, we discussed the deployment options, the faced issues and the flaws in our solution.

Conclusion and perspectives

Internet of Things is a new way of using internet. By connecting objects, opportunities seem endless. Today, IoT solutions and applications are numerous and the IoT Market keeps growing at a fast pace.

Although IoT is in full expansion, there are some issues that may slow its evolution. In fact, vendors were trying to keep a low production cost and a small time to market in order to have a bigger share of the IoT market. This rush had some effects on the IoT ecosystem like the lack of standardisation, fragmentation into alliances, security issues, etc.

This report focused on security. One of the security solutions that can be used in the IoT context are trust and reputation systems.

Trust and reputation mechanisms seem as a promising solution to solve the security dilemma, especially for non critical IoT applications. Non critical applications do not have strict security policies or legal issues and are tolerable to a margin of error. Thereby, we proposed a generic trust and reputation model that helps all kinds of non critical IoT applications assess the trustworthiness and the reliability of different types of IoT devices. The idea is to implement a trust and reputation platform at the support layer - e.g. cloud infrastructures, edge networks, etc - and propose it as a service for IoT applications and end-users.

To validate its effectiveness, we implemented our trust and reputation model on a large scale IoT infrastructure. This infrastructure is part of the A3DB project aiming to locate noise polluters over the Carouge city, it contains 1000 noise sensors. Thanks to the cooperation of the two project partners, Orbiwise and SABRA, we were able to perform tests on these 1000 noise sensors. First we used simulated data to validate our model. Then we applied our solution on the noise sensors in an attempt to validate its effectiveness in detecting misbehaving devices.

Our experiments showed positive results. They also showed that our model needs some reviewing especially in the way we combine parameters. A non-linear combination could be more realistic.

We believe that the performance of our trust and reputation model can be enhanced. One of the promising improvements are deployment options. By deploying our trust framework on edge devices, the scalability and the speed of our solution can increase drastically. Furthermore, machine learning (ML) techniques can boost the trust evaluation process. To use ML techniques, we need the processing power of cloud infrastructures. A combination of edge and cloud can bring the strengths of the two deployment options: a global trust framework deployed on cloud with global data and enough processing power to train ML-based trust models, and local trust frameworks deployed on edge devices and applying the trust models learned at cloud level. Moreover, we can apply our trust and reputation model on more applications using different types sensors. This will help us test and enhance the genericity of our solution.

References

1. Robachevsky BA. There is No Perimeter in IoT Security | Internet Society. In: Internet Society [Internet]. 2 Jun 2017 [cited 4 Jan 2018]. Available: https://www.internetsociety.org/blog/2017/06/there-is-no-perimeter-in-iot-security/

2. Daimi K. Computer and Network Security Essentials. Springer International Publishing; 2017.

3. Suo H, Wan J, Zou C, Liu J. Security in the Internet of Things: A Review. 2012 International Conference on Computer Science and Electronics Engineering. 2012. doi:10.1109/iccsee.2012.373

4. G. Yang, J. Xu, W. Chen, Z. H. Qi, and H. Y. Wang. Security characteristic and technology in the internet of things. Nanjing Youdian Daxue Xuebao . 2010;30.

5. AWS IoT – Amazon Web Services. In: Amazon Web Services, Inc. [Internet]. [cited 4 Jan 2018]. Available: https://aws.amazon.com/fr/iot/

6. ThingWorx IoT Platform | PTC [Internet]. [cited 4 Jan 2018]. Available: https://www.ptc.com/en/products/iot/technology-platform-thingworx

7. Azure IoT Suite | Microsoft Azure [Internet]. [cited 4 Jan 2018]. Available: https://azure.microsoft.com/fr-fr/suites/iot-suite/

8. DunavNET. IoT Lab [Internet]. [cited 4 Jan 2018]. Available: https://www.iotlab.eu/

9. FIWARE [Internet]. [cited 4 Jan 2018]. Available: https://www.fiware.org/

10. Hernandez G, Arias O, Buentello D, Jin Y. Smart nest thermostat: A smart spy in your home. Black Hat USA. pdfs.semanticscholar.org; 2014; Available: https://pdfs.semanticscholar.org/f1aa/f326c8b2cb6a94fa105b9910125e61920714.pdf

11. Tahir R, Tahir H, McDonald-Maier K, Fernando A. A novel ICMetric based framework for securing the Internet of Things. 2016 IEEE International Conference on Consumer Electronics (ICCE). 2016. doi:10.1109/icce.2016.7430694

12. Zhu X, Badr Y, Pacheco J, Hariri S. Autonomic Identity Framework for the Internet of Things. 2017 International Conference on Cloud and Autonomic Computing (ICCAC). 2017. doi:10.1109/iccac.2017.14

13. Mendez DM, Papapanagiotou I, Yang B. Internet of Things: Survey on Security and Privacy [Internet]. arXiv [cs.CR]. 2017. Available: http://arxiv.org/abs/1707.01879

14. Sicari S, Rizzardi A, Grieco LA, Coen-Porisini A. Security, privacy and trust in Internet of Things: The road ahead. Computer Networks. 2015;76: 146–164.

15. El-Maliki T, Seigne J-M. Efficient Security Adaptation Framework for Internet of Things. 2016 International Conference on Computational Science and Computational Intelligence (CSCI). 2016. doi:10.1109/csci.2016.0046

16. Foukia N, Billard D, Solana E. Privacy Verification Chains for IoT. Lecture Notes in Computer Science. 2017. pp. 737–752.

17. Bohli J-M, Gruschka N, Jensen M, Iacono LL, Marnau N. Security and Privacy-Enhancing Multicloud Architectures. IEEE Trans Dependable Secure Comput. 2013;10: 212–224.

18. De Capitani di Vimercati S, di Vimercati SDC, Erbacher RF, Foresti S, Jajodia S, Livraga G, et al. Encryption and Fragmentation for Data Confidentiality in the Cloud. Lecture Notes in Computer Science. 2014. pp. 212–243.

19. Eder T, Nachtmann D, Schreckling D. Trust and Reputation in the Internet of Things. Universität Passau, Tech Rep. 2013; Available: https://web.sec.uni-passau.de/projects/compose/papers/Eder_Nachtmann_Trust_and_Reputation_in_the_Internet_of_Things.pdf

20. Badcock C, Gambetta D. Trust: Making and Breaking Cooperative Relations. Br J Sociol. 1990;41: 128.

21. Misztal B. Trust in Modern Societies: The Search for the Bases of Social Order. John Wiley & Sons; 2013.

22. Daubert J, Wiesmaier A, Kikiras P. A view on privacy & trust in IoT. 2015 IEEE International Conference on Communication Workshop (ICCW). 2015. doi:10.1109/iccw.2015.7247581

23. Nils Gruschka &. Internet of Things Architecture (IoT-A) Project Deliverable D4.2 - Concepts and Solutions for Privacy and Security in the Resolution Infrastructure. 2012;1.0. Available: http://www.meet-iot.eu/deliverables-IOTA/D4_2.pdf

24. Gaillard E. Les systèmes informatiques fondés sur la confiance : un état de l'art [Internet]. Loria & Inria Grand Est; 2011 Nov p. 23. Available: https://hal.inria.fr/hal-00662479

25. Jøsang A, Ismail R. The Beta Reputation System. In Proceedings of the 15th Bled Electronic Commerce Conference. 2002. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.60.5461

26. Abdul-Rahman A, Hailes S. Supporting trust in virtual communities. Proceedings of the 33rd Annual Hawaii International Conference on System Sciences. doi:10.1109/hicss.2000.926814

27. Golbeck JA. Computing and applying trust in web-based social networks. University of Maryland at College Park. 2005.

28. Sabater J, Sierra C. Reputation and social network analysis in multi-agent systems. Proceedings of the first international joint conference on Autonomous agents and multiagent systems part 1 - AAMAS '02. 2002. doi:10.1145/544852.544854

29. Kamvar SD, Schlosser MT, Garcia-Molina H. The Eigentrust algorithm for reputation management in P2P networks. Proceedings of the twelfth international conference on World Wide Web - WWW '03. 2003. doi:10.1145/775152.775242

30. Marsh SP. Formalising trust as a computational concept. University of Stirling; 1994; Available: http://dspace.stir.ac.uk/handle/1893/2010

31. Gwadera R, Riahi M, Aberer K. Pattern-Wise Trust Assessment of Sensor Data. 2014 IEEE 15th International Conference on Mobile Data Management. 2014. doi:10.1109/mdm.2014.22

32. Sayan C, Hariri S, Ball G. Cyber Security Assistant: Design Overview. 2017 IEEE 2nd International Workshops on Foundations and Applications of Self* Systems (FAS*W). 2017. doi:10.1109/fas-w.2017.165

33. Chen D, Chang G, Sun D, Li J, Jia J, Wang X. TRM-IoT: A trust management model based on fuzzy reputation for internet of things. Computer Science and Information Systems. 2011;8: 1207–1228.

34. Ishmanov F, Kim SW, Nam SY. A robust trust establishment scheme for wireless sensor networks. Sensors . 2015;15: 7040–7061.

35. Journée internationale contre le bruit - Genève à la pointe de la lutte contre le bruit routier - 21.04.2016 | Bruit [Internet]. 21 Apr 2016 [cited 20 Feb 2018]. Available: http://ge.ch/bruit/actualites/journee-internationale-contre-le-bruit-geneve-la-pointe-de-la-lutte-contre-le-bruit-routier

36. Carr JJ, Brown JM. Introduction to Biomedical Equipment Technology. Prentice Hall; 1998.

37. Sharma AB, Golubchik L, Govindan R. Sensor faults: Detection methods and prevalence in real-world datasets. ACM Trans Sen Netw. 2010;6: 1–39.

Appendix

A. Lost measurements

As shown in Figure A.1, we are experiencing some measurements loss. This loss can last for days. The reason can be a interruption in the communication channel or a device malfunction.

Figure A.1 : Loss in sensor measurements

B. Accuracy tests (simulations)

A. Errors and Results

To validate our accuracy evaluation mechanism, we performed a number of tests on simulated datasets. The goal was to assess the effectiveness of our approach in detecting different types of errors. This section presents the results of our tests after introducing some known kinds of errors, other than “offset” error which was presented as an example in chapter IV. In each error type, we present used datasets (simulating sensors), results of estimations using out three estimation methods (Spatial covariance, Spatial average and Temporal correlation) and trust scores of each sensor’s measurements.

1. Offset error

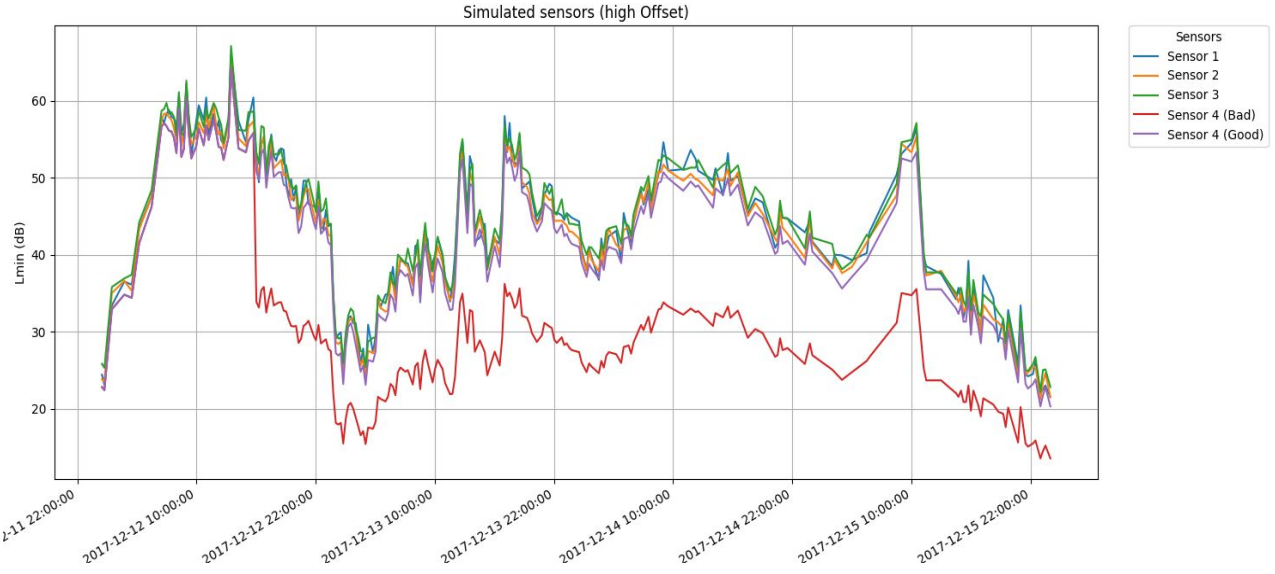

Figure B.1 : Datasets representing simulated sensors with one sensor having offset error

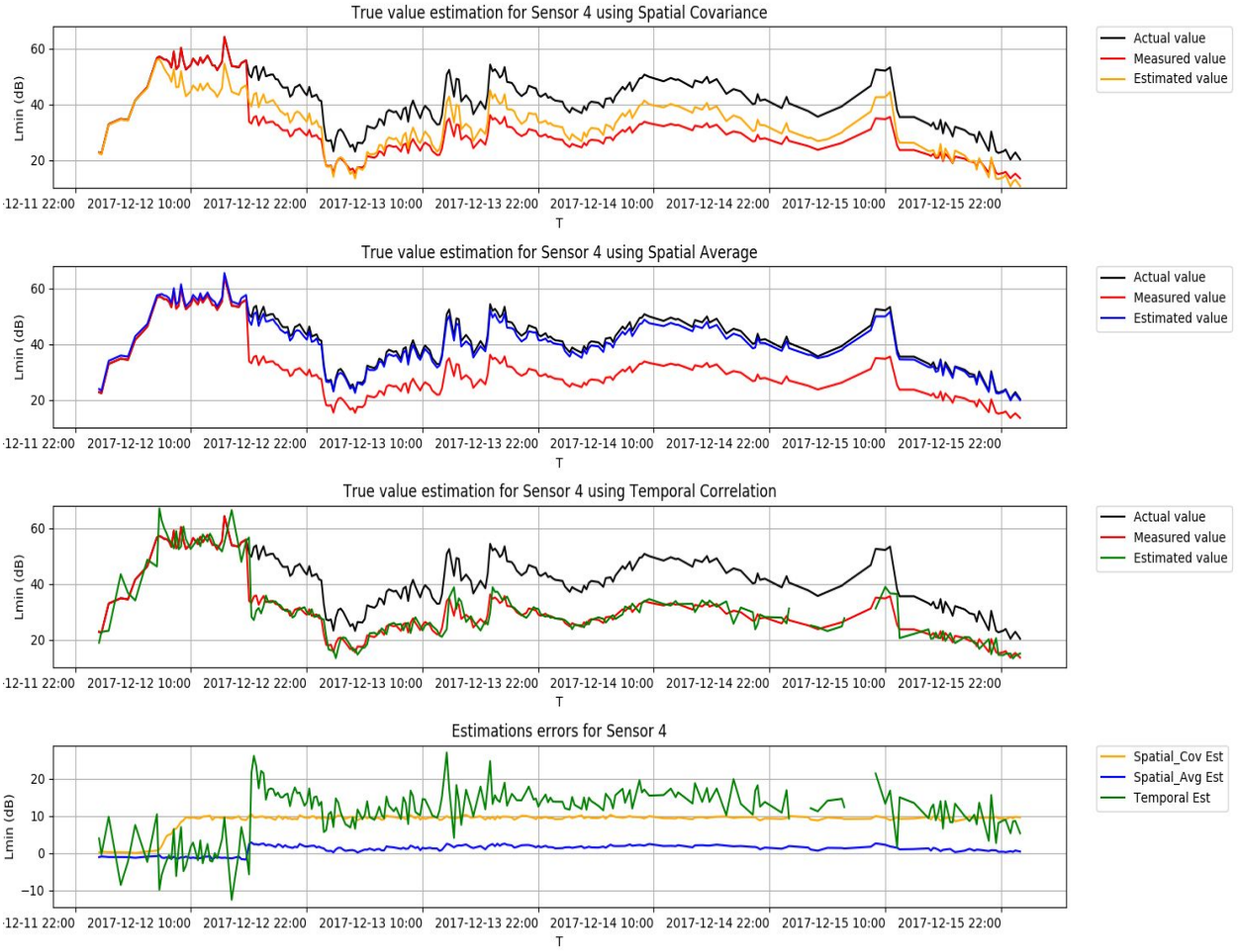

Figure B.2 : Estimations results and estimation errors of actual values of the erroneous sensor (sensor 4)

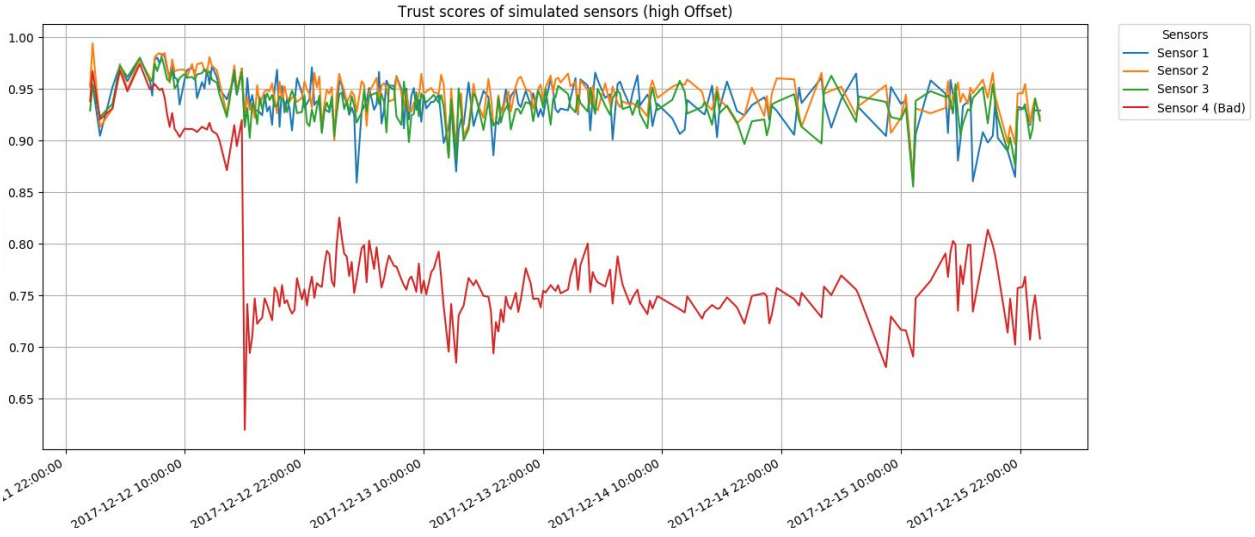

Figure B.3 : Trust scores of each sensor measurements

2. High variance (fluctuations) error

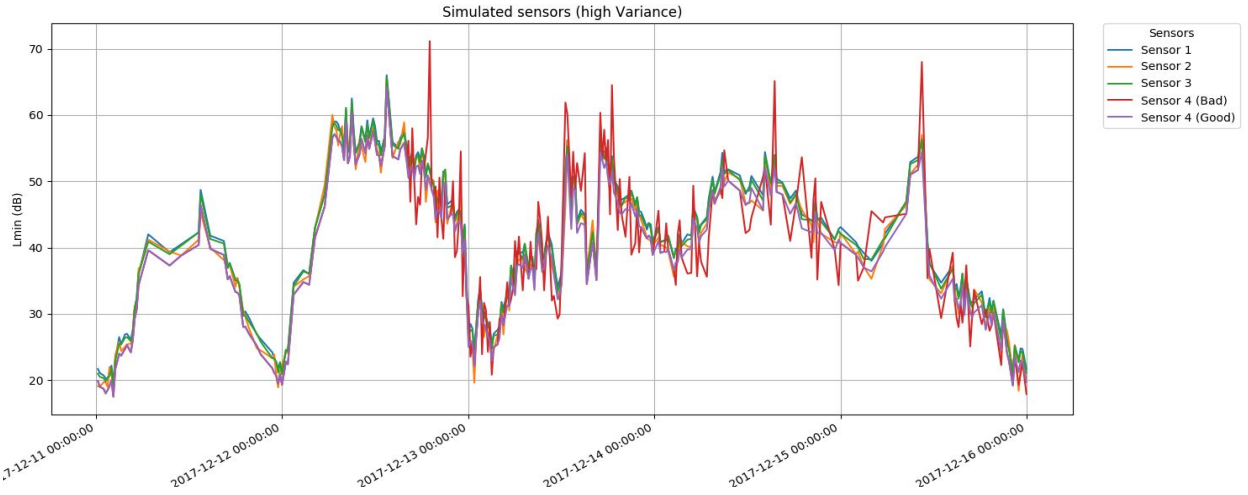

Figure B.4 : Datasets representing simulated sensors with one sensor having high variance error

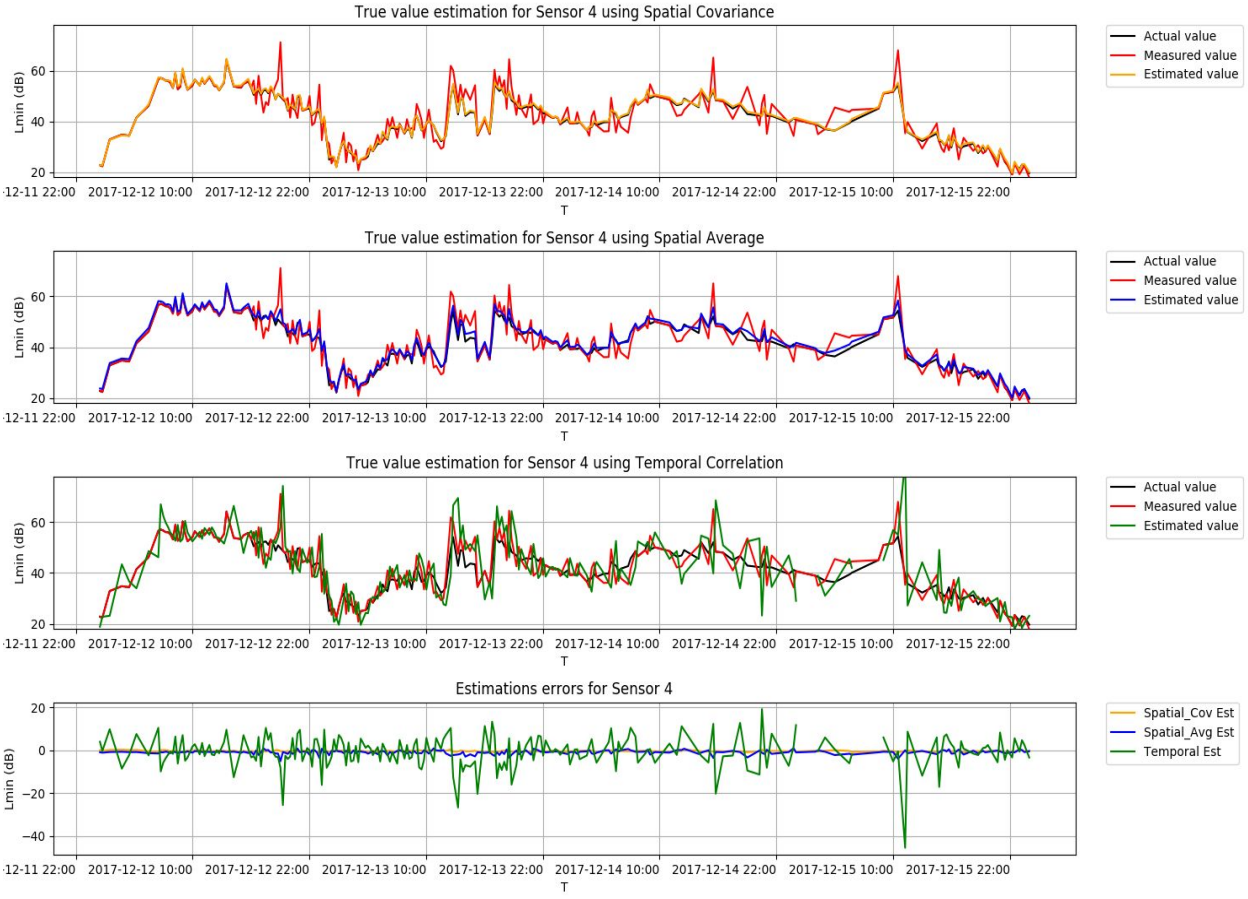

Figure B.5 : Estimations results and estimation errors of actual values of the erroneous sensor (sensor 4)

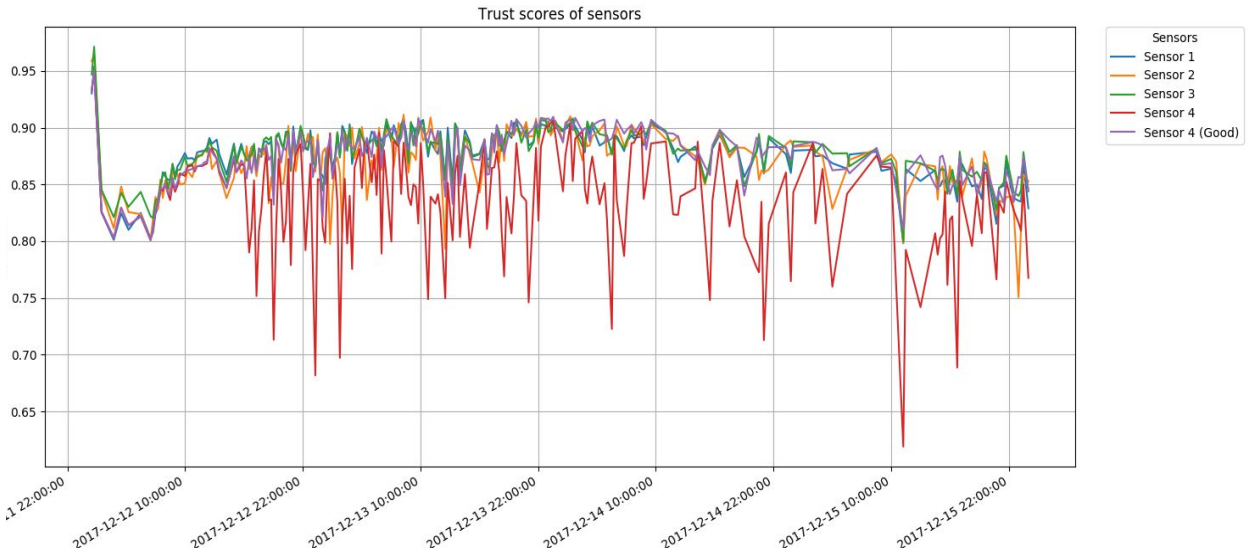

Figure B.6 : Trust scores of each sensor measurements

3. Offset + High variance error

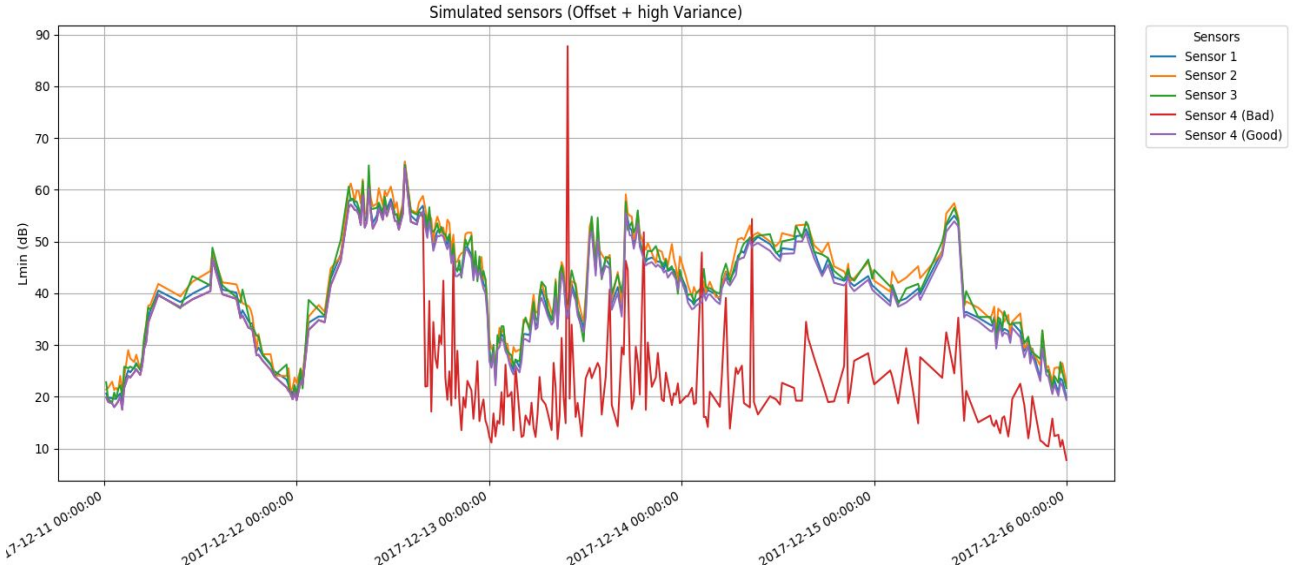

Figure B.7 : Datasets representing simulated sensors with one sensor having offset+high variance error

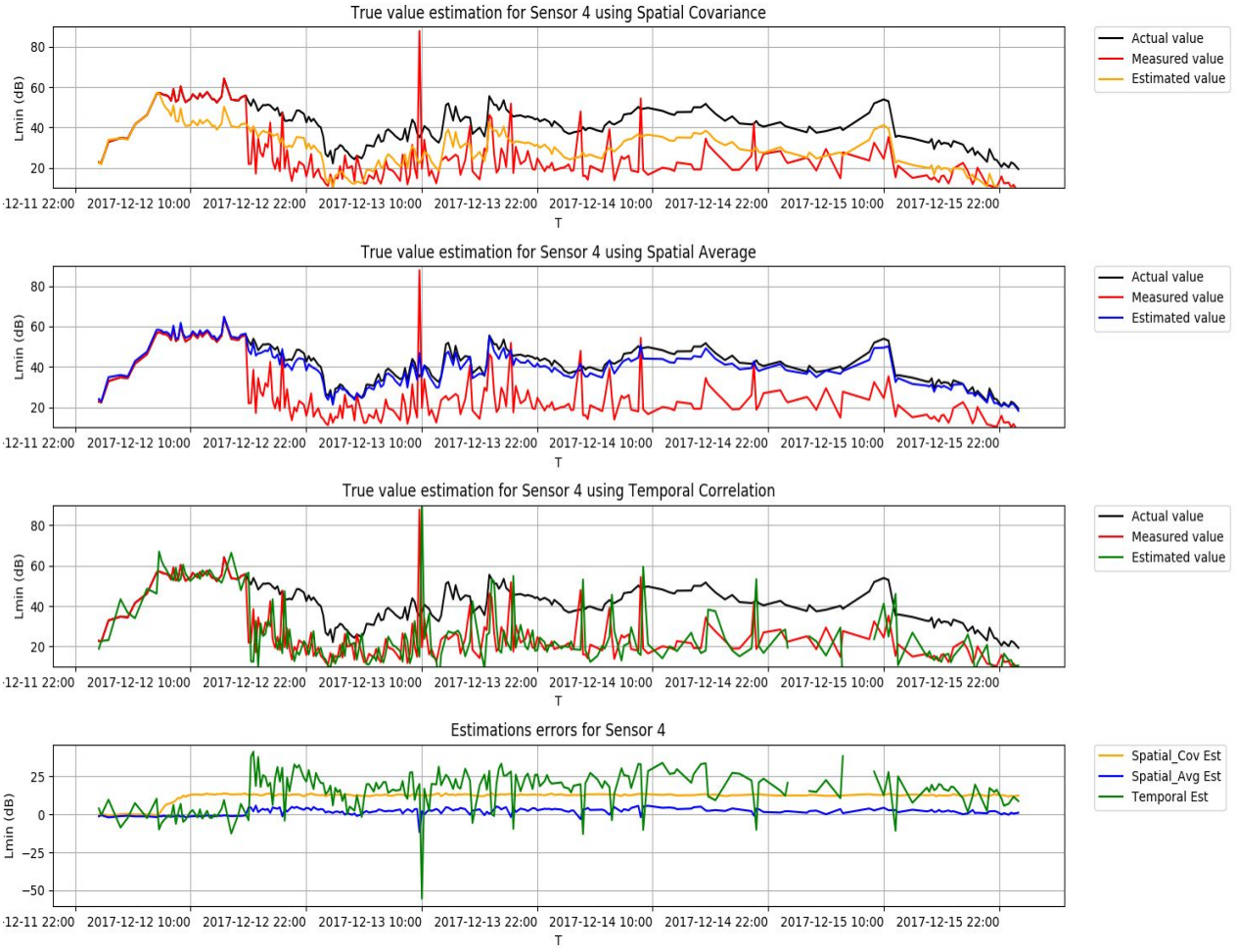

Figure B.8 : Estimations results and estimation errors of actual values of the erroneous sensor (sensor 4)

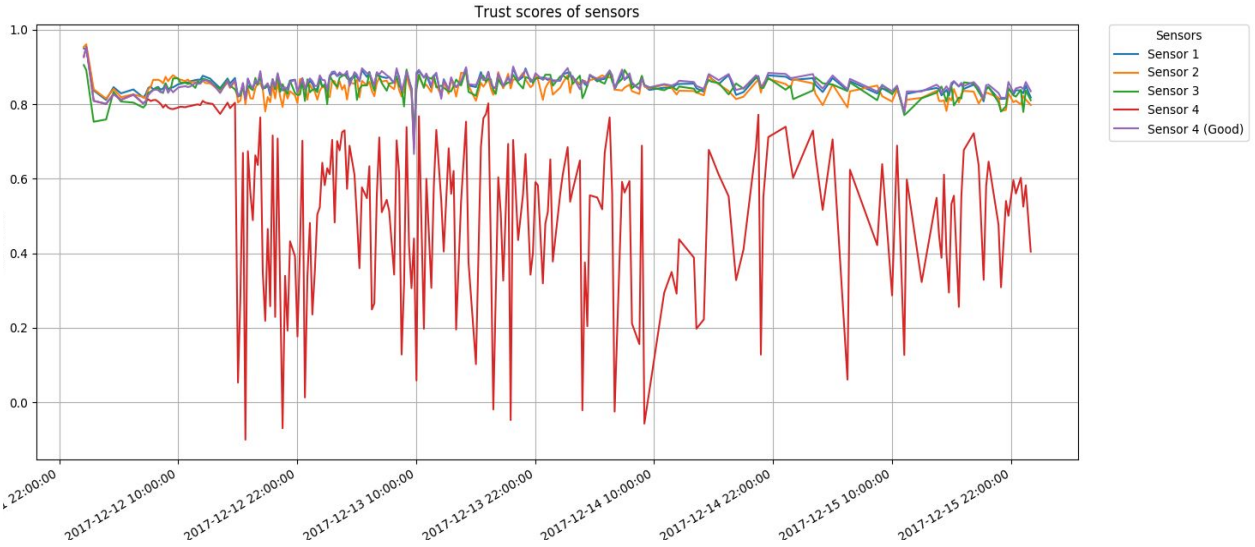

Figure B.9 : Trust scores of each sensor measurements

4. Constant values error

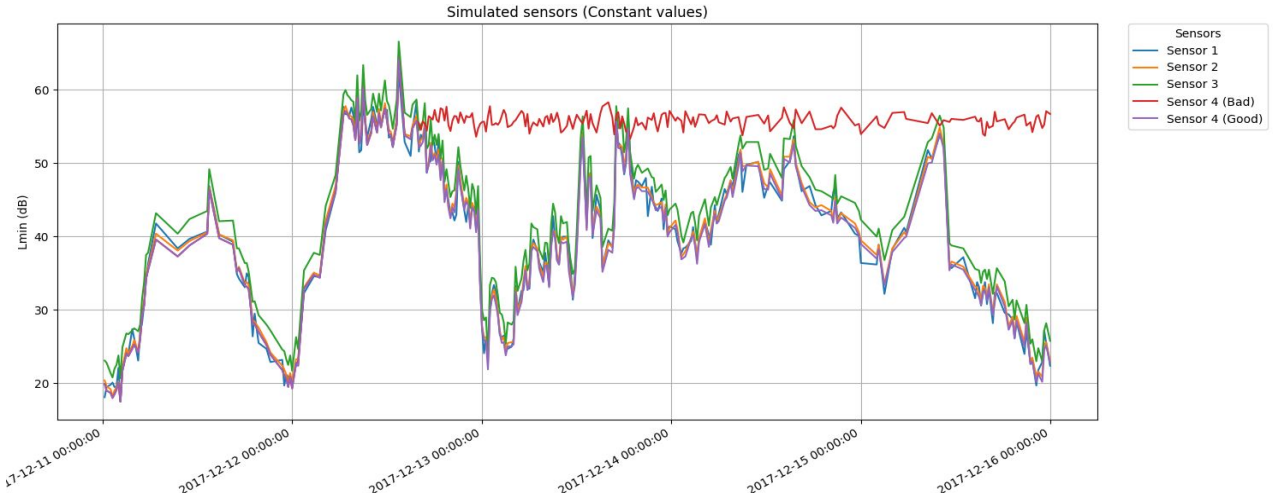

Figure B.10 : Datasets representing simulated sensors with one sensor having constant values error

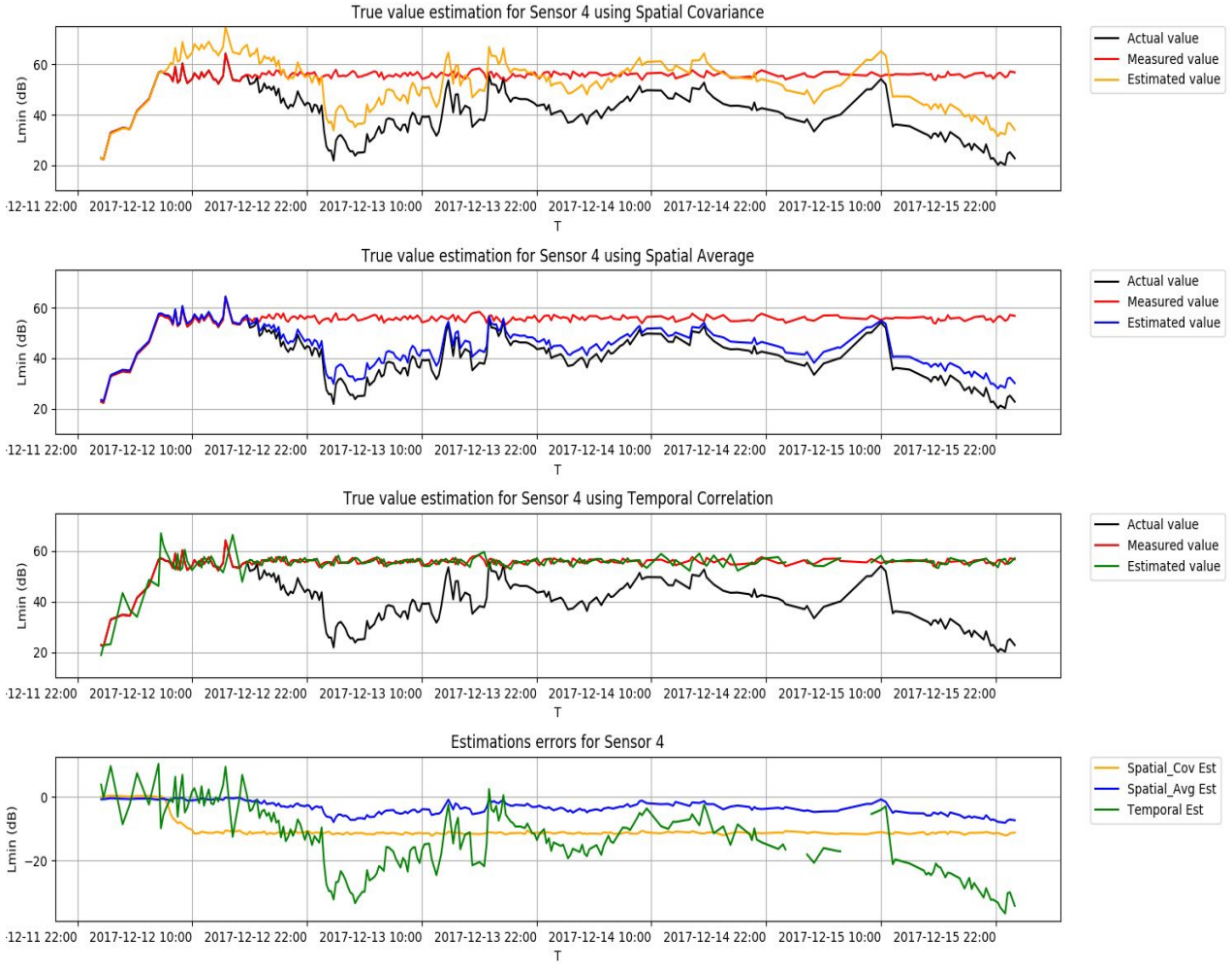

Figure B.11 : Estimations results and estimation errors of actual values of the erroneous sensor (sensor 4)

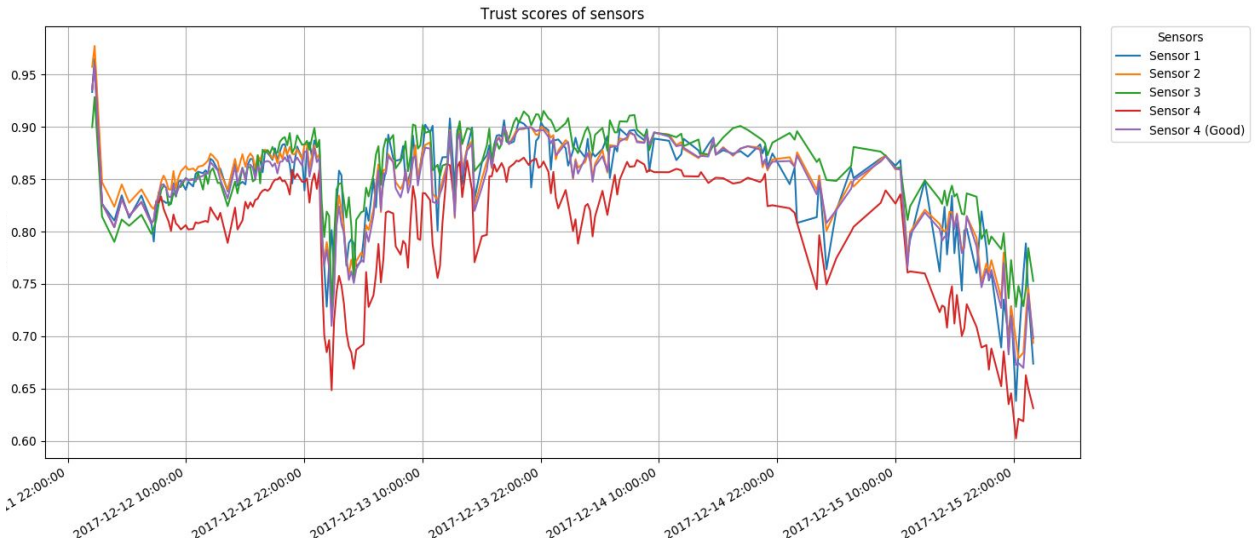

Figure B.12 : Trust scores of each sensor measurements

B. Choice of estimation weights

We notice that spatial covariance and spatial average are the better methods. Spatial average is immuned against offset errors. Spatial covariance is good at estimating the actual value in the presence of high variance errors. On the other hand, Temporal correlation have bad results. The problem is that our method uses the recent measurements to estimate the future one. This makes the temporal estimations follow the erroneous measurements.

To calculate the final estimations, we propose to compose the three estimation methods using these weights: $\alpha = 0.4$, $\beta = 0.4$ and $\gamma = 0.2$; where α, β and γ are the weights of the estimations of respectively the spatial covariance, spatial average and temporal correlation methods.

C. Trust and reputation tests on real data

To validate the overall effectiveness of our solution, we performed tests on different groups of real sensors. This section presents the trust and reputation estimation results of a group with good sensors and a group having a bad sensor misbehaving.

A. Tests on a good group

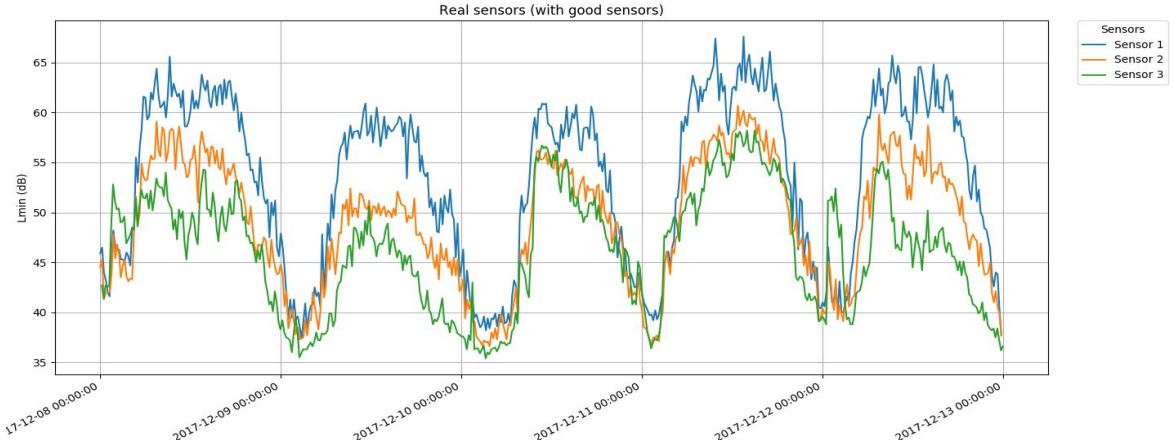

Figure C.1 : Datasets representing real sensors

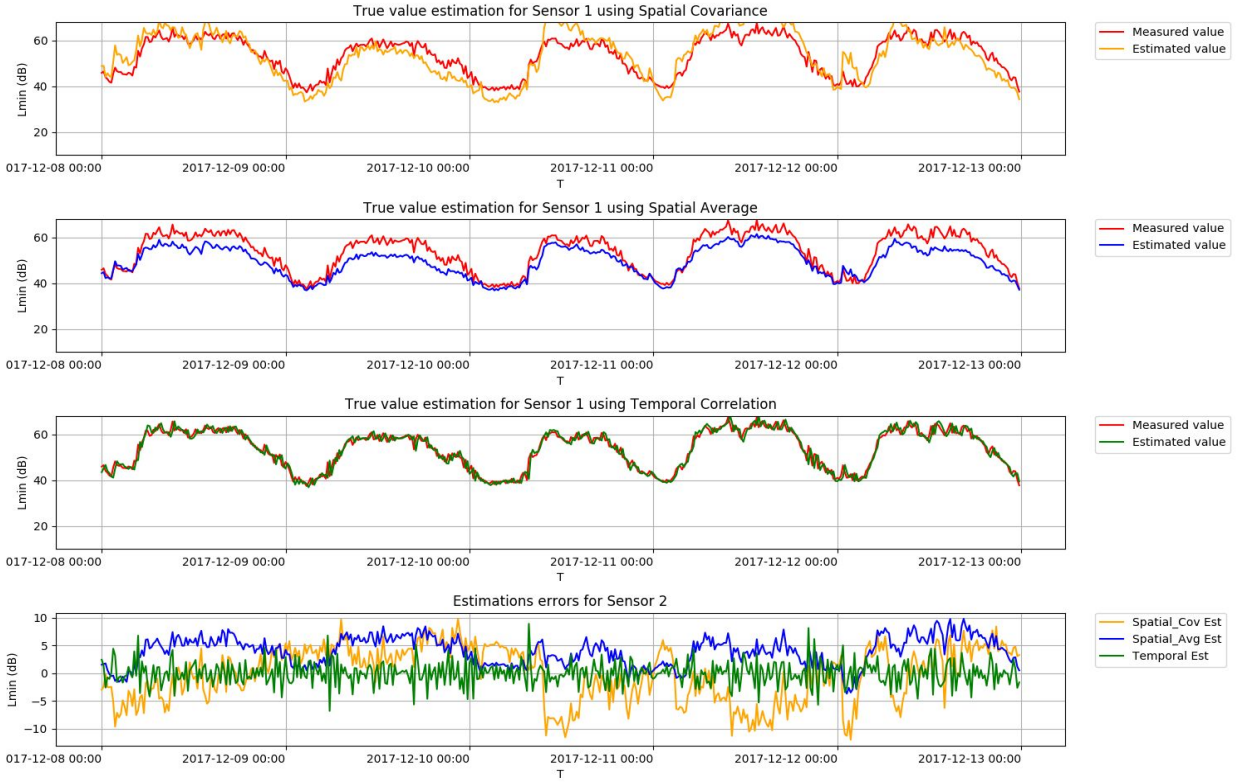

Figure C.2 : Estimations results and estimation errors of actual values of the one of the sensor

Estimations errors reflect the accuracy of the measurements according to each method because we are using measured values as the actual values.

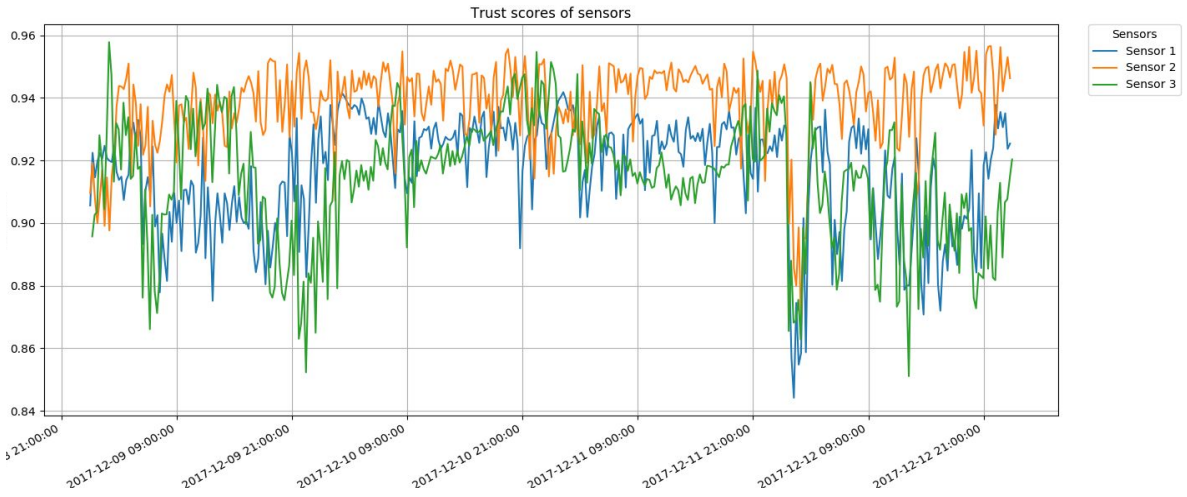

Figure C.3 : Trust scores of each sensor measurements

Here are the results of reputation scores calculation for this group:

Sensor ID	Sensor 1	Sensor 2	Sensor 3
Reputation score 1 (Average Trust)	0.911915	0.939715	0.916891
Reputation score 2 (Thresholds)	0.997925	0.997925	0.997912

Table C.1 : Reputation scores of sensors

Parameters thresholds used for Reputation score 2 are :

$$Accuracy = 0.8 \; ; \; Sensitivity = 0.1 \; ; \; Availability = 0.1 \; ; \; MRR = 0.45$$

B. Group with a bad sensor

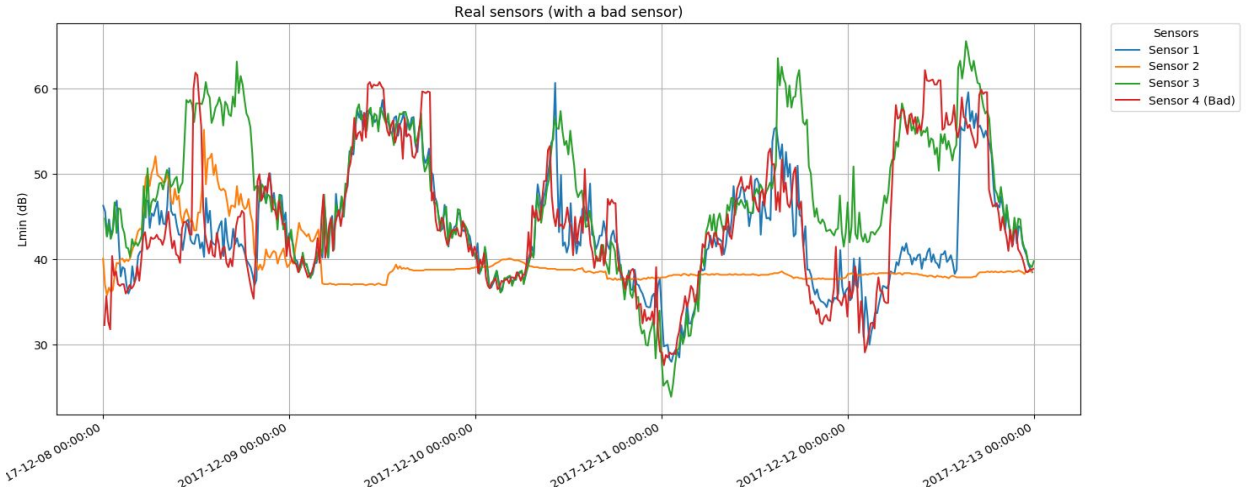

Figure C.4 : Datasets representing real sensors with a bad sensor

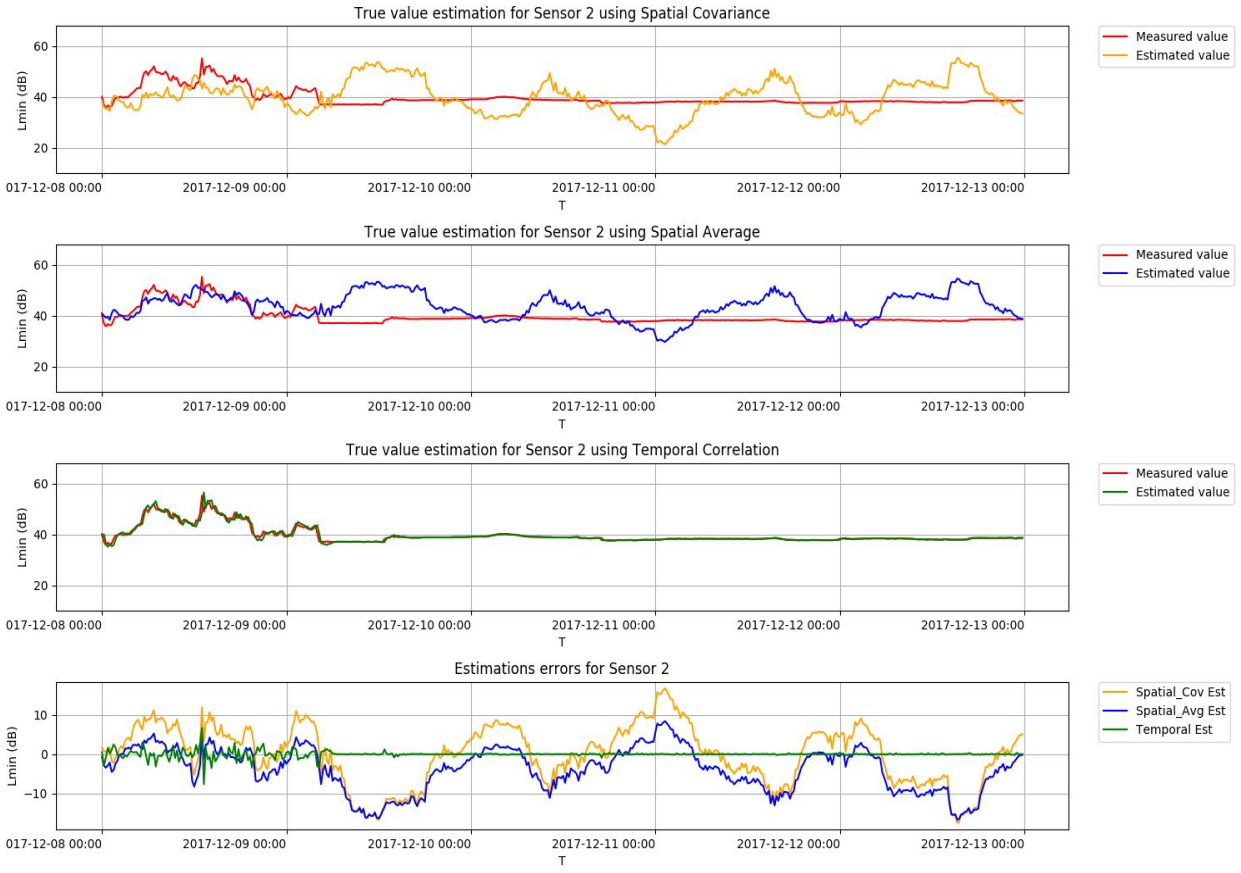

Figure C.5 : Estimations results and estimation errors of actual values of the bad sensor of the group

Estimations errors reflect the accuracy of the measurements according to each method because we are using measured values as the actual values.

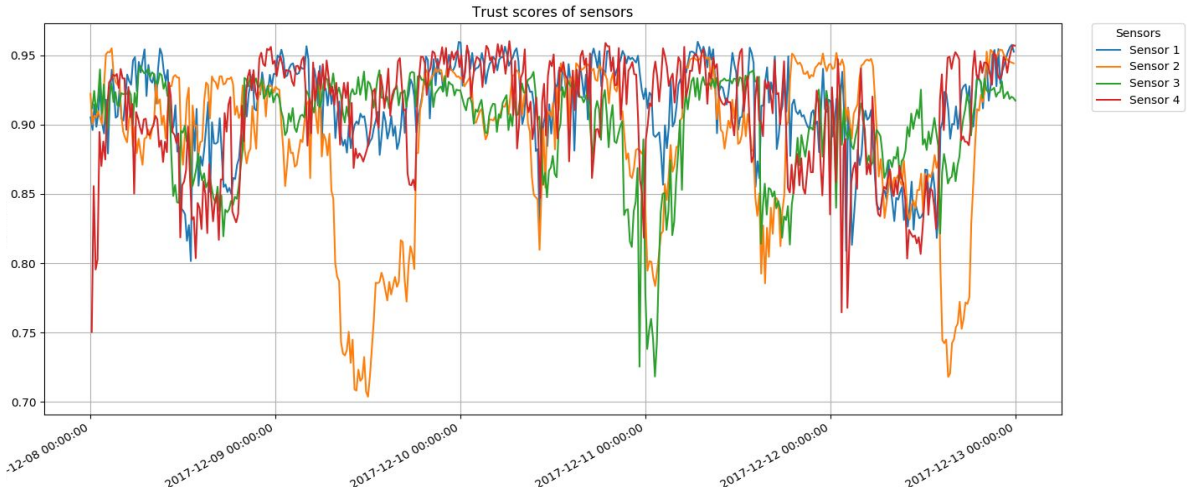

Figure C.6 : Trust scores of each sensor measurements

Here are the results of reputation scores calculation for this group:

Sensor ID	Sensor 1	Sensor 2	Sensor 3	Sensor 4
Reputation score 1 (Average Trust)	0.912695	0.887270	0.899297	0.908186
Reputation score 2 (Thresholds)	0.991684	0.835759	0.977131	0.983402

Table C.2 : Reputation scores of sensors

Parameters thresholds used for Reputation score 2 are :

$$Accuracy = 0.8 \; ; \; Sensitivity = 0.1 \; ; \; Availability = 0.1 \; ; \; RRR = 0.45$$

List of Figures

List of Tables