Sarah Haas, BSc.

Design and Implementation of a Hybrid Communication Approach for Smart Factories

MASTER'S THESIS

to achieve the university degree of

Diplom-Ingenieur

Master's degree programme: Telematics

submitted to

Graz University of Technology

Supervisor

Ass.Prof. Dipl.-Ing. Dr.techn. Christian Steger
Institute for Technical Informatics

Advisors

Ass.Prof. Dipl.-Ing. Dr.techn. Christian Steger
Dipl.-Ing. Holger Bock, Infineon Technologies Austria AG

Graz, September 2016

Abstract

Smart factories are the new concept which should enable companies to produce highly customized products in small batch sizes in an autonomously working factory. Smart factories use "Internet of Things" technologies to achieve this goal. They form a context-aware environment in which sensors and actuators, as well as simulation models for a physical and a virtual environment are combined. The combination enables the factory to produce autonomous and react to problems quickly. Such environments are very different from traditional manufacturing facilities and produce an enormous amount of data which needs to be processed and distributed trough the factory.

To distribute the data several communication scenarios need to be considered, and techniques need to be chosen which assure the ability to meet the real-time requirements and can also deal with the huge amount of data in terms of data transfer rate. Besides this, the security of the sent data and also the sending and receiving devices is crucial. It is necessary to prevent trespassers from being able to read data sent by any device and also write data to a device or machine. Therefore, several security concepts need to be found that fulfill the security requirements to protect the data traffic.

In this thesis, a hybrid communication approach using wired and wireless communication technologies is proposed that could be applied on the production floor of a smart factory environment. Security mechanisms are selected to protect the data from trespassers. The security mechanisms ensure several things: no unauthorized person can access the data, no manipulation was performed on the data during transmit, and the identity of the sender and receiver. A threat analysis is done to evaluate whether the selected security mechanisms are suitable for the proposed hybrid communication approach and fulfill the security requirements.

Kurzbeschreibung

Smart Factories sollen es ermöglichen, individualisierte Produkte in vollautomatisierten Produktionsstätten in sehr kleinen Mengen herzustellen. Smart Factories entspringen dem Industrie 4.0 Kontext und benutzen unter anderem Technologien, die für das Internet der Dinge entwickelt wurden. Mit diesen Technologien ist es möglich kontextsensitive Produktionsumgebungen zu schaffen, in denen Sensoren, Aktuatoren und auch Simulationsmodelle der physischen und virtuellen Umgebung verwendet werden um Produkte autonom zu produzieren und auf etwaige Fehler im Produktionszyklus automaisch reagieren zu können.

Durch die große Menge von Kommunikationspartnern überall in einem Smart Factory Umfeld ist die Menge an Daten wesentlich größer als in einer klassischen Fabrik. Es müssen verschiedenen Kommunikationskonzepte verwendet werden um eine robuste, schnelle Kommunikation für jeden Bereich der Fabrik gewährleisten zu können, die mit großen Datenmengen umgehen können und eine hohe Bandbreite bieten. Außerdem steht die Sicherheit der Daten im Vordergrund. Es muss gewährleistet werden, dass keine nicht-autorisierten Personen Zugriff auf die gesendeten Daten haben oder gar Daten in das System einspielen können. Um dies sicherzustellen müssen je nach Kommunikationsszenario verschiedene Sicherheitsmechanismen angewendet werden.

Ein hybrides Konzept für die Kommunikation zwischen Maschinen und Gateways wird in dieser Arbeit vorgestellt. Des Weiteren wird gezeigt wie man ein solches Konzept mit Sicherheitsmechanismen versehen kann, die gewährleisten, dass keine nicht-autorisierten Personen Zugriff auf die Daten haben, die Daten während der Übertragung nicht verändert wurden und die Identität eines Geräts sichergestellt wird. Eine Risikoanalyse der gewählten Sicherheitskonzepte wird durchgeführt um sicherzustellen, dass die genannten Sicherheitsanforderungen erfüllt sind.

AFFIDAVIT

Acknowledgments

First, I would like to thank my thesis advisor Ass.Prof. Dipl.-Ing. Dr.techn. Christian Steger of the Institute for Technical Informatics at Graz University of Technology for his guidance during the development of this thesis. Prof. Steger's office was always open whenever I needed his advice.

I would also like to thank my advisor at Infineon Technologies, Dipl.-Ing. Holger Bock, for his guidance and advises regarding technical questions. Furthermore, I want to thank Andreas Wallner, MSc for his participation and valuable input regarding all aspects of this thesis.

I would also like to acknowledge Dipl.-Ing. Thomas Ruprechter, Dr. Norbert Druml, Dr. Rainer Matischek and Dr. Josef Haid of Infineon Technologies for their discussions and helpful input.

I must thank my parents, Josef and Edeltrud, my brother David and my partner Thomas for providing me with unfailing support and continuous encouragement throughout my years of study. Finally, I want to thank my fellow students and good friends Marco, Jakob and Thomas. I am very grateful for their valuable help and assistance during my studies. This accomplishment would not have been possible without them. Thank you.

List of Figures

Contents

1 Introduction and Motivation

1.1 Motivation

In the last decades, production facilities in developed countries got rather expensive because of growing wages for employees and also increasing taxes. Due to cheap production in low-wage countries and inexpensive workforce, many companies are leaving Europe and the United States to produce elsewhere. Factories in low-wage countries are mostly not equipped with high-tech production automation machines. As a consequence of very cheap employees, the factories there do not need high-tech production automation machines because employees can do most of the work and still cost less than the machines.

In the last years, the industry's wish to produce customizable products increased rapidly as customers more and more ask for individual products. This is difficult to achieve in factories located in low-wage countries because they were built for mass production. Therefore, the University of Stuttgart in Germany invented the concept of "Smart Factories" [1]. Research in this area was initiated by the German government [2] as a strategy to convince companies to keep their production facilities in central Europe. These factories can deal with the requirements of customizable products and therefore, small batch sizes using state-of-the-art computing technologies which are used in the "Internet of Things" context. Such environments should stop companies leaving Europe and the United States due to the need for less and well-educated workforce while being able to produce highly customized products.

1.2 Introduction

The idea of smart factories is to create a manufacturing environment which works highly autonomous and can deal with problems in the production process automatically. Figure 1.1 shows a possible concept for the realization of a smart factory where machines communicate to products and other machines, and humans can track machine states or production flows. To achieve the goal of machines automatically adapting to the changes in the production environment, reliable information transition and direct execution of decisions is necessary. The factory is designed

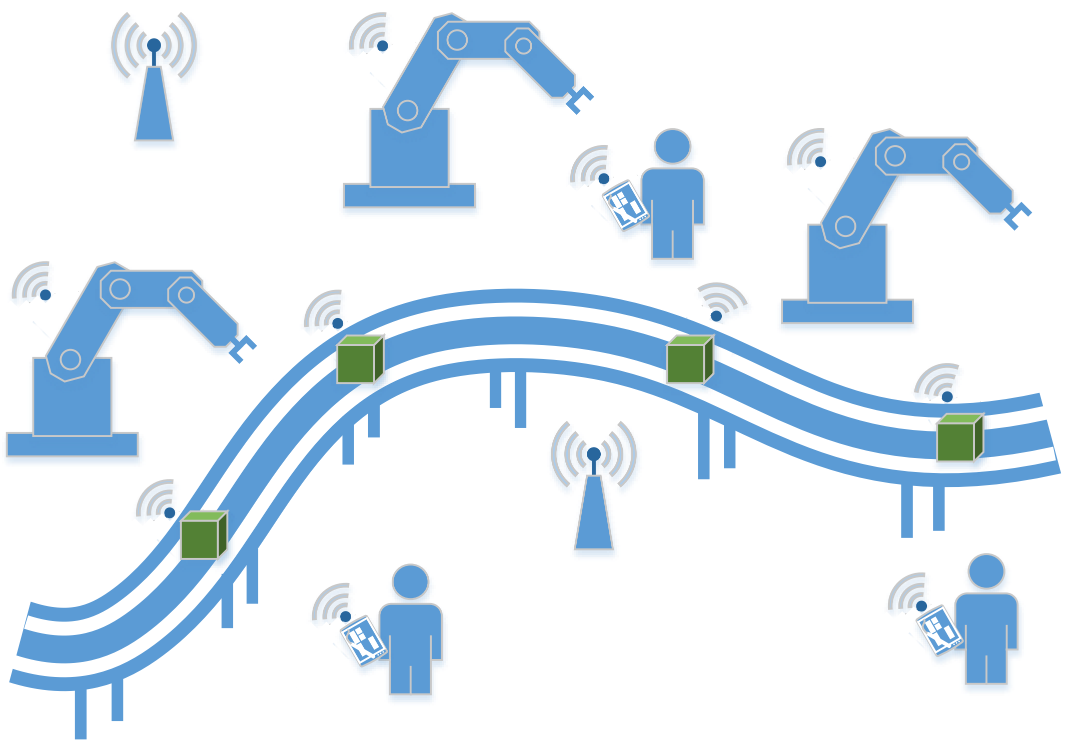

Figure 1.1: Conceptual design of a smart factory. No central infrastrucuture for controlling is needed anymore as machines automatically adapt to changing conditions. Employees can check machine states and products anywhere at any time.

to help employees and machines to execute their tasks. To support the employees, the factory needs context-aware applications which provide information about the physical environment in the factory. Such context-aware applications, for example, are all kinds of sensors distributed through the factory. The factory not only relies on information from the physical but also from the virtual world such as simulation models or electronic documents. The combination of the physical and virtual world enables the factory to produce highly autonomous and deal with all kinds of problems arising in the production process such as overheating of a machine.

In smart factories, a huge amount of data is available anywhere anytime. This is necessary to guarantee fast production processes and the ability to react to problems in time to prevent major errors in the production process or guarantee the safety of employees on the production floor. Communication approaches, therefore, are an important part of smart factories as they need to fulfill the requirements of reliable and fast transition of an enormous amount of data in various areas in a smart factory as well as transmission to a data storage that is accessible through the Internet. As in each area various communication partners are working, different kinds of communication scenarios occur. Some obvious communication scenarios which will definitely arise in a smart factory can be seen in the following list:

- Human-to-Machine communication
- Machine-to-Machine communication
- Controller-to-Machine communication
- Device-to-Controller communication

The communication scenarios discussed here are always bidirectional as the partners communicate with each other in both ways.

Human-to-Machine communication will be necessary when humans interact with machines such as changing a machine configuration, the machine alerting humans to take some specific action or just providing information to humans. This kind of communication can be achieved by providing displays and input devices with user interfaces. Input devices can be, for example, smartphones, keyboards, tablets, or bar codes.

Machine-to-Machine communication might occur when machines need to work together to create a product. The communication there needs to work as fast as possible. Otherwise, the production process would be slowed down, or a machine would modify a product in the wrong way. For this kind of communication, there are various ways to do it. Several wired and wireless technologies could be used here but with each technology, other issues regarding timing behavior or security arise. Different technologies and solutions regarding security and reliable communication will be discussed in Chapter 2 as one main topic of this thesis will focus on Machine-to-Machine communication.

Controller-to-Machine communication is very important as controllers, in this case, are the devices which process the data from the physical and virtual world and react to problems or distribute new tasks. This kind of information needs to be sent with absolute reliability to avoid e.g. errors by machines because of a wrong configuration. Furthermore, the data sent between controllers and machines is highly confidential and needs to be secured appropriately. Controllers are also some kind of gateway to the data storage of the factory. Machines send their data to the controller and the controller distributes the data. Controllers also send data to the machines to change configurations or the production flow. Therefore, this kind of communication needs to be able to deal with massive amounts of data. In this case, also wireless or wired technologies, or a combination of both could be used. This is also a main topic of this thesis and will be addressed in later chapters.

Device-to-Controller communication is needed to send the data from sensors to the controllers so that the controller can react on changes in the real world environment. This is also time critical information as products might be produced wrong if the information is not received in time and the controller cannot change the production task. As there are hundreds of sensors and they are mostly distributed somewhere in the environment, it is more meaningful to communicate using wireless technologies.

The discussed scenarios might arise within a smart factory, but there are further possibilities of interactions such as employees that want to check the production flow of a facility from a foreign country over the Internet. Availability of the data everywhere in the world at any time is another idea of smart factories. Furthermore, the collected data needs to be processed to extract and process information. This could be done by using, for example, cloud computing. The processed information needs to be stored to make it available to every machine, controller or employee that needs the information. A cloud storage could, for example, be used to store the data from any production facility of a company in one data storage. Cloud computing and cloud storage are two concepts that are recommended for smart factories by experts. This will be discussed in Chapter 2 but not implemented as it would exceed the scope of this thesis.

All of the data sent in any of the communication scenarios can be distributed using wired or wireless technologies and furthermore, it needs to be available from everywhere in the world which makes it necessary that the smart factory is connected to the Internet. Due to this, the need for security arises as the data produced by the factory is usually confidential. Companies can incur massive losses due to trespassers which have access to confidential data. Furthermore, such people could manipulate data and send it to the controllers or machines which might lead to failures on machines or errors in the production process.

To avoid this, security mechanisms need to be found which prevent trespassers from reading or writing data from or to the machines, controllers or other devices. Finding security mechanisms suitable for some of the given communication scenarios will be one main topic of this thesis.

A hybrid communication approach consisting of wireless and wired technologies will be implemented. More precisely, the concepts Machine-to-Controller and Machine-to-Machine will be implemented. For data transfer, communication protocols will be used that can deal with the challenges of IoT and smart factory environments. Different kinds of security mechanisms depending on the communication approach will be evaluated. The communication overhead due to the security mechanisms will also be discussed. The main goal of the security concepts is to ensure confidentiality, integrity and authentication of the wireless communication approach, and integrity and authentication of the wired approach. The wireless communication will be used to send sensitive data while the wired communication will be used for non-confidential data.

This thesis is developed as part of the *Semi4.0* project [1] and will be divided into several parts. Chapter 2 contains related work regarding smart factories. Furthermore, communication technologies usable for smart factory concepts, state-of-the-art technologies used in current factories and also combinations of such technologies are discussed. Communication protocols suitable for smart factory concepts as well as security mechanisms in the context of IoT and smart production facilities and security technologies are shown in Chapter 2.

In Chapter 3 the general architecture of the hybrid communication approach will be shown. The chosen communication protocols and technologies will be discussed in detail. Furthermore, the hardware and software used to implement a simple demonstrator of the concept will be discussed in Chapter 4.

A Threat analysis of the different communication scenarios will be done in Chapter 5. The threat analysis consists of several parts as two different communication technologies are used in this thesis. Furthermore, an evaluation of the package overhead due to the security mechanisms will be shown.

In Chapter 6 the technical limitations and future work will be discussed. Furthermore, the thesis will be concluded, and a recapitulation of the whole work will be done in this chapter.

[1] *Semi4.0* is an EU project funded by ECSEL.

2 Related Work

In this section, the related work of this thesis will be discussed. First, the general concepts of smart factories will be shown. Then, wired and wireless communication technologies that are currently used in industrial environments or might be suitable for smart factories, as well as protocols that could be used in this context will be discussed. Finally, security concepts fulfilling different security requirements will be described.

2.1 Smart Factories

The first step towards the Internet of Things(IoT) and therefore, towards the concept of smart factories was made in 1991 by Mark Weiser [3] who coined the term *ubiquitous computing* for the next big revolution in computing:

> *Ubiquitous computing names the third wave in computing, just now beginning. First were mainframes, each shared by lots of people. Now we are in the personal computing era, person and machine staring uneasily at each other across the desktop. Next comes ubiquitous computing, or the age of calm technology, when technology recedes into the background of our lives.*

Even if technology and industry are still a distance away from this vision, it was essential for the development of the Internet of Things. Atzori et al. [4] define the Internet of Things as a world-wide network of interconnected ”things” or objects that are uniquely addressable and use standard protocols and technologies for communication. Furthermore, the authors discuss the main technologies needed to enable the Internet of Things in the real world. Communication technologies are some of the most important technologies in the field of Internet of Things. As the size, weight and energy consumption of current radios decreases, wireless technologies for communication are widely used in the Internet of Things context.

Atzori et al. [4] also state that RFID technologies will be used widely as they provide tags that can be used passively or actively. RFID is a shortcut for Radio Frequency Identification and is a technology based on readers and tags where the tags are powered by the electromagnetic field created by the reader. The technology is mostly used to identify objects.

Passive tags do not have a battery on board which means that they cannot power their radio by themselves but need the RFID reader to provide the energy to do so which limits their communication range to some centimeters.

Active RFID tags are powered by batteries which enables them to power the radio by themselves which allows them to communicate over much higher distances than passive tags.

Furthermore, the authors discuss the need for wireless sensor networks for the Internet of Things as they do not need a reader like RFID but communicate using, for example, technologies based on the IEEE 802.15.4 standard. IEEE 802.15.4 is used for wireless personal area network describing the lower layers of the OSI model which is designed to consume very low power and provide low complexity. These properties make it suitable for wireless sensor networks. The authors claim that those technologies will be widely used for communication in the Internet of Things.

Atzori et al. [4], furthermore, give several domains where the Internet of Things concept could be applied. The main applications are the transportation and logistics domain, health care domain, personal and social data domain, and the smart environment domain.

One subdomain of the previously mentionend smart environment are smart factories. Zuehlke [1], in his paper, defines his vision of a smart factory in the context of *Industry 4.0* as:

> *Adopting the basic principles of the internet-of-things we should talk about a factory-of-things as a vision for our future factories. The factory-of-things will be composed of smart objects which interact based on semantic services. There will be no hierarchy in the traditional sense instead the objects will self-organize to fulfill a certain task.*

Furthermore, he states that it is necessary to think of several aspects when trying to build smart factories. Some of the aspects are:

- Developing technologies for humans as deserted factories are an unrealistic scenario.
- Reducing complexity due to the usage of self-organization on system level or decentralized hierarchies that consist of self-adapting modules.
- Allow reuse of components and reduction of planning effort by applying standards to all aspects of automation.

Industry 4.0 [5] is the name for the fourth industrial revolution in Germany which is the integration of IoT and interconnected objects in manufacturing. A similar approach is done by the US government and called *Smart Manufacturing*. Kang et al. [6] describe the differences and similarities between the two approaches. Smart Manufacturing focuses on technology and also on strategy such as cyber-physical

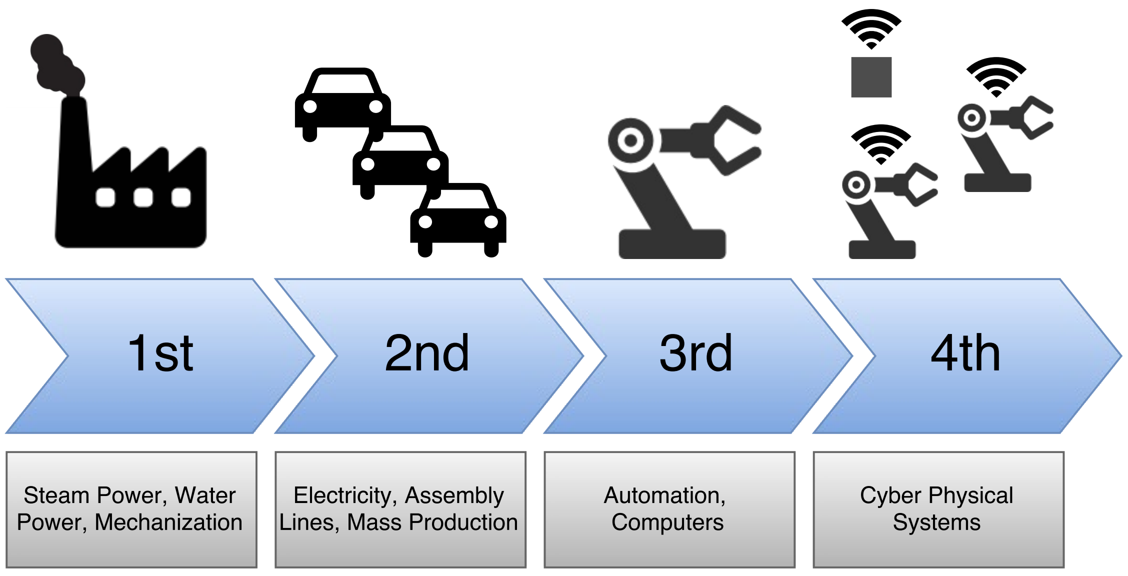
1st
2nd
3rd
4th
Steam Power, Water Power, Mechanization
Electricity, Assembly Lines, Mass Production
Automation, Computers
Cyber Physical Systems

Figure 2.1: Innovations that enabled the fourth industrial revolutions.

systems(CPS), IoT, big data/data analytics, cloud computing, sensors, smart energy and additive manufacturing whereas Industry 4.0 focuses mainly on technology. In the paper, they give definitions for all different types of technologies either Industry 4.0 or Smart Manufacturing address.

Lee [7], in his paper, gives a detailed description of a CPS and shows a concept to implement such a system. He defines CPS as:

> *[...] a complex engineering system that integrates physical, computation and networking, and communication processes. CPS can be illustrated as a physical device, object, equipment that is translated into cyberspace as a virtual model. With networking capabilities, the virtual model can monitor and control its physical aspect, while the physical aspect sends data to update its virtual model.*

Figure 2.2 shows the 5-stage architecture developed by Lee [7] which could be used to implement a cyber-physical system.
Level 1 is the connection level which describes the connections between machines, sensor or any other objects in the factory and the acquisition of data from those components.
Level 2 is the conversion level where the huge amount of data such as machine states or production flow information from the components is processed and converted to meaningful information that is readable by humans. Signal processing, feature extraction or predictive analytics regarding machine health are performed on the data to enable monitoring of the machines.

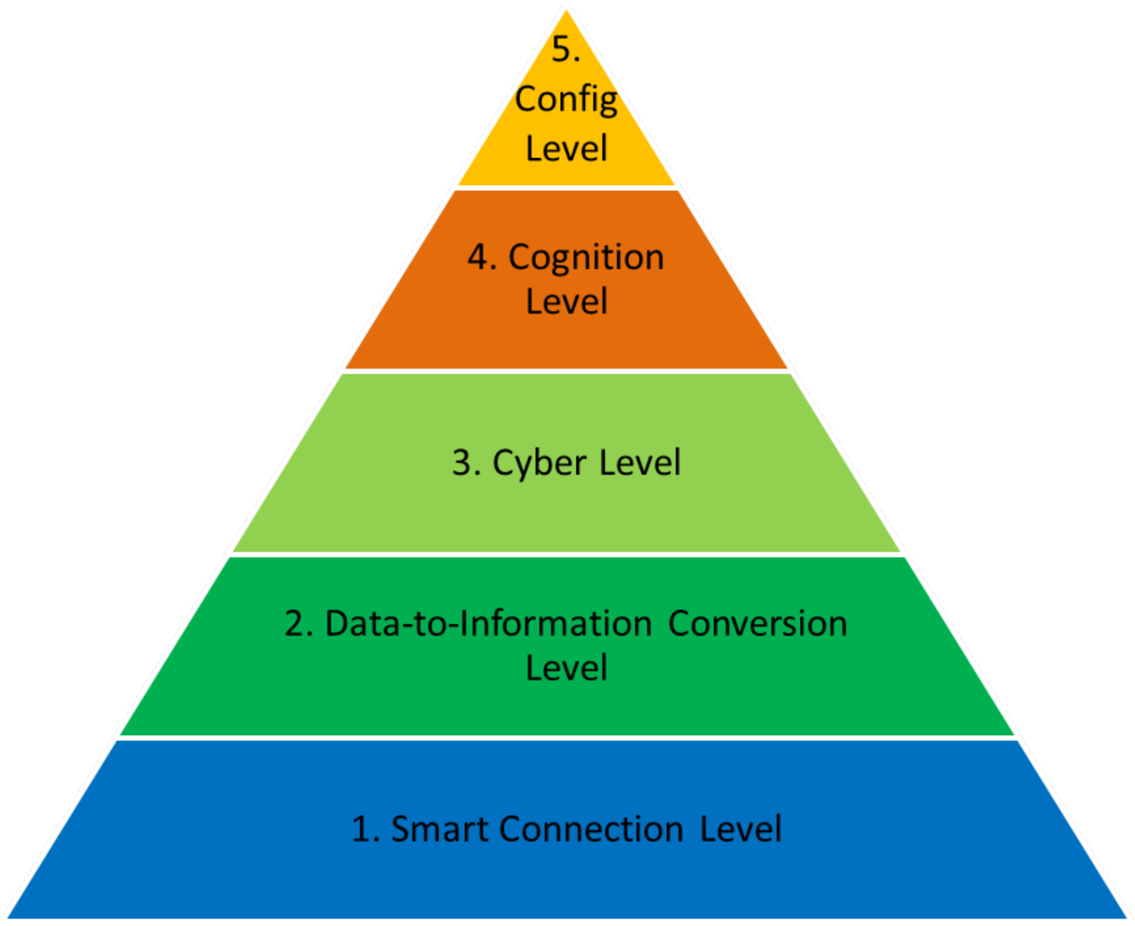
5. Config Level
4. Cognition Level
3. Cyber Level
2. Data-to-Information Conversion Level
1. Smart Connection Level

Figure 2.2: A 5-stage architecture for cyber-physical systems.

In level 3, the cyber level, all information regarding sensor values or machine outputs are processed. Information sharing between machines is also performed in this level. Furthermore, single machine performances are compared with the performances of the fleet to improve production speed or quality, or predict future behavior. This enables the factory to optimize the production flow.

Level 4 is the cognition level which provides reasoning information for experts to make decisions regarding the production process. Info graphics are generated here to present the data in an understandable way to the expert users.

Finally, level 5 is the configuration level where actions are taken to make machines self-configure, self-adapt and self-maintain.

In his paper, Lee [7] furthermore addresses the advantage of CPS to be able to manage and present data to several decision makers by using, for example, cloud computing. Cloud computing enables ubiquitous, on-demand access to a large amount of servers, storages, computing devices and services. By using cloud computing, the data represented to users and devices was already processed such that just necessary information is left and the users and devices do not have to further pre-process the data to extract useful information. Furthermore, the data can be stored in the cloud which means that factories do not need in house data storages anymore. Therefore, factories do not need to maintain the storage servers and host a network infrastructure.

As smart factories provide a lot of information and interconnect a large amount of objects, there are several risks that can occur. In his paper, Hertel [8] gives some issues that can arise in the context of Industry 4.0. He describes a model to identify threats, analyze which parts of the system are affected by which threat and define the security mechanisms needed to avoid the identified threats. The author further elaborates on different forms of security risks such as targeted attacks, human failure, technical failure or act of nature and analyzes the possibly affected parts of the system and gives suggestions on how to overcome each risk. The paper sums up all possible security risks that could arise in such a system. The security issues in smart factories will be further described in Section 2.3.

2.2 Communication Approaches

In this section, different communication approaches that could be applied in a smart factory will be discussed. As there are wired, wireless and also hybrid approaches this section will contain related work of all three topics. Furthermore, different protocols suitable for smart manufacturing will be discussed and compared.

2.2.1 Machine-To-Machine Communication

In the context of smart factories the catchphrase *M2M* occurs very often. M2M is an abbreviation for machine-to-machine and is a communication principle defined in a specification by the ETSI [9]. They define M2M communication as:

> *[...] the communication between two or more entities that do not necessarily need any direct human intervention. M2M services intend to automate decision and communication processes.*

M2M is the basis for concepts such as IoT or sensor networks and is therefore also suitable for smart manufacturing. In their paper, Weyrich et al. [10] give an overview of wireless M2M technologies which might be useful in smart factories.

They also state the disadvantages of wired M2M communication compared to wireless M2M communication. Wired networks are very static and its very costly to change the wiring scheme whereas for wireless schemes it is easy to add, remove or exchange devices. Furthermore, wired networks need a lot of planning before they can be set up and it is very costly to set them up. Wireless networks are easy to set up and are rather cheap because they use technologies such as RFID or WLAN.

On the other hand, the authors state that wired networks provide much better robustness and availability than wireless network technologies. Weyrich et al. [10] also mentioned that wired networks more easily provide real-time communication which is very important in production environments. However, they outline the biggest advantage of wireless M2M as ubiquity which means that every product's status and position in the facility is known all the time.

To enable communication, protocols are needed for M2M. The authors discuss several possibilities for communication protocols which provide low memory consumption and can deal with limited bandwidth. They discuss IPv6, MQTT, CoAP and DDS as currently suitable protocols for wireless M2M but state that there are many more which might be used in this context. These protocols and some more will be discussed later in this section.

Finally, the authors give an overview of current state-of-the-art wireless technologies and compare properties such as range or, throughput with each other.

The following list gives the most important discussed wireless technologies in the paper of Weyrich et al. [10]. Advantages and disadvantages will also be stated in this list as well as the possibility to guarantee real-time communication with them:

- **LTE:** In his paper, Haryantho [11] describes LTE as a communication standard which enables several mobile phones or data terminals to communicate all over the world with each other using air as a medium and providing a very high bandwidth. LTE is based on GSM and UMTS and uses an IP-based network architecture to pass the data to cell towers. The communication range of LTE is up to 10 kilometers which around 100 times higher than the range of other standards. This enables LTE to communicate also to devices outside of the factory such as sensors distributed somewhere outdoors.

  The throughput of LTE is about 150 Mbit/s. The installation of a LTE network is rather expensive, as most sensors or machines do not have LTE modules integrated and therefore, need to be added later. Furthermore, the infrastructure on the provider side is huge.

- **WLAN:** The IEEE 802.11 standard WLAN is discussed by Rech [12] as a wireless communication technology that is able to transmit up to 600 Mbit/s with IEEE 802.11n and a range of up to 100 meters either on the 2.4GHz or the 5 GHz frequency band. It needs routers and access points to be able to transmit their data to other devices. Nowadays, every mobile device is able to communicate using WLAN which makes this technology very interesting.

  WLAN is based on the wired technology ethernet and uses a mechanism similar to ethernet which checks if anyone is currently sending data. The mechanism is named CSMA/CA and before sending data always checks if the medium, in this case, the air is free. CSMA is an abbreviation for carrier sense multiple access and checks if currently another transmission is going on on the medium as the medium is shared between many parties. CA is a shortcut for collision avoidance and creates a random delay if the medium is detected as being occupied. After this delay the medium is checked again and the data is transmitted if its free. This mechanism avoids that several parties try to send at the same time as the delay is randomized.

  In contrast to LTE, WLAN does not need a centralized communication infrastructure and provides a higher bandwidth but a much lower range. WLAN is a suitable technology for smart factories where there is a need to transmit a high amount of data.

  There is an extension of WLAN named Industrial WLAN. Industrial WLAN (IWLAN) is actually very similar to WLAN but it provides much more robustness, low latency and enables a soft real-time behavior. Soft real-time means that the result's usefulness decreases after the deadline and the quality of the system decreases but the system is not destroyed or humans are not harmed when the deadline is exceeded.

  Cena et al. [13] evaluate IWLAN regarding the response times and therefore, the real-time ability. The authors' experiments show that when there is low traffic in the network the response times are bounded, so it could provide real-time. If there is more traffic on the network the mechanisms of IWLAN cannot produce a deterministic behavior in any case. Most of the time responses occur within a specific latency but sometimes its not predictable. This makes IWLAN, at least for light traffic, able to provide real-time. Anyhow, IWLAN is very expensive to set up in a factory as they need special access points.

In this master's thesis, WLAN will be used as one of the communication technologies. WLAN is a standardized technology that provides a high data rate. Due to the fact, that WLAN is standardized, a huge amount of hardware comes with WLAN radios on board which makes hardware providing WLAN rather cheap and a large amount of vendors to buy WLAN hardware from. Furthermore, one WLAN access point is able to handle around 270 connected clients, therefore, WLAN scales very well when using several access points.

- **Bluetooth:** In their paper, Ferro et al. [14] describe the Bluetooth standard as a low-power system that was originally invented to replace cables in short range areas. It operates on the 2.4 GHz frequency band and provides a throughput of up to 706.25 kbits/s in a range of at most 100 meters. In reality, the range is not higher than 30-40 meters with the strongest available radios.

  Bluetooth is designed as a point-to-point technology which means that it is not able to create mesh networks as many other wireless technologies do to send data over several other devices to the target device. Bluetooth needs a master device with which all other devices can communicate. At most 7 slave devices can be connected to a master device. This constellation is named Piconet. Slaves can act as gateways to other Piconets where again a master and several slaves communicate. This topology allows bluetooth to form rather complex networks. Bluetooth devices can act as both master or slave but not at the same time. Therefore, Bluetooth does not need any special infrastructure to form networks. Bluetooth is designed to transmit only small amounts of data due to the low data rate but for connecting sensors in IoT applications the data rate is high enough. Due to these properties, Bluetooth is also a suitable technology for smart factories.

  Recently, another Bluetooth technology named Bluetooth Low Energy (BLE) was released. In his paper, Gomez [15] gives a very detailed overview of BLE. He states that BLE also communicates on the 2.4 GHz frequency band with a data rate of up to 1 Mbit/s. BLE consumes a significantly lower amount of energy than Bluetooth and is theoretically able to connect around 2 billion slaves to one master. In reality, it is possible to connect around 200 slaves per master. This enables BLE to form much simpler networks but due the point-to-point architecture also BLE is based on, it cannot form mesh networks. The master performs TDMA which is an abbreviation for time division multiple access. This means that each slave gets one specific time slot where he can communicate with the master.

In contrast to Bluetooth, BLE devices are only active when they reach their time slot to transmit and receive. All the other time BLE devices stay in sleep mode and consume almost no energy. Bluetooth devices are always active and therefore consume much more power. As for Bluetooth, setting up BLE is rather cheap as no special infrastructure is needed and the huge amount of devices connectable to one master makes it even easier to set it up than Bluetooth.

- **Zigbee:** Zigbee is a low-power communication technology used mostly for industrial or medical applications based on the IEEE 802.15.4 standard as stated by Baronti et al. [16]. IEEE 802.15.4 specifies the lower layers of low-cost, low-speed wireless personal area networks. Zigbee can provide data rates of up to 250 kbit/s on the 2.4 GHz frequency band. The transmission range of Zigbee is limited to at most 100 meters line-of-sight depending on the environmental conditions.

  Zigbee, in contrast to Bluetooth, is able to form mesh networks which enables it to transmit data over long distances. Each network must provide a coordinator device which is needed to create the network and maintain it. Zigbee provides a beacon-enabled mode which allows it to grant real-time ability. This is done by using guaranteed time slots(GTS) instead of CSMA/CA which is typically used.

  Furthermore, Zigbee comes with some security features. The coordinator is used as a key storage and trust center. Zigbee uses AES128 per default for encryption. Due to these special abilities, Zigbee would be well suited for smart factories as it provides real-time transmission of small amounts of data and also encrypts the data to avoid eavesdropping of unauthorized persons. However, most sensors or machines do not provide Zigbee radios which limits the number of possible devices drastically. To add Zigbee radios to existing devices is also very expensive. Furthermore, it needs the target device in the line of sight of the coordinator device.

- **WirelessHART:** In their paper, Song et al. [17] define WirelessHART as

  > *At the very bottom, it adopts IEEE 802.15.4 as the physical layer. On top of that, WirelessHART defines its own time-synchronized MAC layer. Some notable features of WirelessHART MAC include strict 10ms time slot, network wide time synchronization, channel hopping, channel blacklisting, and industrystandard AES-128 ciphers and keys. The network layer supports self-organizing and self-healing mesh networking techniques. In this way, messages can be routed around interferences and obstacles.*

Additionally to these features, WirelessHART provides a central network manager that schedules communication and maintains the routes in the network and therefore, ensures the network performance. In contrast to Zigbee or Bluetooth, WirelessHART is able to guarantee a maximum communication delay. To ensure this, a very accurate timer is needed in the WirelessHART modules. As timing is the essential component of WirelessHART to ensure a maximum latency when transmitting data, the devices are very expensive and need a special gateway as network managers. WirelessHART provides a range of 250 meters with a troughput of 250 kbit/s on frequency band 2.4 GHz.

- **RFID:** Radio-Frequency Identification is a wireless technology that is used to identify and track tags that are attached to objects. Those tags contain information such as an ID. It can be used, for example, to track the assembly parts of some product in the factory. To get the information from the tags a reader is needed. The tag does not need to be in the line of sight of the reader as electromagnetic waves are used. This enables tags to be attached inside of an object so that it cannot be seen by the customer when the product is finished.

  In his paper, Want [18] describes passive and active tags. He states that passive tags have no battery and get the energy to send their data from the electromagnetic field created by the reader. The energy harvested from the field is enough for the tag to send their information. Therefore, it is necessary that a reader is very close to the tag to read it.

  Want also describes active tags as battery powered tags that can send data without having a reader nearby over several hundred meters. RFID tags are mostly used for identification of objects and not to store a lot of data or process it. Passive RFID tags can theoretically transmit in a range of up to 6 meters with a data rate of 100 kbit/s. This technology would be very useful in smart factories as the position of each assembly part should always be known in a smart factory environment.

  A very big advantage of RFID in contrast to the other wireless technologies is, that the tag is powered by the electromagnetic field of the reader which prevents unauthorized persons from reading the tag from, for example, outside the factory. This brings a huge security advantage compared to the other technologies where the traffic produced by those technologies can easily be read by an unauthorized person from outside the factory.

- **NFC:** In his book, Finkenzeller [19] gives a detailed overview of the near-field communication protocol and its operation modes. NFC uses electromagnetic induction between two antennas to exchange information between two devices similar to RFID. With NFC, in contrast to RFID, it is possible to not only read from tags but also write to them. In the communication there is always an initiator and a target device. To communicate, the initiator always needs to start the communication. NFC also provides an active and a passive mode.

The active mode is the peer-to-peer mode where two NFC devices communicate to each other. This means that both devices can act as initiator but also as target. For example, device one wants to send data and acts as initiator. The magnetic field created by device one moves trough the antennas of device two and device two acts as target and receives data. Afterwards, device two can act as initiator and send data. Then device one acts as target. In this mode, any type of data can be transmitted. In the active mode, the electromagnetic field created by the initiator is interrupted after the data was sent.

The other two modes are passive modes. The most important difference to the active mode is that after data transmission by the initiator is done, the electromagnetic field is not interrupted. The initiator waits for the response from the target.

The *reader emulation mode* allows NFC devices such as smartphones to read or write information on NFC tags. The NFC device acts similar to an RFID reader and the tags similar to RFID tags in this mode but the NFC device can write to the tag.

The *card emulation mode* enables NFC devices such as smartphones to act as smart cards, for example, for payment. The NFC device acts as target and responds to the data it got from the reader.

NFC operates in a range of around 5 centimeters with a throughput of 424 kbits/s. Similar as RFID, NFC could also be very useful in a smart factory and provides the same security feature as RFID as the tags and traffic between two devices cannot be read from far away.

	max. Range	**Data rate**	**Integration Effort**	**Cost**	**Advt./Disadvt. +/-**
LTE	10 km	150 Mbit/s	High	Low	+ low latency + high efficiency + small chip size − complex infrastructure − low scaling potential
WLAN	100 m	600 Mbit/s	High	Low	+ simple infrastructure + high scaling potential + high efficiency − routers and access points needed

					− high collision potential
IWLAN	100 m	450 Mbit/s	High	High	+ high efficiency + high scaling potential + soft real-time − special access points needed − expensive
Bluetooth	100 m	706.25 kbit/s	Low	Low	+ small chip size + no infrastructure needed − low efficiency − huge number of pico nets
BLE	40 m	1 Mbit/s	Low	Low	+ small chip size + no infrastructure needed + huge number of devices connected to one master − low efficiency
Zigbee	100 m	250 kbit/s	High	High	+ high scaling potential + build-in security − low efficiency − line-of-sight needed − access points
Wireless HART	250 m	2 measure-ments	Low	High	+ high efficiency + real time + build-in security − special gateway needed − very expensive
RFID	6 m	100 kbit/s	Low	Low	+ small chip size + simple infrastructure + no encoding needed due to small range − low efficiency

					– cannot form networks – readers needed
NFC	10 cm	424 kbit/s	Low	Low	+ small chip size + no infrastructure + smartphone as reader + no encoding needed due to small range – low efficiency – cannot form networks

Table 2.1: Overview of wireless technologies regarding their properties. Cost describes the expensiveness of the needed hardware. Integration Effort describes the effort to integrate the technology in an existing network. The last column lists advantages marked with + and disadvantages marked with −.

Table 2.1 gives a short recapitualtion of each wireless technology regarding properties such as range or data rate. Each wireless technology provides advantages and disadvantages. Stenumgaard et al. [20] wrote a paper regarding the challenges and conditions of wireless M2M communication in industry. They state that electromagnetic interference is one of the major issues for wireless communication technologies in an industrial environment.

Furthermore, the authors state that the time delay of transmissions due to electromagnetic interference is rather high, especially when the wireless technology uses re-transmission mechanisms. This behavior can lead to huge delays in data transmission which might be a problem for critical applications as the real-time constraints might be exceeded.

In the paper, Stenumgaard et al. [20] also compare some of the above mentioned technologies regarding their time delay depending on the electromagnetic interference. The results show that regarding the sensitivity on electromagnetic interference LTE is the most promising technology as it provides a high data rate but at the same time a low time delay.

Finally, the authors give a summary of issues and solutions to ensure better robustness of wireless technologies in an industrial environment. The summary includes points such as the problem of highly reflective environments which means that many reflections of radio waves are created due to wave propagation. This might lead to problems as the actual signal as well as the reflections of this signal are detected by the receiver. This might cause errors as incorrect data is received. To overcome this problem, multiple antennas could be installed in the production facility to ensure a better wave propagation.

Wireless technologies, apparently, have to deal with some real issues that were not solved yet, which do not occur when using wired technologies.Wired technologies could be extended, modified or combined with wireless technologies to also fit into smart factories. Furthermore, wired technologies often provide real-time ability which is very important in many processes of a production facility such as emergency stops of machines or data transfer to ensure a optimized production process. The following list gives current state-of-the-art wired technologies used in production facilities:

- **Profibus:** This standard for fieldbus communication uses serial ports to connect devices to the bus. There are two different types of Profibus, Profibus DP and Profibus PA.

  Profibus PA is an abbreviation for process automation and was designed for usage in hazardous areas. It provides a much lower data rate than Profibus DP.

  Profibus DP is a shortcute for decentralized peripherals and is used to communicate with sensors and actuators via a centralized controller. Profibus DP provides a data rate of around 12 Mbit/s. The bus is monitored by a master and can be operated in single- or multi-master mode. The bus is standardized so that all devices with D-sub serial ports [21], regardless of the vendor, can communicate to each other. Therefore, this is one of the most used fieldbuses in automation environments.

  Tovar et al. [22] show how Profibus can support real-time communication in industrial environments. They propose two different approaches to guarantee real-time where the first one supports tighter deadlines and the second one increased non-real-time throughput. They adopted the first approach and created a deadline-based priority mechanism. This approach was only applied on high-priority messages where real-time was necessary. They showed that Profibus is well suited to perform both real-time and nonreal-time communication on the bus.

- **Profinet:** The Process Field Net is based on Industrial Ethernet and designed for communicating with all kinds of devices in an industrial environment. It provides a particular strength in delivering data within real-time. It is divided into 3 protocol levels:
  - IRT (Isochronous Real-Time) provides delivery of data within cycle times of less than 1 ms
  - RT (Real-Time) is able to deliver data within cycle times of up to 1 ms
  - TCP/IP is used for non time-critical communication with a worst case reaction time of 100 ms

Feld [23] states that Profinet provides a real-time protocol suitable to transfer cyclic and acyclic data which makes it a very efficient fieldbus for industrial automation concepts. He also mentions that Profinet support different real-time classes, local real-time scheduling and synchronized real-time scheduling.

Local real-time scheduling is used for applications with a cycle time of around 5 ms. It uses a prioritization to assure that real-time data is preferred over other traffic. With 100 MB/s the throughput of Fast Ethernet is enough to enable real-time traffic besides other traffic.

Synchronized real-time scheduling is done when cycle times of around 1 ms and below are needed. To assure such small cycle times, each application gets a different time slot where exclusively this application is allowed to transmit data.

These mechanisms enable Profinet to communicate in real-time with any device connected to the bus.

- **DeviceNet:** This fieldbus is based on CAN [24] and consists of up to 64 nodes per segment. Due to the limited number of nodes, DeviceNet is designed to have several subnets in the network that communicate to each other using special routers. CAN is used for serial communication mainly in the automotive industry but is also used in other time-critical industrial environments. Messages are sent with priority-based arbitration on the bus.

  In their paper, Lian et al. [25] state that the messages in DeviceNet are designed in a way that each message requires a response from another device. DeviceNet is able to communicate in real-time. To do so, it is necessary to establish a special connection between the devices. When messages of the real-time type are sent, the CAN identifier routes are used to transfer the data to its destination node.

  DeviceNet is optimized for short messages and the transmission delays for high priority messages can be guaranteed which enables real-time on DeviceNet buses.

- **EtherNet/IP:** This technology is based on Ethernet and uses IP as well as TCP or UDP to transmit its packets in the network. It uses IP for packet routing, TCP for explicit messaging and UDP for real-time control messages.

  In their paper, the ODVA [26] state that the TCP messages are used for diagnostic and configuration data as well as to establish real-time data transfer between devices. The UDP messages contain time critical data and due to the low protocol overhead provides a small packet size and enables multicasting in the network.

  EtherNet/IP, furthermore, uses unconnected and connected messaging. Unconnected messaging is used for connection establishment and low-priority data that is sent infrequently using TCP. Connected messaging is used to send frequent messages using UDP or real-time data.

The messaging connections are also divided into two types: explicit and implicit connections. Explicit connections are point-to-point connections between two nodes and messages are sent using TCP. Implicit connections are used for cyclic data transfer and support point-to-point as well as point-to-multipoint connections and uses UDP for data transmission.

	Connection Port	**Modes**	**Data Rate**	**Advt./Disadvt. +/-**
Profibus	D-sub serial	1) Process Automation 2) Decentralized Peripherals	12 Mbit/s	+ multi-master + standard port − no build-in real-time
Profinet	Ethernet	1) IRT 2) RT 3) TCP/IP	100 MB/s	+ real-time + standard port − time synchronization
DeviceNet	CAN	1) real-time 2) non-real-time	1 MBit/s	+ real-time + message prioritization − limited #nodes per segment
EtherNet/IP	Ethernet	1) Explicit 2) Implicit	1000 Mbit/s	+ no master + real-time − high protocol overhead

Table 2.2: Overview of wired technologies regarding their properties. The last column lists advantages marked with + and disadvantages marked with −.

Table 2.2 gives an overview of the wired communication technologies. As already stated, it is possible to combine wired and wireless technologies to enable a smart factory concept. In their paper, Cena et al. [27] mention some possibilities to combine some of the previously mentioned technologies to ensure real-time communication and also enable the usage of legacy devices in a smart factory. With such combinations existing factories could be transformed to smart factories.

One combination would be a Profibus with WLAN. The WLAN transceiver is installed on the Profibus' master device. The master could then send its data further to other devices using WLAN. A gateway is needed where the data that should be send or received is stored in buffers to be able to keep the Profibus functionality. Profibus needs acknowledgments for sent data and requests data from devices. The functionality of Profibus can be left unchanged when including such buffers.

A second combination shown is DeviceNet with WLAN. This can be achieved by modifying the DeviceNet routers and add a WLAN port to them. With this combination it is not necessary to add another component to the devices connected to the bus, which would keep the cost low. However, currently there are no such devices available. But is would be easy for DeviceNet manufacturers to add a WLAN module to the existing DeviceNet routers.

The previous two approaches do not provide real-time. Therefore, Cena et al. [27] mention some approaches that would guarantee real-time. The Profinet IO extension with WLAN uses a special bridge to connect Profinet to WLAN. The bridge is able to assign time slots that provide contention free access to the medium and therefore, guarantee real-time communication.

Another approach is to combine EtherNet/IP with WLAN. As EtherNet/IP uses TCP or UDP it is easy to extend with an access point and communicate with other access points. With this technique, each device can prioritize the traffic it wishes to send. This priority influences the route of each package and the priority of the package on each device that forwards the package. The prioritization guarantees real-time communication.

All the previous possibilities where linked to WLAN. Another possible wireless technologie is IEEE 802.15.4 which specifies the lower layers of low-rate wireless personal area networks (LR-WPAN). WLAN has been tested in industrial applications and proven suitable but the long battery life and low cost of LR-WPAN makes it also a suitable technology for such scenarios.

The first approach using LR-WPAN combines it with fieldbuses. To achieve this extension it is necessary to use gateways that translate service of the application layer of the fieldbus to use the LR-WPAN functionality. The MAC protocols of fieldbuses and LR-WPAN are not able to communicate to each other, therefore, this gateways are needed to translate. The gateway also acts as a coordinator for the LR-WPAN.

The other approach is combining LR-WPAN with EtherNet/IP to provide real-time. Here, Cena et al. [27] state that the best solution to combine those two is to use Zigbee, which is based on LR-WPAN, as it provides real-time communication and can deal with any network based on the IP protocol.

In their paper, the authors showed that it is possible to provide real-time behavior when using wireless technologies combined with already existing fieldbuses or ethernet. However, extending existing production facilities would be rather expensive as a lot of components need to be exchanged or added to enable such technology combinations.

2.2.2 M2M Protocols

After the possibilities regarding communication technologies were discussed, it is necessary to continue with suitable protocols. Protocols consist of a set of rules regarding syntax, semantic, synchronization and error handling. Devices use the same protocol to communicate in a defined way with each other. The protocol defines, for example, where in a package the payload starts or ends. Most more advanced protocols such as IP or TCP use a combination of protocols to transmit data. The different protocols are mapped onto layers and perform different tasks. Some are used for routing in the network, others are used for transport and still others are used to communicate with the application a user interacts with such as browsers.

In smart factories, protocols are needed as well to communicate with sensors or machines. In this subsection, several protocols will be described that might be used in a smart factory environment. Currently, several protocols are considered as suitable for IoT and therefore also for smart factories. The following list gives the currently most important protocols in this context.

- **MQTT:** The specification by Oasis [28] describes MQTT as:

  > *[...] a Client Server publish/subscribe messaging transport protocol. It is light weight, open, simple, and designed so as to be easy to implement. These characteristics make it ideal for use in many situations, including constrained environments such as for communication in Machine to Machine (M2M) and Internet of Things (IoT) contexts where a small code footprint is required and/or network bandwidth is at a premium. The protocol runs over TCP/IP, or over other network protocols that provide ordered, lossless, bi-directional connections.*

  It provides mechanisms to ensure a very small protocol overhead and furthermore, notifies clients when they get disconnected from their broker. The broker is the server in the MQTT scenario. Each client can publish messages that are related to a topic defined by the client. The publish messages are sent to the broker and contain some payload such as the value from a temperature sensor. To receive the messages of a specific topic, a client must subscribe to this topic. Clients send a subscribe message for a specific topic to the broker and always get new data when the client that publishes the data of this topic sends new data to the broker.

  Furthermore, MQTT deals with three qualities of service for message delivery. The first one, *At most once*, means that messages are delivered according to the best effort of the network but might get lost. The second, *At least once*, means that the messages are definitely delivered but the same message might be delivered several times. The last, *Exactly once*, means that the message is definitely delivered and arrive at the receiver exactly once.

MQTT can be used with TLS to secure the connection between client and broker. Due to this security mechanism, MQTT can be used to transmit highly confident data.

Standford-Clark et al. [29] define an extension of MQTT, namely MQTT-SN which was designed for sensor networks. It was designed for devices on non-TCP/IP networks such as Zigbee. Some of the major differences to MQTT are:

- To reduce the packet size, the topic names in the publish messages are replaced by short 2 byte IDs.
- Predefined topics were invented so that there is no need to register a new topic and the client does not need to receive a list of possible topics.
- The clients can find brokers without knowing their IP address because a discovery mechanism was added.
- Offline keep-alive was invented so that messages that would be sent to battery-powered devices are buffered until they wake up.

MQTT will be used as a communication protocol in this master's thesis. As it provides low overhead and scales very well for large, dynamic networks it is suitable for smart factories. The publish-subscribe principle, MQTT is based on, enables the broker to push messages to the client which means that no polling is necessary. This reduces the overhead significantly. Furthermore, it provides TLS which enables secure data transmission. This feature is another reason why MQTT is very well suited for a smart factory environment.

- **CoAP:** Shelby et al. [30] defined the CoAP protocol in the RFC7252 as a web transfer protocol that can be used in lossy networks to communicate with constraint devices. It is based on a request-response architecture between endpoints. CoAP uses URIs and HTTP media types such as GET or POST. Furthermore, it supports discovery of provided resources and uses UDP for message transport. CoAP uses servers and clients in their request-response scenario. The client request a specific resource using an URI and the server responds with the requested values.

  Messages can be marked as *confirmable* which means that when they are sent, a timer is started. When the timer expires, the message is retransmitted using an exponential back-off mechanism until the sender receives an acknowledgement message from the receiver. To identify which transmission received an ACK, a message ID is added to each messages which is send in the ACK message for identification. It is possible that an endpoint cannot provide a confirmation message, then he sends a reset message which means the packet was received but some error occured. Messages can also be marked as non-confirmable which means that there is no need for response from the receiver and also the sender does not care if the receiver got the message. The message ID is here also used to identify duplicate receptions.

When GET requests are sent from a client marked as confirmable a *piggypacked* response is sent. This means that the payload is attached to the ACK message from the server that contains the requested resource. So, first, the client sends a request to the server, the server replies with an ACK and the attached payload and finally, the client acknowledges the reception of the the servers ACK.

The last possible message type are separate responses. This is done when the server cannot respond immediately to a confirmable request, so no piggypack response is possible, because, for example, the new data value from a sensor is not yet available. Then the server just sends an ACK without payload to confirm that he got the GET request. Later, when the data is available, the server sends a separate confirmable response containing the payload that is then acknowledged by the client. The message ID is used for every message type for identification.

- **XMPP:** In their book, Saint-Andre et al. [31] define the eXtensible Messaging and Presence Protocol as a technology for real-time communication where the sent data structure uses the XML format. XML [32] is a widely used markup language. The format is very simple, and machine and human readable. It is very common in exchanging data over the Internet due to the simple structure.

  XMPP's architecture is based on a client-server concept. Everyone can run a XMPP server similar to the world wide web where everyone can provide websites on his server. In principle, clients communicate with each other using a unique name. The servers perform routing between clients in the same domain and are used for communication between domains. In this case, servers communicate with each other and forward messages from clients in their domain to other servers. Servers are also used to provide security features such as authentication, channel encryption, or prevention of address spoofing.

  XMPP also provides gateways which are used for translation between XMPP and other protocols. Multihop is not supported by XMPP. It is necessary to connect all clients directly to the server. XMPP uses TCP as underlying transport protocol. For XMPP clients it is optional to use a TLS secured communication with the server. However, end-to-end encryption would provide much better security as it is not guaranteed that the communication between two XMPP servers is encrypted.

- **DDS:** Pardo [33] states that the Data Distribution Service is based on a publish-subscribe principle and provides an interface that defines the data-distribution service on the application level. DDS aims to enable real-time, high-performance data exchange between communication partners. It is designed for big-data applications and might be used in IoT applications.

The architecture is rather simple as nodes can create topics and publish some data to the topics. Other nodes can subscribe to the topics and get the published data. In contrast to MQTT, DDS does not provide connection management. Nodes don't care if messages cannot be delivered. DDS supports quality of service and can specify parameters regarding, for example, discovery. Furthermore, DDS doesn't need a broker but uses a bus to connect the publishers and subscribers. As a transport protocol, it can use TCP, UDP, shared memory or any other transport specification.

- **AMQP:** In his paper, Vinoski [34] describes the Advanced Message Queuing Protocol as a message-oriented reliable protocol that provides publish-subscribe and point-to-point communication. It uses a message broker that is able to provide two kinds of services, exchanges and message queues.

  O'Hara [35] stated that with exchange, the message should be sent to a specific receiver and is put in a single queue the client can read from. When using the message queue mode, the message is copied to the queues of each client that subscribed to the topic given by the message. These message queues use the publish subscribe principle, the exchange mode uses direct messaging. When sending a message, the client decides whether it wants to exchange the message or to put it in the message queues. When using the message queue, the messages are stored on the broker either in the queue until the messages can be delivered or simply in the memory.

  AMQP also provides different delivery guarantees similar to the ones of MQTT, namely at-most-once, at-leat-once, and exactly-once. For transport, it usually uses the TCP protocol but can also be extended to use, for example, UDP or other protocols.

- **WAMP:** In their book, Bahga and Madisetti [36] describe the Web Application Messaging Protocol as:

  > *[...] a sub-protocol of Websocket which provides publish-subscribe and remote procedure calls (RPC) messaging patterns. WAMP enables distributed application architectures where the application components are distributed on multiple nodes and communicate with messaging patterns provided by WAMP.*

  WAMP usually uses Websockets for transport between two peers but can also use any other message-based bi-directional communication. The communication between two peers is called session and lasts until one client ends the communication. Clients can have several roles.

  In the publish-subscribe mode they can be publisher or subscriber. WAMP uses brokers to distribute the data to nodes. As WAMP supports RPC, the client can have two more roles, caller or callee. A caller invokes a remote procedure with arguments. A callee, on the other hand, executes the procedure invoked and returns the result to the caller.

The network structure of WAMP also contains routers which route calls and events. They can either be in broker mode or in dealer mode. The broker mode is used for publish-subscribe and performs the typical tasks of a broker. In dealer mode, they handle RPCs and route the calls from the caller to the corresponding callee.

- **WebSocket:** Fette [37] defines the WebSocket protocol as a two-way communication between a client and server. WebSockets, in contrast to other web applications, use only one TCP connection to transmit several data packets in both directions. Furthermore, WebSockets uses push/pull and not polling for data transfer. When establishing a connection between a client and a server, a handshake is performed. First, the client sends a message using HTTP with some options set in the header indicating that the clients wants to establish a WebSocket connection. The server replies with a HTTP message containing fields that acknowledge the connection via WebSockets . After that, the data traffic is sent using the WebSocket connection. The packets sent consist of a very small overhead and the payload. If a packet is sent containing no payload, the connection is closed. WebSockets provide the optional functionality of using TLS to secure the connection.

  Herfs et al. [38] state that WebSockets get more and more interesting for companies in the automation sector due to the low latency after the connection was established. The WebSocket protocol could be simplified for the use in factories as the step to use a HTTP connection could be skipped and a secure, bidirectional communication could be established directly. This would mean that clients and servers using WebSockets that are not able to deal with HTTP could use WebSockets for communication. Features regarding the security of JavaScript could also be removed as they are not needed in a manufacturing environment.

  As this is a rather new technology, that provides a very low latency and a very low protocol overhead after the connection was established, it is suitable for smart factory environments. Furthermore, it provides push/pull instead of polling, which also reduces the traffic overhead significantly. Due to these reasons, the WebSocket protocol will be used to transfer data between the machines on top of Ethernet.

Table 2.3 shows an overview of the previously discussed protocols. As all protocols have different advantages and disadvantages, it might be necessary to combine several protocols to create a reliable, robust data transmission between devices, servers and data storages such as clouds. One main goal will be to enable real-time communication in some areas of a smart factory to make sure that important messages such as control signals or error messages from machines are delivered in time.

	Architecture	**Advt./Disadvt. +/-**
MQTT	Publish/Subscribe	+ low package overhead + different QoS − no message prioritization − broker necessary
CoAP	Request/Response with REST-like resources	+ mode for guaranteed message delivery − UDP for transport
XMPP	Client/Server	+ XML structured packets + real-time − server necessary
DDS	Publish/Subscribe	+ no broker + real-time + different transport protocols − Nodes don't care if message delivered
AMQP	1) Publish/Subscribe 2) P2P	+ queuing of messages + TCP for transport − broker necessary
WAMP	1) Publish/Subscribe 2) RPC	+ WebSocket for transport + clients have several modes − broker/router necessary
Websocket	Client/Server	+ real-time + bidirectional, full-duplex communication − High overhead when setting up the connection

Table 2.3: Overview of IoT protocols regarding their properties. The last column lists advantages marked with + and disadvantages marked with −.

2.3 Security

Communication between sensors, controllers, machines, or the data cloud need to be secured to prevent unauthorized persons to be able to retrieve confidential data. Wireless communications needs to be secured as it is much easier to listen in on wireless traffic than on wired traffic. For intercepting wired traffic, a hacker needs access to the network, wireless traffic can easily be spied on using simple tools. An eavesdropper could, for example, wiretap the traffic of a wireless network from outside of the house.

The issues of eavesdropping on wireless communication in a smart factory is a problem as unauthorized persons might receive information that is highly confidential and publishing of this data would lead to huge financial losses for the company. Wireless communication results in more issues than just the possibility of eavesdropping. The issues as well as security solutions regarding smart factories and machine-to-machine communication will be discussed in this section.

2.3.1 Smart Factory Security

Lass et al. [39] state that it will be necessary to integrate IT security in Industry 4.0 factories when the factory is designed and not after the factory was completed. They also claim that security mechanisms should be integrated in Industry 4.0 hardware components when they are designed and built.

It is very hard to add security controllers to legacy devices as they need to be connected to the hardware somehow which on the one hand involves a great deal of expense and on the other hand is accompanied with a lot of additional work. By considering such security controllers as parts of the hardware when the device is designed a large amount of money and time can be saved when such devices include security mechanisms in their design.

Lass et al. [39] also discuss the necessity of the integration of security concepts in legacy devices as the conversion of existing factories to smart factories will happen gradual.

Security issues regarding software such as issues with viruses or worms are a major problem of legacy devices and machines in state-of-the-art factoris. Junker [40] states that such malicious software leads to overload or breakdown of the communication network or major failures in PLCs. To solve the issues in existing factories, Junker references to the Top 10 threats and solutions published by the *German Federal Office for Information Security.* [2]

After that he states that the main issue of Industry 4.0 production facilities might be the increased networking between devices. Software issues such as the ones of

[2] https://www.bsi.bund.de/DE/Themen/Industrie_KRITIS/Empfehlungen/ICS-Betreiber/empfehlungen-betreiber_node.html

legacy devices might cause much more failures in smart factories as due to the increased networking viruses or worms could be distributed much faster. Furthermore, he states that the targeted attacks on production facilities increase which are much worse than viruses or worms that coincidentally enter the system. Targeted attacks mostly aim to extract confidential data from the attacked system which could ruin a company when, for example, the chemical formula of steel produced in a company would be published.

Wallner et al. [41] propose a security solution to prevent targeted attacks on Industry 4.0 factories. They state that it is essential to provide secure authentication on machines and devices in the production floor, ensure integrity of messages between devices and guarantee deterministic encryption to guarantee that the real-time constraints are not exceeded. Furthermore, the key and authorization management must be able deal with the huge amount of keys and roles in the factory.

To achieve the goal of preventing targeted attacks, secure elements enabling real-time computation that guarantee tamper resistant keys stored on the hardware need to be added to every device that communicates inside the factory. The authors also state that device identification and secure updates are a very important aspect of the concept. Device identification is the first step of securing the currently unsecured communication via bus systems on the production floor. The authors suggest to use a Trusted Identity Manager for device identification. Furthermore, it is necessary to ensure that messages were not manipulated between sender and receiver by using cryptographic hash algorithms.

In contrast to Wallner et al. [41] who uses existing security mechanisms to secure a smart factory, Sadeghi and Wachsmann [42] state that it is necessary to develop security concepts especially for Industry 4.0. The authors mention that a large amount of embedded systems exist that provide trusted computing based on secure hardware or processor architectures with secure execution. But they also state that:

> *[...] all these approaches are too complex for low-end embedded systems, which are typically designed for specific tasks and optimized for low power consumption and minimal costs. Often they must provide multiple features and meet strict real-time requirements. Security solutions for these devices are typically based on hardware-enforced isolation of security-critical code and data from other software on the same platform.*

Finally, the authors state that state-of-the-art security technologies do not scale to such large networks with real-time requirements and often constraint devices such as sensors. Therefore, they claim that it is necessary to design new security protocols and IoT security mechanisms.

2.3.2 M2M Security

Barki et al. [43] state that a lot of research is done on M2M communication but only few focus on security aspects. In their paper, they give an overview of security challenges regarding M2M and also investigate possible solutions regarding scalability and suitability in M2M communication scenarios. The authors state that one main issue with M2M devices is that they are expected to operate over a long period of time without the need for maintenance. The main threats of M2M communication can be seen in the following list:

- Physical attacks targeting hard- and software such as side channel attacks [44], malicious software, or destruction or theft of a device.
- Logical attacks targeting functionality of the device such as impersonation (attacker spoofing the identity of a server) , denial of service [45], or relay attacks [46].
- Data attacks targeting data traffic such as privacy attacks [47], data modification and false information injection, or selective forwarding/interceptions (delaying or dropping intercepted packets).

Barki et al. [43] compared several existing security solutions for M2M communication regarding scalability, resource constraints, mobility, robustness, delay constraints, types of communications and device heterogeneity for different security services. They state that there are no concepts providing appropriate scalability, robustness and delay constraints in all security services which would be essential for secure M2M communication in smart factories. Furthermore, the authors state that it might not be possible to design a M2M communication security concept that can be applied in any field of application for M2M due to the different requirements of each domain such as home automation, smart factories or wireless sensor networks in public.

Ahmadzadegan et al. [48] propose a framework for secure M2M communication. When designing the framework they payed attention to energy efficiency, reliability and security. In order to ensure security, the framework needs to fulfill the requirements of

- **Confidentiality** is necessary to ensure that only authorized persons have access to the data sent by devices.
- **Integrity** must be ensured that the data was not modified between the sender and receiver.
- **Authentication** is necessary to make sure that devices sending or receiving data are the ones they pretend to be.

Depending on these security requirements and the requirements of reliability and energy efficiency, the authors proposed a 5-staged triangular concept. This triangular trade is used to model the trade offs between the factors security, reliability and energy efficiency.

The authors state that a combination of the 5 stages would lead to an ideal M2M communication model regarding the 3 influencing factors. Each stage deals with a different level in the M2M architecture. Stage 4, for example, deals with the trade offs between data transmission, data collection and data analysis. The proposed stages do not give any advice about which architecture or technology should be used but describe which requirements must be fulfilled to ensure a secure M2M communication.

Mahkonen et al. [49] propose a demo of a secure M2M communication architecture that sends data from constraint devices over the Internet to a cloud server where they also state technologies that could be used to secure such a scenario. The data is sent using a gateway to the data cloud. Furthermore, the M2M devices can be accessed using the Internet via this gateway.

To authenticate the devices on the server, the *Generic Bootstrap Architecture* (GBA) is used as very little user interaction is needed to use it. GBA uses the authentication and key agreement protocol (AKA) and also utilizes shared secrets between devices.

To ensure that the data is not manipulated between the device and the data cloud, the digital signature scheme ECDSA is used. The authors state that the proposed architecture scales very well for thousands of devices, the devices can be access through the Internet and it is not possible to manipulate the sent data.

2.3.3 Security Technologies

In the previous sections several architectures and technologies were described to ensure a secure communication between different communication partners using different technologies. Wireless communication has other requirements than wired communication in the context of smart factories. Therefore, different technologies ensuring the security of data sent wired and wireless in a smart factory will be given here.
There are two main kinds of cryptography systems: symmetric and asymmetric cryptography.

Bellare [50] gives basic definitions of symmetric and asymmetric cryptography. Asymmetric-key or public-key cryptography relies on two different keys where one is available for the public and the other one is private to the owner. The public key of a receiver can be used by anyone to encrypt a message but decryption is only possible with the private key of the receiver. Therefore, to secure the communication it is only necessary to keep the private key private as decryption with the public key is not possible. Two main usages of public-key cryptography are public-key encryption and digital signatures.

Symmetric-key algorithms use the same key for encryption and decryption of data. These keys are named *shared secret* and are private to the devices communicating with each other using an symmetric-key algorithm. The main drawback of symmetric-key cryptography in contrast to public-key cryptography is that the *shared secret* needs to be accessed by all communication partners once. Therefore, public-key cryptography is often used to transfer the shared secret of symmetric-key algorithms. Furthermore, there are two types of symmetric-key encryption, namely stream cipher and block cipher. Stream ciphers encrypt digit by digit of message whereas block ciphers encrypt block by block where one block consists of several digits.

A widely used symmetric cryptography algorithm is AES [51]. The *Advanced Encryption Standard* is a block cipher using a block size of 128 bit and supporting keys with lengths 128 bit, 192 bit and 256 bit. As the block size is 128 bit (=16 byte) it is possible to represent them as a 4x4 matrix named *state matrix*. The algorithm consists of 4 different steps. First, the keys needed later are generated. Second, the initial round is performed of adding round keys is performed. This means that each byte of the state is xored with one byte of the round key. Third, the rounds consisting of 4 different steps are computed. The number of rounds depends on the size of the keys. Fourth, the final round is performed. The following enumeration contains the steps performed in the rounds:

1. **SubBytes:** In this step, each byte of the state matrix is replaced by another byte according to a substitution box that operates non-linear.
2. **ShiftRows:** Each row of the state matrix is shifted cyclically to the left by a certain offset. The first row is not shifted, the second row is shifted by one byte to the left, the third row is shifted by two bytes to the left and so on for 128 and 196 bit keys. For 256 bit keys the row one is not shifted, row two shifted by one byte to the left, row three is shifted by three byte and row four is shifted by four byte to the left.
3. **MixColumns:** This step affects the columns of the state matrix as each columns is multiplied with a fixed polynomial to provide diffusion in the cipher.
4. **AddRoundKey:** Each byte of the state matrix is bitwise xored with one byte of the subkey of the current round that was generated in the initial step of the algorithm.

A very popular asymmetric algorithm is the *Diffie-Helman key exchange* [52] that is used to securely exchange keys between two parties over a public channel. To perform a key exchange the following steps by the two parties have to be taken:

- Both parties agree on a prime p and g which is a primitive root modulo p. These parameters are public.
- The first party chooses a secret integer a and the second party chooses a secret integer b. Then both compute public values A and B and sent it to each other
- From the messages A and B both parties can compute the shared secret s.

Figur 2.3 shows an illustration of the process where red marked parameters are secret and green marked parameters are public. Furthermore, the figure contains the used formulas for the computation of the parameters.

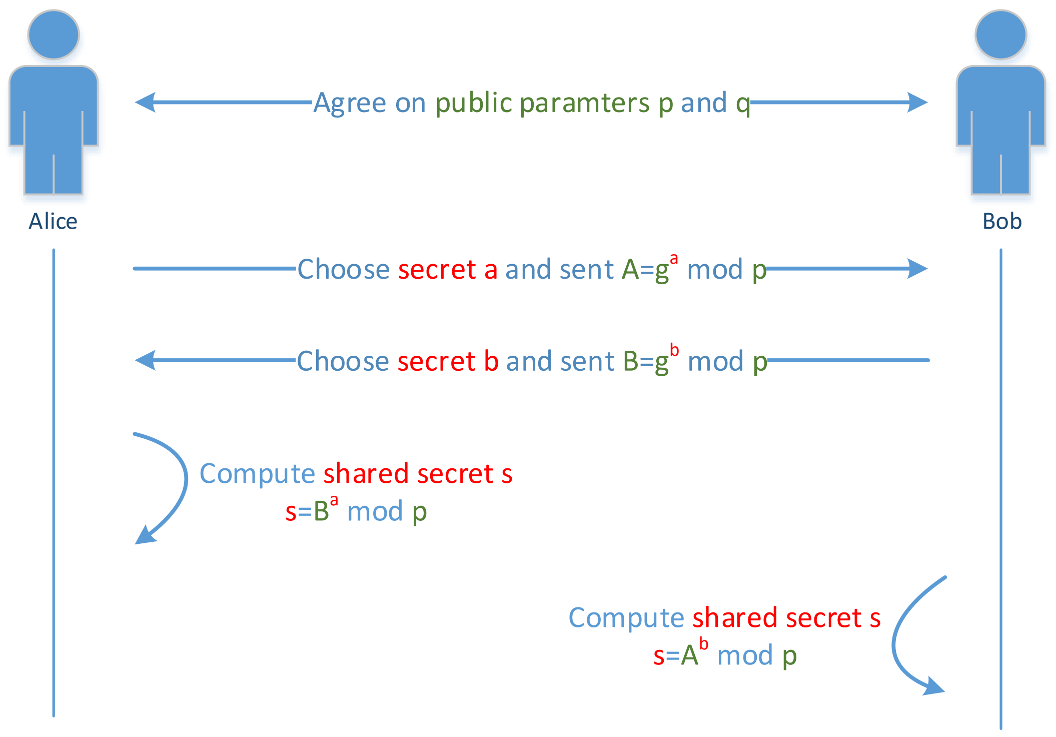
Agree on public paramters p and q
Alice
Bob
Choose secret a and sent A=g^a mod p
Choose secret b and sent B=g^b mod p
Compute shared secret s
s=B^a mod p
Compute shared secret s
s=A^b mod p

Figure 2.3: Schematic illustration of the Diffie-Hellman Key Exchange process.

In this master's thesis, TLS [53] will be used to encrypt data transmitted over a wireless medium. TLS uses a symmetric-key algorithm to encrypt messages. However, it also uses asymmetric cryptography to exchange the key for the symmetric-key encryption. TLS provides authenticity, confidentiality and integrity which makes it a suitable security mechanism for a smart factory environment. To initiate a TLS connection, a handshake is performed where a client and a server agree on the symmetric-key algorithm and the shared secret using public-key cryptography and also using certificates to authenticate each other. A detailed description of TLS will be given in Chapter 3.

Digital Signature

A digital signature is used to ensure authenticity of messages. To digitally sign a message, the sender first computes a hash of the message and encrypts the hash with his private key. Then the message is sent to the receiver that can verify the signature by computing the hash of the message, then using the public key of the sender to decrypt the message. If the decrypted and computed message match, the signature was valid which ensures that the message was not modified between sender and receiver.

To ensure that the data sent between the machines, ECDSA [54] will be used in this master's thesis to sign the data. ECDSA is based on the digital signature algorithm [55] but uses the elliptic curve discrete logarithm problem (ECDLP) [56] instead of the discrete logarithm problem. The ECDLP is to find the discrete logarithm of a random elliptic curve element with respect to some base point that is public. The larger the used elliptic curve is, the more difficult the problem is to solve. A more detailed description of the ECDSA will be given in Chapter 3.

3 System Design

In this chapter, the description of the system regarding the architecture of the approach will be discussed. First, the general hybrid communication architecture and several scenarios describing why this architecture could be applied in a smart factory context will be shown. The term *hybrid*, in the context of this thesis, relates to the fact that wired and wireless communication approaches are combined to meet the requirements of data transmission in a smart factory environment. Second, the used protocols and communication techniques will be shown in detail. Finally, the security mechanisms to protect the transmitted data will be discussed.

3.1 Hybrid Communication Architecture

Figure 3.1 gives an overview of an example architecture which might be used in a smart factory. The figure shows the ability to interact with the system from anywhere in the world any time as it is connected to the Internet. The production lines are the scope of this thesis and can be seen in more detail in Figure 3.2. In Figure 3.1, only three production lines are shown but a factory contains of course much more than three production lines. In Figure 3.2, just two machines were added to the picture but obviously a production line can consist of many more machines. Production lines can communicate to each other using the gateways and the intranet.

The parts shown in Figure 3.2 will be implemented except of the data cloud as it would have exceeded the scope of the thesis which is to implement a hybrid communication concept. The approach enables a fast, reliable communication between machines using a wired technology and also a wireless communication. The overhead between machines and controllers is kept very low due to suitable protocols. The three communication partners implemented are the two machines and the broker/gateway which could make data available on the Internet or a data cloud.

Machines are used to manipulate products such as assembling several parts together. Furthermore, they consume and produce a large amount of data to perform manipulations. In smart factories much more data is collected from every device in the facility to optimize production flows automatically as the machines and controllers can automatically adapt to the environmental changes. Machines provide

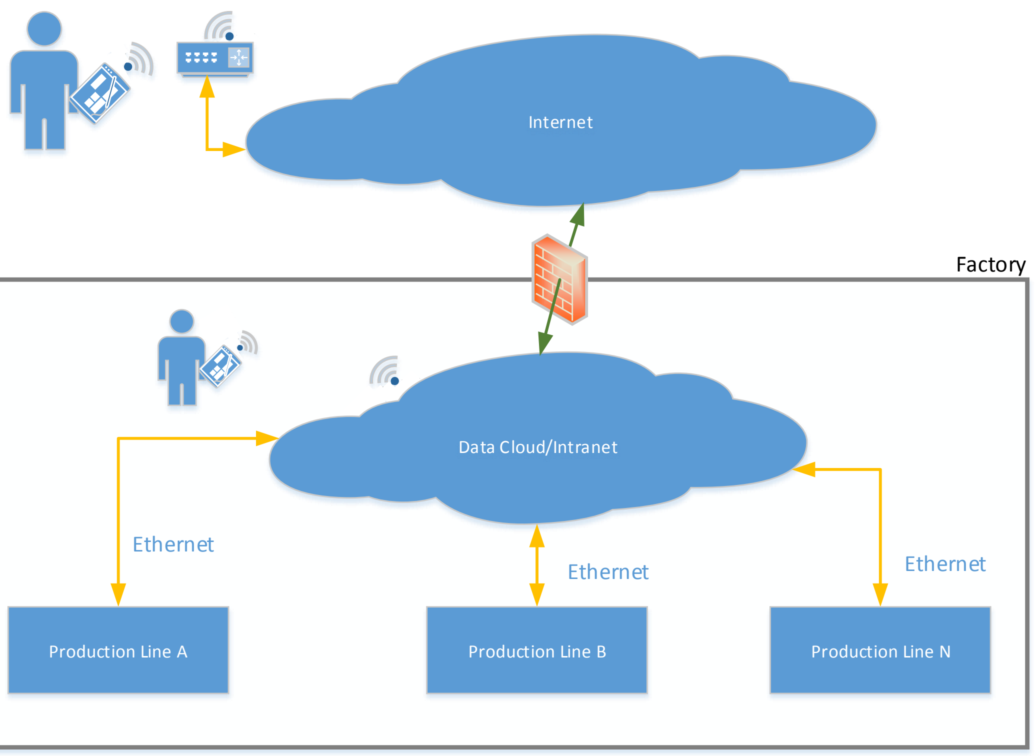

Figure 3.1: Overview of a possible structure of a smart factory developed during this thesis. A detailed description of a production line can be seen in Figure 3.2.

data such as their consumed energy, information about the currently manipulated product, or other maintenance data. The following list gives some examples of usages of maintenance data provided by machines:

- Tracking vibrations or energy consumption of machines to predict when maintenance will be required for the machine.
- Compare machines regarding their performance by tracking, for example, the number of manipulated products, energy consumption and other parameters to discover machines which perform, for example, worse than the average machine.
- Generation of graphs representing, for example, the time period needed to manipulate a product depending on the configuration than can be used by employees to optimize configurations or production flows.
- Machines could send alerts to other machines when a thresholds such as operation temperature is exceeded. The other machines could then reconfigure the production process by instructing other machines to provide the needed product. In this case, the machines would adapt to the failure of one machine

automatically and choose another machine to take the tasks of the defective machine.

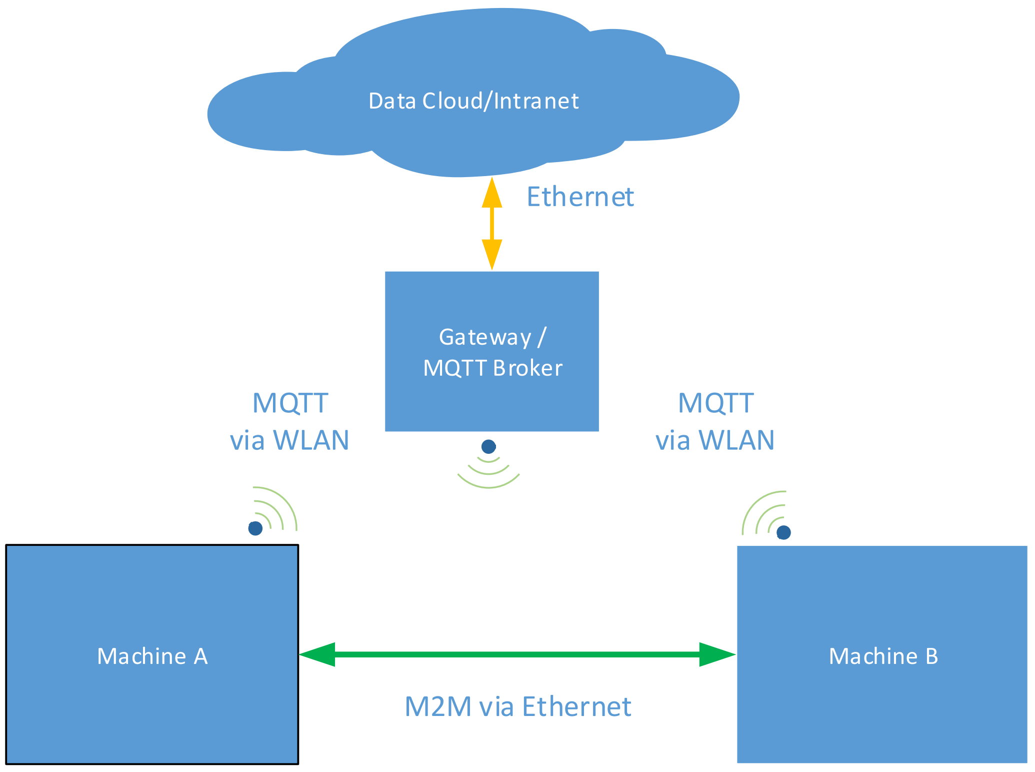

Figure 3.2: Overview of the hybrid communication appraoch implemented in this thesis with used protocols and communication technologies.

These are just some examples but there are many more usages for the maintenance data produced by the machine. Machines can also read data from the products they manipulate. They can identify the product and get further information about it regarding, for example, the material a product consists of. This can furthermore, influence the manipulation performed by the machine or it could change the production flow of the product. The following list gives some examples for scenarios that can occur depending on the product:

- A machine can, for example, manipulate two kinds of products where one consists of wood and the other one consists of metal. The product itself contains the information about its material in, for example, a tag. This tag is read by the machine and depending on the material the machine performs different manipulations.

- Depending on the products material, the production flow could be different. If, for example, the products material is wood the next machine to manipulate it is machine X whereas when the product consists of metal, the next machine would be machine Y. Therefore, the currently manipulating machine needs to inform the machine which should manipulate the product next so that the next machine can prepare itself for the product.

These are also just examples for scenarios where the product influences the production flow or the machine itself. All the scenarios show that the machine needs to make decisions depending on different environmental changes. To do so, they need to communicate to each other and also the controllers. In this thesis, the controller acts as broker and gateway. The broker is used to distribute non time-critical data between machines and the gateway is needed to store the data in the cloud or send it to other gateways that can forward the information to other machines or employees. The following list will give some examples for where a broker is needed:

- The maintenance data produced by machines needs to be stored in the data cloud. The data is sent to the broker and depending on the topic the broker decides whether it should store the data in the cloud. Later the stored data can be read by other brokers, employees or services which maintain the whole factory and make decisions about which machine needs maintenance or if machines perform worse than the average machine.

- Machines also extract informations from the products they currently manipulate. This information can be sent to the broker so that other machines can get details about the next products. For example, a production line consists of 5 machines. When a new product is inserted into the first machine the informations about the product can be sent to the broker so that, for example, the last machine gets the details of the product to be able to predict when it will be finished or predict assembly parts it will need in the near future.

- As employees can access all information in the factory from everywhere in the world at any time, they might have realized that a machine performs very bad. To overcome this issue, a new configuration or update could be added to the machine. There the broker is needed to send the new configuration to the machine.

Of course, there are many more scenarios but it would exceed the scope of the thesis to list every possibility. After the generally used architecture was described, the details regarding protocols and the communication technologies need to be discussed in the next section.

3.2 Protocols and Communication Technologies

To communicate between the machines, the wired technology *Ethernet* is used to ensure direct, fast and reliable communication between the machines. The WebSocket protocol is used for communication on top of Ethernet to connect the machines as it provides a bidirectional connection and almost no overhead after the connection was established. The communication between a machine and the broker is done using *WLAN* and as a protocol *MQTT* is used which relies on the publish-subscribe principle. The different communication technologies and protocols will be shown in this section and the reasons why they were chosen in this scenario will also be described. Furthermore, the security technologies and algorithms used to secure the data transmissions are discussed in detail.

3.2.1 Wireless Communication

WLAN

The wireless communication in this scenario is done using WLAN as already discussed in Section 2.2.1. WLAN does not provide real-time communication, but as the communication between the machine and the broker/gateway is not liable to real-time constraints, non real-time is not a problem. WLAN provides a high bandwidth which enables the machine to send any possible information to the data cloud. Furthermore, WLAN consumes a lot of power which, in general, is a problem in smart environments. But due to the fact that the machines are connected to the power grid and the broker/gateway is also connected to the grid, this is not a problem in this case. WLAN was already shortly described in Section 2.2.1 of the related work. Now, the transmission of data will be described in detail again.

As WLAN uses air as a transmission medium for the data, it is necessary to check whether the medium is free. If two different senders would send their data to the same receiver at the same time, a collision would occur. Therefore, WLAN uses CSMA/CA to make sure as few collisions as possible occur. CSMA is an abbreviation for carrier sense multiple access.

Carrier sense is used by the sender to check if any other transmission from another sender is currently going on. Multiple access means that several senders can access the same medium.

CA is a shortcut for collision avoidance and is necessary as the possibility of hidden nodes in a wireless network exists. Hidden nodes are a known wireless networking problem. It means that a node or client is visible from a wireless access point but not from another node. Figure 3.3 shows an example.

There, node A and B can both be seen by the access point, but the range of A is not enough to see B and the range of B is not enough to see A. If now both would check the medium to send to the access point, they would think the medium is free as they cannot sense any other node sending data. This situation would lead to a collision on the access point. To avoid this situation, CA is used. WLAN uses the

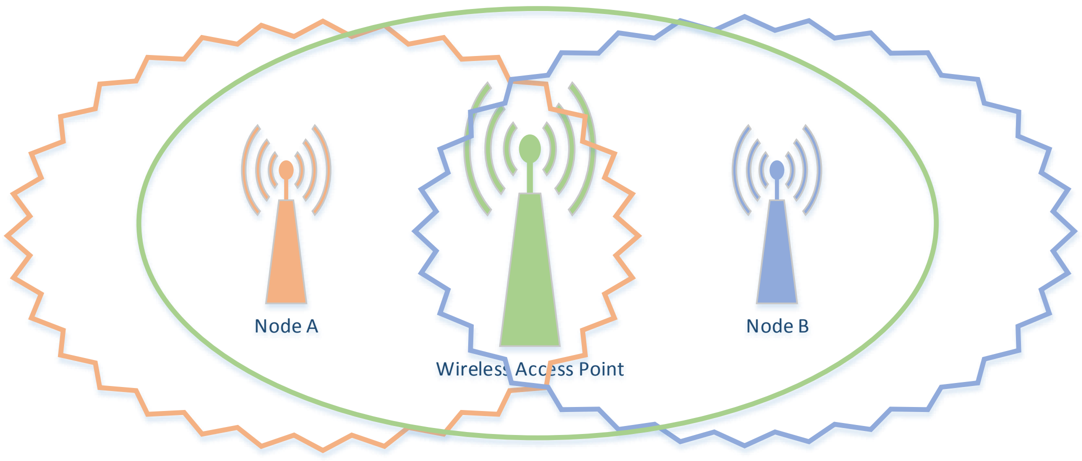

Figure 3.3: Hidden node problem showing that node A and node B cannot see each other but the access point is able to see both.

two signals RTS and CTS to assure that the medium is free and the node can send its data to the access point.

RTS is an abbreviation request-to-send. The sender sends this message to the receiver and if the medium is free the response is CTS which is a shortcut for clear-to-send. After that the sender is sure that the receiver will get his message and sends the data. The receiver sends an acknowledgement to the sender when he received the data. This is named the 4-step handshake of WLAN. Figure 3.4 shows the 4-step handshake procedure of sending data from a WLAN transmitter to a receiver.

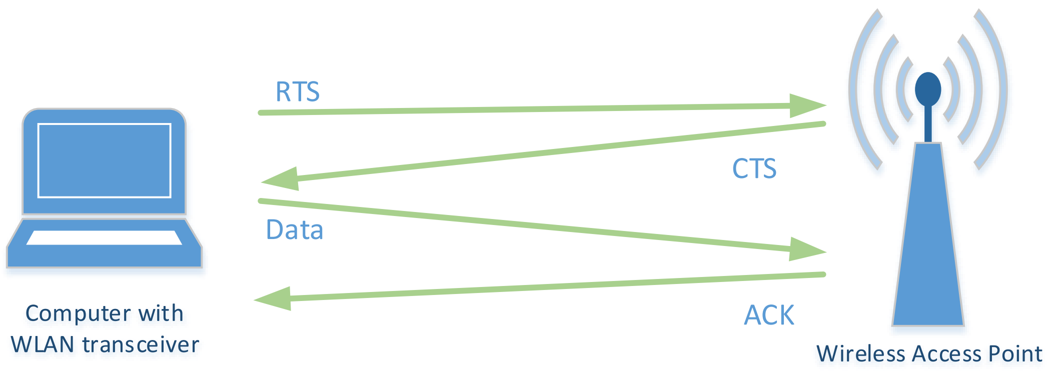

Figure 3.4: 4-step handshake when sending data using WLAN.

If the sender does not receive a CTS after sending RTS or if the sender detects that the medium is currently occupied, the sender waits for a random time until he checks again if the medium is free to send the data. The delay is random to prevent the case where several senders try to send at the same moment because with a fixed delay they would try to send at the same time over and over again. This mechanism avoids collisions but cannot prevent it, therefore, it is necessary to use a communication protocol that retransmits packets if they are lost due to collisions.

MQTT

As already discussed in Section 2.2.2, MQTT is used as a communication protocol in this thesis. MQTT is a lightweight communication protocol for machine-to-machine communication that, in this case, relies on TCP as transport protocol to ensure that each packet will be transmitted to the receiver. It is used in this scenario as it provides a very small overhead and categorizes data depending on topics which makes it very easy to deal with a large amount of data from several communication partners. Furthermore, it is able to deal with small bandwidth and high latency. As WLAN is used to transmit between the machines and the broker/gateway, MQTT is very good equipped to deal with the issues of wireless communication technologies.

MQTT was shortly described in Section 2.2.2 in the related work chapter but will be described in detail here again. First of all, MQTT is based on the publish-subscribe principle [57]. Publish-subscribe is a messaging pattern where the senders or publishers of data do not send the data directly to a receiver but instead group the data into different classes using so-called *topics*. Receivers or also called subscribers can state which topics they want to get information about and will only receive informations about topics they expressed interest in. Publishers have no idea who receives the data published by him and subscribers have no idea who published the data.

MQTT uses a so-called broker to connect topics to publishers and subscribers. The publisher sends his data regarding a specific topic to the broker. The subscribers can choose from a list of topics which they are interested in and will then get the corresponding data from the broker. A typical MQTT publish-subscribe scenario can be seen in Figure 3.5.

Topics can have several levels separated by a slash. An example would be *topic/subtopic/subsubtopic*. There is no limit for the number of levels. This can be used to group the data regarding to different contexts. For example, one could create a topic for the sensor type *temperature* and the subtopics could be the different sensors which provide temperature values. An example would be *temperature/sensor1*, *temperature/sensor2* and *temperature/sensor3*.

A subscriber which would like to get all temperature values measured by different sensors could then use a so-called multilevel wildcard. The subscriber could subscribe to the topic *temperature/#*. This means that the subscriber gets data from the topics *temperature/sensor1*, *temperature/sensor2* and *temperature/sensor3*. So,

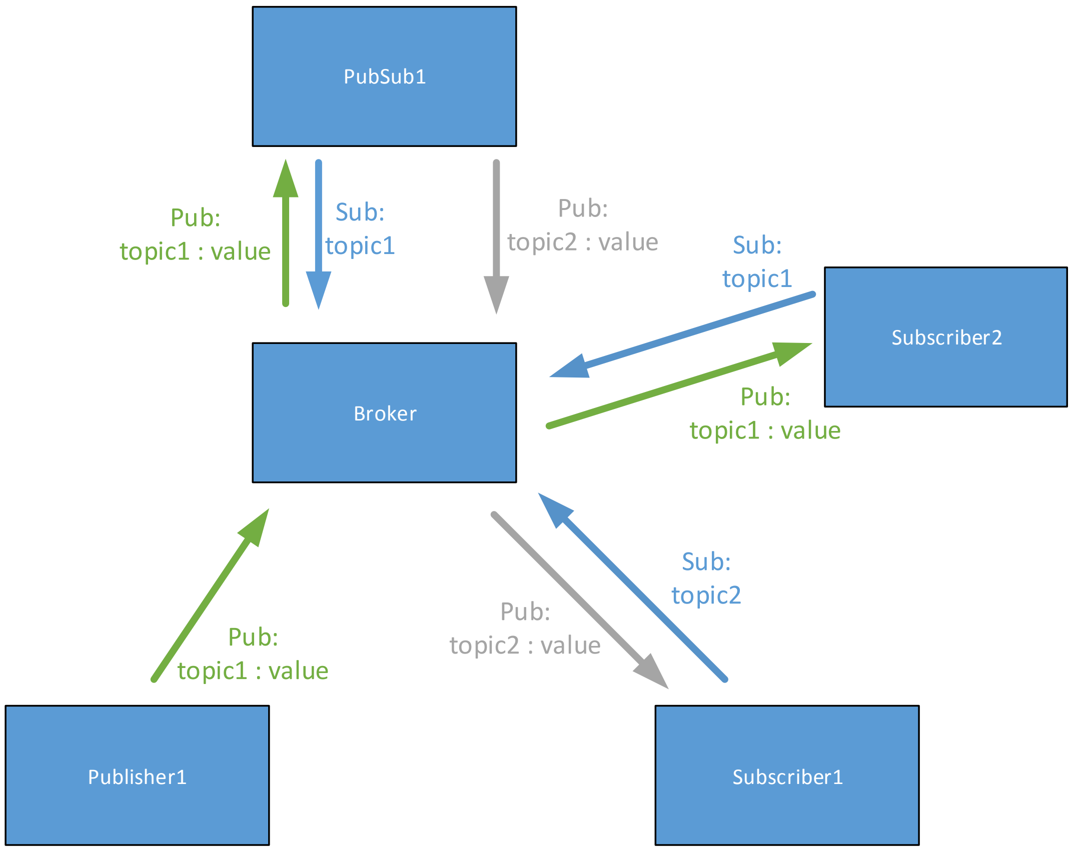

Figure 3.5: Example scenario for MQTT publish-subscribe with several publishers and subscribers using a broker to distribute data.

the multilevel wildcard overcomes the issue of subscribing separately to every available topic regarding a special context. This also means that when a new subtopic, for example, *temperature/sensor4* is created, the subscriber using the multilevel wildcard automatically also receives the data from topic *temperature/sensor4*.

To connect to an MQTT broker, the client first sends a connect message containing its unique identifier, username and password for authentication and authorization, and several other data fields for maintaining the connection. To distinguish between different publishers and subscribers it is necessary that each client has a unique name to identify the client and its current state. The broker also needs a unique name as it is possible to have several brokers in the same network. A broker can subscribe to the topics another broker provides. After a client sent the connect message, the broker checks if the user credentials authorize the client to connect to the broker and sends a connection acknowledgement containing a return code. Return code *0* means that the client is connected successfully to the broker. All other codes are error codes describing why the connection was refused.

To secure the connection between subscribers, publishers and the broker, TLS can be used. TLS will be explained in Section 3.3 in detail. The possibility to enable a security mechanism like TLS that provides confidentiality, integrity and authenticity is another reason to use MQTT. The properties of TLS ensure that no attacker can read any information sent between publishers, subscribers and the broker.

3.2.2 Wired Communication

Ethernet

In contrast to the connection of the machines to the gateway/broker, a wired connection is used to communicate between the machines. The reason is that when using a wireless communication medium collisions can occur easily. As already shown in Section 3.2.1, the hidden node problem can lead to collisions. This can lead to a very high latency when transmitting data due to retransmissions and random delays when the medium is occupied. Such a high latency is not acceptable when machines in a production process communicate as the production process must not be slowed down due to high latency in data transmission.

The machines are connected via Ethernet to transmit data. Ethernet is used here as just Ethernet ports are needed which are included in many devices. The cables are rather cheap and produced by many different vendors. Furthermore, the configuration is very simple and there are hundreds of protocols that can be used on top of Ethernet.

Ethernet provides a wide variety of data transfer rates. Depending on the cable category and the material (fiber optic or copper). The *Cat5e* cable used in this thesis provides 1000 Mbit/s. In contrast to WLAN, Ethernet uses CSMA/CD to check if the medium is occupied and avoid collisions. CSMA was already described for WLAN and is the same for Ethernet. CD is a shortcut for collision detection. The detection of collisions is achieved by listening to the medium while sending. If the data recognized while listening is the same as the sent one, no collision occurred. Otherwise, a *Jam* signal is sent by the node that detected the collision to inform other nodes about it. Then the sender waits for a short, random time period and tries to resend the data if the medium is free.

WebSocket

WebSockets provide a bidirectional, persistent connection between client and server using push to send data to each other. As WebSockets provide very low latency and a very small protocol overhead once the connection was established. It is suitable for data transmission between machines connected via Ethernet. Low latency was the most important criteria when choosing a communication protocol as the production process must not be slowed down due to high latency when transmitting data between machines.

When communicating via WebSockets, the first step is to connect the client to the server using a handshake. First, the client sends a HTTP message to the server initiating the communication vie WebSockets. The following list shows the most important fields when sending the message from client to server:

- **GET /test HTTP/1.1**: A HTTP GET request with HTTP version 1.1 on resource *test*
- **Host: example.server.com**: URL of the server to connect to
- **Connection: Upgrade**: Set connection to *Update* to signal the server that the current connection should be changed
- **Upgrade: websocket**: Signals the server that WebSocket should be used to communicate
- **Sec-WebSocket-Key: d763kvnsd93ladf+238f**: Random value the server uses to verify that the message was read by the server

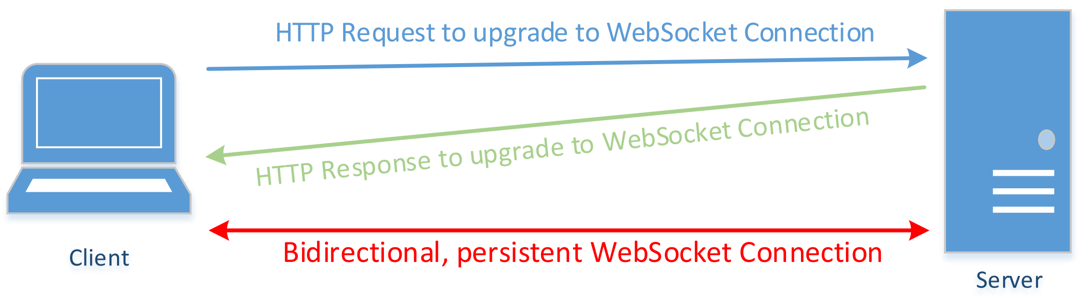
HTTP Request to upgrade to WebSocket Connection
HTTP Response to upgrade to WebSocket Connection
Bidirectional, persistent WebSocket Connection
Client
Server

Figure 3.6: Handshake scenario when establishing a WebSocket connection between client and server.

The answer by the server is also a HTTP message with a response code telling the client if the communication now uses WebSockets. In the following list the most important fields of the server response can be seen.

- **HTTP/1.1 101 Switching Protocols**: Response code 101 means that the WebSocket connection can be used from now on
- **Upgrade: websocket**: Acknowledging the use of Websockets
- **Connection: Upgrade**: Acknowledging the upgrade of the connection

- **Sec-WebSocket-Accept**: The server adds his global unique identifier to the random value sent by the client, hashes is using SHA1 and sends it back to the client. The client can verify if his random value was included in the hash to be sure that the server got his upgrade request.

After that, the connection is a bidirectional, persistent connection between client and server. This process is illustrated in Figure 3.6. WebSockets do also provide the possibility to establish a connection secured by TLS. Therefore, port 443 instead of 80 needs to be used. The upgrade request and response have to be sent using HTTPS to enable a secure connection.

As WebSockets will be used to communicate between machines using Ethernet as it provides a very low protocol overhead after the connection was established and it furthermore, would provide real-time communication which is very important for the production flow in a smart factory environment.

3.3 Security Mechanisms

As different communication approaches are used, different security mechanisms need to be applied to ensure data security. The used techniques are discussed in detail in this section. The security requirements of confidentiality, integrity and authentication for the wireless communication are ensured by using TLS and the requirements integrity and authentication for the wired communication are ensured by using ECDSA.

3.3.1 Wireless Communication Security - TLS

TLS was already mentioned in Section 2.3 and will be described here in detail. TLS is a cryptographic protocol used for secure data transmission. It provides confidentiality by using a symmetric encryption algorithm after the connection was established. The keys used for the symmetric encryption are exchanged in a previous step using asymmetric cryptography. TLS optionally ensures authenticity as the communication partners exchange certificates to determine their identity. Furthermore, integrity is provided as a *message authentication code* [58] is used to ensure that the data was not modified or that no packets are missing during transmission.

The *message authentication code* (MAC) ensures that a message was sent from the stated sender and that it was not modified during transport. It uses a secret key and a input message of arbitrary length to compute a MAC. The receiver of the message can check if the MAC is valid if he has the secret key the MAC was generated with.

With the provided security features, TLS is able to guarantee a secure connection between authenticated partners where no message can be manipulated while it is transmitted. The steps that need to be taken to create a TLS connection are:

- A client sends a *ClientHello* message to the server containing the highest TLS version it supports, the possible cipher suites, a suggested compression method and a random number.
- The server sends a *ServerHello* message to the client containing the chosen TLS version, a random number, the chosen cipher suite and compression method. The server also sends a certificate and his public key, and can optionally request a certificate from the client and checks it. The server, furthermore, sends a *ServerKeyExchange* message depending on the cipher suite to initiate the asymmetric key exchange. Finally, the server sends a *ServerHelloDone* message to the client.
- The client responds with the certificate if it was requested and checks the certificate from the server. Then the client sends a *ClientKeyExchange* containing a *PreMasterSecret* encrypted with the public key of the server. The

PreMasterSecret is later used to compute the shared secret for the symmetric encryption algorithm.

- Afterwards, the client sends a *CertificateVerify* message containing a signature over the previous messages using the private key of the client. The signature can be verified by the server with the client's private key to ensure that the client has access to the private key.
- Both client and server use the *PreMasterSecret* and the random numbers exchanged in the beginning to compute the shared *MasterSecret*.
- After that, the client sends a *ChangeCipherSpec* message to indicate that the communication will be encrypted from now on. Then the client sends an encrypted *Finished* message containing a hash and MAC of the handshake messages to indicate that the handshake is done now and the encrypted communication can start.
- The server then decrypts and verifies the sent MAC and sends a *ChangeCipherSpec* message if the MAC is valid to indicate that his messages are encrypted from now on. Then he also sends an encrypted *Finished* message.
- The client also decrypts and verifies the message from the server and if the MAC is valid, the handshake is completed.

After the completion of the handshake, the messages are encrypted with a symmetric algorithm using the computed *MasterSecret*. Figure 3.7 shows the main messages and parameters exchanged when performing the TLS handshake.

In this thesis, TLS version 1.2 using SHA-256 as hashing function, Diffie-Hellman Key Exchange as asymmetric cryptography algorithm and AES as a symmetric encryption scheme are used. All named algorithms were already described in Section 2.3 and will not be explained again.

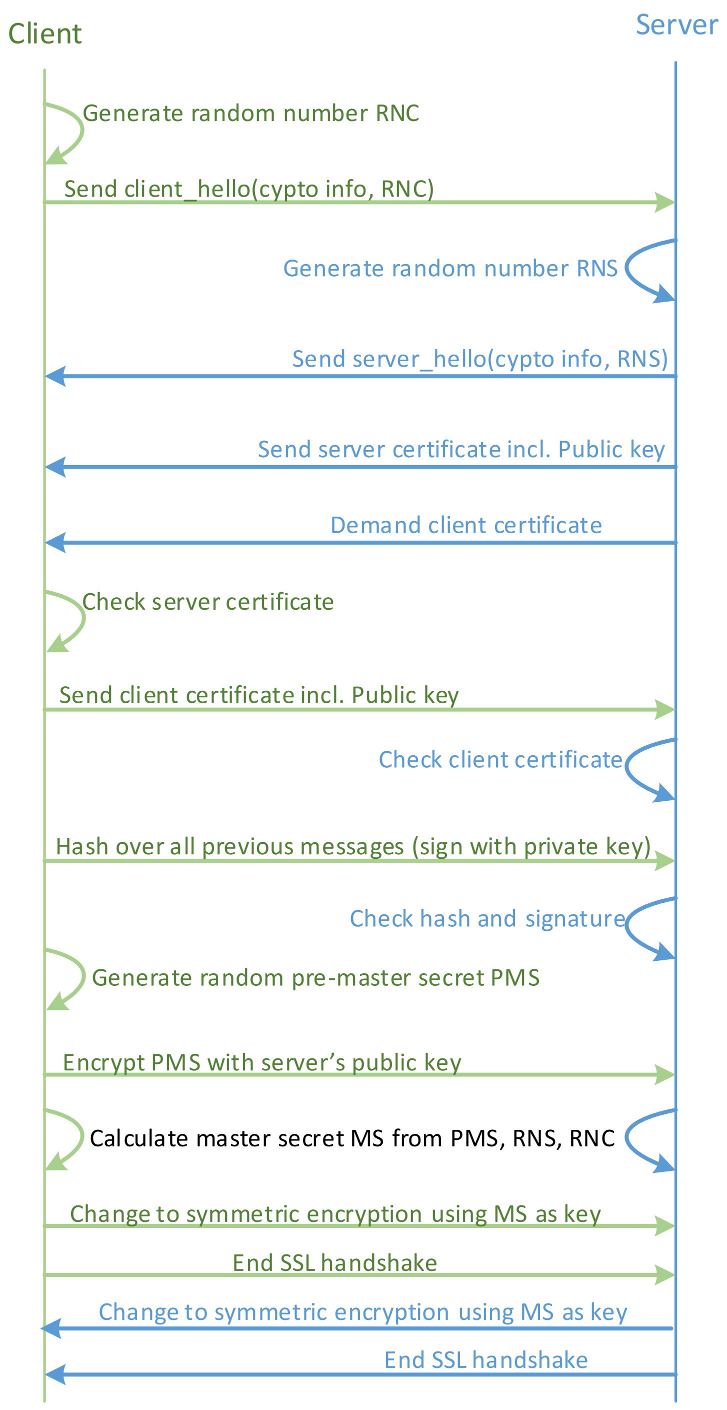
Client
Server
Generate random number RNC
Send client_hello(cypto info, RNC)
Generate random number RNS
Send server_hello(cypto info, RNS)
Send server certificate incl. Public key
Demand client certificate
Check server certificate
Send client certificate incl. Public key
Check client certificate
Hash over all previous messages (sign with private key)
Check hash and signature
Generate random pre-master secret PMS
Encrypt PMS with server's public key
Calculate master secret MS from PMS, RNS, RNC
Change to symmetric encryption using MS as key
End SSL handshake
Change to symmetric encryption using MS as key
End SSL handshake

Figure 3.7: TLS handshake in detail containing all exchanged random values, certificates and keys to establish a secure connection.

3.3.2 Wired Communication Security - ECDSA

The Elliptic Curve Digital Signature Algorithm is a variation oft the Digital Signature Algorithm (DSA) using the elliptic curve discrete logarithm problem (ECDLP) instead of the discrete logarithm problem. The basic principle of ECDSA were shortly discussed in section 2.3.

To sign a message, the two parties need to agree on the *curve*, a base point G of prime order on the curve and n, the multiplicative order of G. With those parameters, the sender generates a key pair by selecting a random integer d_a and a public key curve point $Q_a = d_a \times G$. After that, the sender computes a hash of the message, chooses a random, private integer and uses the last bits of the hashed message to generate a signature pair (r, s) and sends it with the message to the receiver.

The receiver can verify the signature by knowing the public key curve point Q_a as well as the *curve*, G and n. The receiver verifies the curve point Q_a to not be equal zero, lie on the curve and when performing an elliptic curve multiplication with n is not the identity element. After that the receiver computes the a hash of the message, computes the value r of the signature pair and checks if the computed r is equal to the r sent by the sender.

A detailed illustration of signing and verifying can be seen in Figure 3.8. It contains all necessary calculations and exchanged parameters.

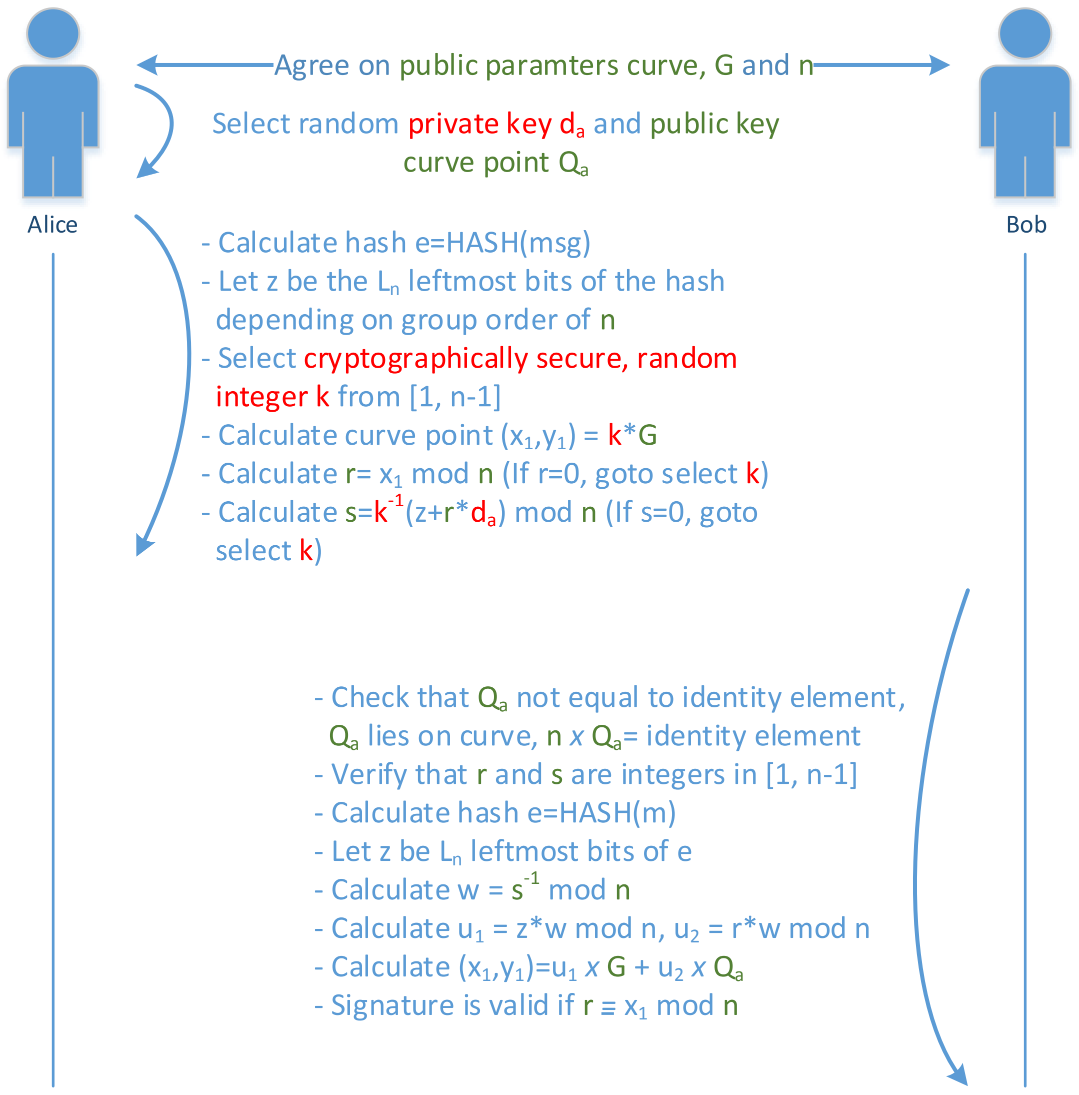
Agree on public paramters curve, G and n
Select random private key d_a and public key curve point Q_a
Alice
Bob
- Calculate hash e=HASH(msg)
- Let z be the L_n leftmost bits of the hash depending on group order of n
- Select cryptographically secure, random integer k from [1, n-1]
- Calculate curve point $(x_1,y_1) = k*G$
- Calculate r= x_1 mod n (If r=0, goto select k)
- Calculate $s=k^{-1}(z+r*d_a)$ mod n (If s=0, goto select k)
- Check that Q_a not equal to identity element, Q_a lies on curve, n x Q_a= identity element
- Verify that r and s are integers in [1, n-1]
- Calculate hash e=HASH(m)
- Let z be L_n leftmost bits of e
- Calculate $w = s^{-1}$ mod n
- Calculate u_1 = z*w mod n, u_2 = r*w mod n
- Calculate $(x_1,y_1)=u_1\ x\ G + u_2\ x\ Q_a$
- Signature is valid if $r \equiv x_1$ mod n

Figure 3.8: A detailed description of the ECDSA with generation and verification of a signature. The red marked parameter are private and the green marked parameter indicate that they are public.

4 Implementation

In this chapter, the implementation of the previously described system will be discussed. The used hardware will be shown and the software implementation with class diagrams and code snippets as well as the used libraries and the development environment will be depicted.

4.1 Hardware

To demonstrate the given architecture, three Raspberry Pi 3 [59] where used as they provide on-board WLAN. Figure 4.1 shows an image of the Raspberry Pi 3.

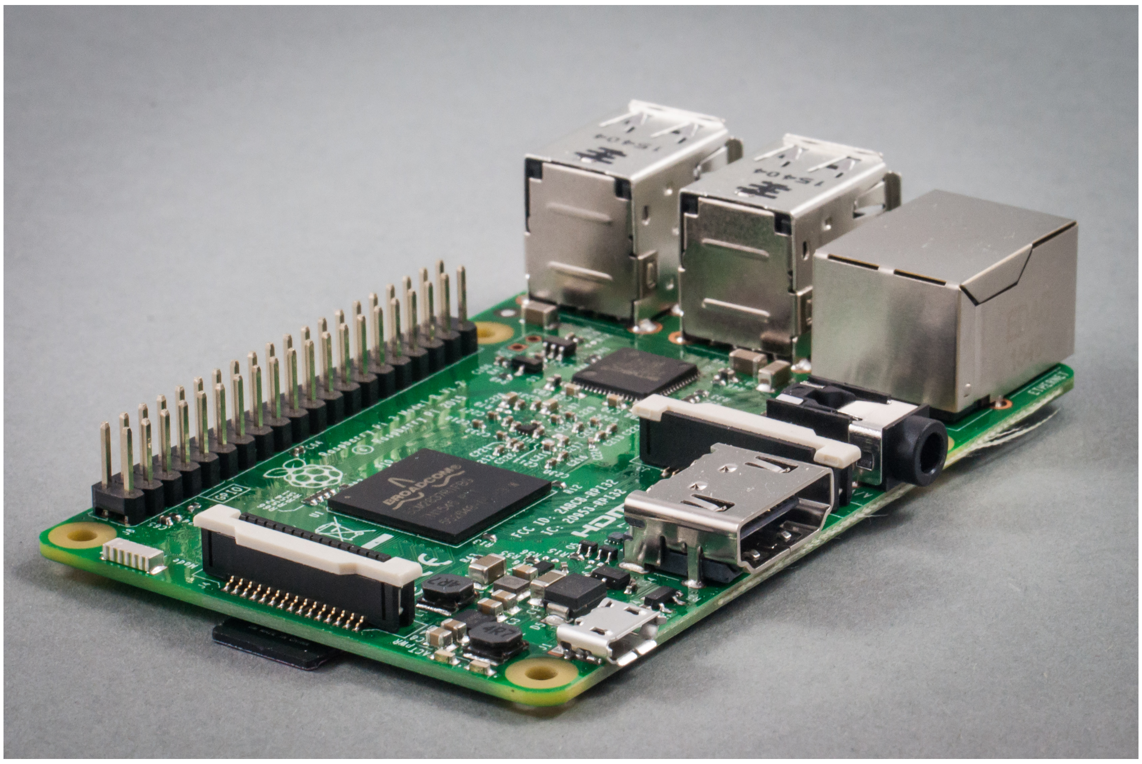

Figure 4.1: Raspberry Pi 3 board with 4 USB ports and WLAN onboard.

The most important technical specifications are given in the following list:

- 1.2 GHz 64-bit quad-core ARMv8 CPU
- 1 GB RAM
- 802.11n WLAN
- 4 USB ports
- 1 Ethernet port
- Bluetooth 4.1/BLE

Each Pi was equipped with a micro USB card with the operating system *Raspbian Jessie* version May 2016 [60] which is a Debian Linux-based operating system especially developed for Raspberry Pi devices. Two of the Pis were used to simulate machines and the third one was used as a broker. All three Pis provide identical hardware.

The hardware setup for the demonstration of the previously described architecture can be see in Figure 4.2. The two Pis simulating the machines are connected using an Ethernet cable and the third Pi which is the gateway/broker is connected to the two machines using WLAN. The gateway/broker Pi acts as WLAN access points in this case. Both machines connect to the broker's offered WLAN for communication using a password. This prevents other clients from connecting to the broker's access point. Furthermore, the number of connections to the access point was limited to three. Two connections for the machines and one for the desktop PC to be able to connect to the machines and the broker remotely.

Modifications on the Raspberry Pi

For the broker, some files had to be modified to enable the Raspberry Pi to act as a WLAN access point. The following list contains all modified files and also shows the necessary changes:

- The first step is to install `hostapd` on the Pi with the command `sudo apt-get install hostapd`. This package is necessary to allow the build-in WiFi module to act as an access point.
- The next step is to ignore the `wlan0` interface in the file `/etc/dhcpcd.conf` by inserting the line `denyinterfaces wlan0` at the bottom of the file but above any `interface` lines.
- After that, the interface configuration file needs to be changed in order to set a static IP address for the access point. This is done by changing the `wlan0` section in the file `/etc/network/interfaces`. The following excerpt shows the necessary changes and additions:

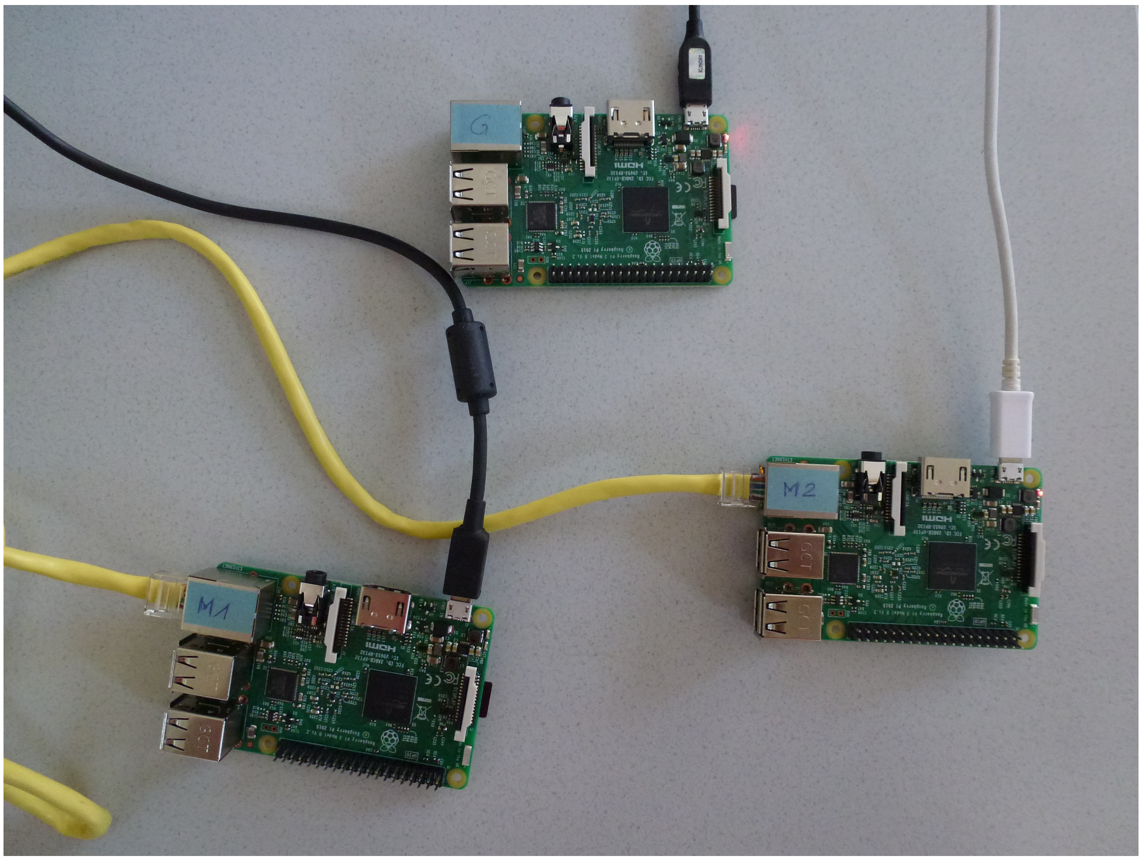

Figure 4.2: Hardware setup for demonstration purposes of the architecture. The two lower Pis are simulating the machines, the upper Pi is used as gateway/broker.

```
allow-hotplug wlan0
iface wlan0 inet static
  address 172.24.1.1
  netmask 255.255.255.0
  network 172.24.1.0
  broadcast 172.24.1.255
#  wpa-conf /etc/wpa_supplicant/wpa_supplicant.conf
```

The configuration sets a fixed IP address for the device with a corresponding network mask, the network address for the WLAN and the broadcast address. To enable the new configuration, several commands need to be called:

- `sudo service dhcpcd restart`
- `sudo ifdown wlan0`
- `sudo ifup wlan0`

After these commands the new configuration for the interface `wlan0` is used.

- The final step is to configure the previously installed `hostapd` package. The configuration file `/etc/hostapd/hostapd.conf` has to be created and the most important parts are shown in the following snippet:

  ```
  1  interface=wlan0
  2
  3  driver=nl80211
  4
  5  ssid=Broker
  6  hw_mode=g
  7  ieee80211n=1
  8  macaddr_acl=0
  9
  10 auth_algs=1
  11 ignore_broadcast_ssid=0
  12 wpa=2
  13 wpa_key_mgmt=WPA-PSK
  14 wpa_passphrase=raspberry
  15 rsn_pairwise=CCMP
  ```

  The statement in line 1 defines the interface affected by the configuration. Line 3 shows the driver which enables the WLAN module to change to the *Access Point* mode (AP).

  Between lines 5 and 8 the general settings of the offered WLAN are set:

  - The `ssid` gives the WLAN name.
  - The `hw_mode` defines that the 2.4 GHz frequency band is used.
  - `ieee80211n=1` means that the n-standard for this WLAN is used. It enables a data rate of 600 Mbit/s.
  - The command `macaddr_acl=0` means that all MAC addresses are accepted. This feature was not used as it would have meant that the broker needs to be changed whenever a machine is changed or another desktop PC is used to remotely connect to the broker and machines.

  Line 10 to 15 contain the authentication credentials to access the WLAN.

  - The command `auth_algs=1` indicates that WPA authentication must be used.
  - `ignore_broadcast_ssid=0` means that the WLAN will be visible and clients do not need to know the SSID. This could be used to protect a WLAN from being seen in public.
  - The statement `wap=2` means that WPA2 is used and `wpa_key_mgmt=WPA-PSK` indicates that a pre-shared key has to be known by the clients to gain access to the WLAN.
  - `wpa_passphrase` is the previously described pre-shared key and can be chosen by the creator of the WLAN.

- `rsn_pairwise=CCMP` means that AES is used as symmetric encryption scheme.

`Hostapd` is starts automatically when the Raspberry boots. To know where to look for the configuration, the line `DAEMON_CONF="/etc/hostapd/hostapd.conf"` needs to be added to the file `/etc/default/hostapd`.

4.2 Implementation of the Hybrid Communication Approach

In this section, the used software libraries and development environment will be discussed. The actual implementation on the previously shown hardware will be depicted for the wired and wireless communication approach.

Before describing the implementation and libraries in detail, the class diagram of a machine can be seen in Figure 4.3. For the broker, no class diagram will be shown as only one class is used that will be described in detail in Section 4.2.2.

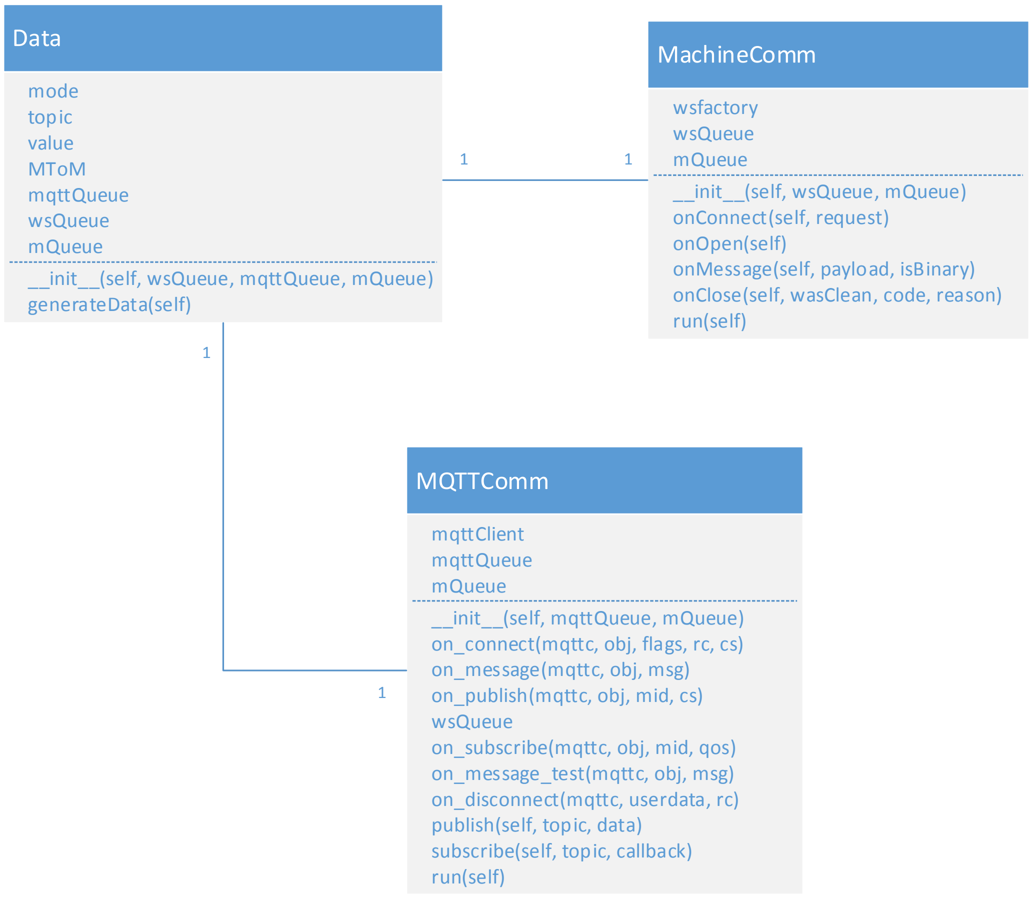
Data
mode
topic
value
MToM
mqttQueue
wsQueue
mQueue
__init__(self, wsQueue, mqttQueue, mQueue)
generateData(self)
1
1
MachineComm
wsfactory
wsQueue
mQueue
__init__(self, wsQueue, mQueue)
onConnect(self, request)
onOpen(self)
onMessage(self, payload, isBinary)
onClose(self, wasClean, code, reason)
run(self)
1
1
MQTTComm
mqttClient
mqttQueue
mQueue
__init__(self, mqttQueue, mQueue)
on_connect(mqttc, obj, flags, rc, cs)
on_message(mqttc, obj, msg)
on_publish(mqttc, obj, mid, cs)
wsQueue
on_subscribe(mqttc, obj, mid, qos)
on_message_test(mqttc, obj, msg)
on_disconnect(mqttc, userdata, rc)
publish(self, topic, data)
subscribe(self, topic, callback)
run(self)

Figure 4.3: Class diagram of the machine showning the MQTT and WebSocket communication.

4.2.1 Libraries and Development Environment

Generally, the used programming language is **Python** version 2.7 on the clients and version 3.5 on the broker. The different versions are needed as different libraries are used on broker and machines. Python is a very popular programming language as it provides many libraries and supports multiple programming paradigms such as object-oriented or procedural programming. To be able to implement the proposed hybrid communication approach, several libraries had to be used. The most important ones are contained in the following list:

- `Threading:` This library enables multi-threading in python. It is needed on the machine as it has to deal with wired and wireless communication. Therefore, several threads are started on each machine to communicate with the broker and with the machine.
- `Queue:` The generated data on the machine can be sent using both the wired and wireless communication. As threads are used, it is necessary to use the `Queue` library to be able to pass the same data to several threads. This library enables safe data exchange between threads. Queues are used for both communication scenarios.
- `Paho-MQTT:` This library is used to implement the MQTT client on the machine. It enables a programmer to connect to a MQTT server, publish data and subscribe to topics. It also provides the possibility to encrypt the sent data using TLS to ensure confidentiality, integrity and authenticity. As it just supports MQTT, it is used for the wireless communication between broker and client.
- `Twisted:` Twisted is an asynchronous networking framework that especially focuses on event-based programming. It is needed to enable the implementation of a WebSocket connection between the machines. It is used for the wired communication exclusively.
- `Autobahn:` This library provides client and server functionality to implement WebSockets. It uses `Twisted`, in this case, as a basis for event-based programming. This library is used on the machines to implement both server and client of the WebSocket used for the wired communication.
- `ECDSA:` This library is used on the machines to sign and verify the signature of the data sent between the machines. It provides 5 different elliptic curves with different key lengths. ECDSA is used for the wired communication to ensure integrity and authenticity.
- `HBMQTT:` The broker for the MQTT communication was implemented using `HBMQTT`. This library provides different levels of *Quality of Service* and also supports TLS to secure the distributed data. This library requires Python

version 3.4 or higher as it is built on top of the library `asyncio` which was introduced in version 3.4.

The used development environment is **PyCharm** by **JetBrains**. This IDE was especially designed to implement Python and provides every necessary functionality to implement a Python program, such as import of libraries, auto-completion, or debugging with breakpoints.

After implementing the programs for the machines and the broker in **PyCharm** on a desktop PC, they are transferred to the Raspberry Pis using a **GIT** repository. This Python programs consist of one file for the machines and one for the broker, and can then be started using the console user interface with the command `python machine.py` or `python broker.py`.

4.2.2 Software Implementation

In this section, the implementation of the MQTT broker and the MQTT and WebSocket client on the machines will be shown. First, the broker will be depicted and the used configuration files and needed certificates for the TLS-secured communication will be shown. Then, the client implemented on the machine will be discussed. Both, the WebSocket and MQTT implementation will be depicted in detail.

Broker

As a broker, the previously mentioned **HBMQTT** broker is used. The implementation was rather easy as only the basic functionality with TLS encryption is used. The very simple broker file is shown in the following code snippet:

```
 1 from hbmqtt.broker import Broker
 2 
 3 config = {
 4     'listeners': {
 5         'default': {
 6             'type': 'tcp',
 7             'bind': '10.0.0.50:8885',
 8             'ssl': 'on',
 9             'cafile': './cert/ca.crt',
10             'capath': './cert',
11             'certfile': './cert/broker.crt',
12             'keyfile': './cert/broker.key',
13         },
14     },
15     'sys_interval': 10,
16     'auth': {
17         'allow-anonymous': 'true'
18     }
19 }
20 
21 broker = Broker(config)
22 broker.start()
```

Between line 3 and 16 the necessary configuration to enable a TLS connection is shown. The configuration defines the IP address and port the broker listens on, and the necessary certificate authority's certificate (CA certificate), broker certificate and broker key to ensure a secure connection. The CA certificate is needed to check if the clients certificate is valid and signed by a trusted authority. The broker certificate is sent to the broker for authentication and was also signed using the CA certificate.

The broker is simply started with the configuration as it can be seen in line 18 and 19. However, the broker's functionality could be extended by implementing so-called `Plugins` that need to be added as entry points to the broker's setup configuration.

The broker can be extended with any plugin such as storing the received data to a database. This was not necessary in this thesis as the data sent to the broker was not processed but just distributed to the machines.

The previously mentioned certificates were created manually. First, a self-signed X509 CA certificate and a key were created using the following command:

```
openssl req -new -x509 -days 100 -extensions v3_ca -keyout
     ca.key -out ca.crt
```

To create a broker's certificate, first, a key needs to be generated with the following command:

```
openssl genrsa -out broker.key 2048
```

Now, a certificate signing request with the generated key needs to be generated. This can be used to generate a certificate using the generated CA certificate.

```
openssl req -out broker.csr -key broker.key -new
```

Finally, the created signing request can be signed using the CA key and a broker certificate is created.

```
openssl x509 -req -in broker.csr -CA ca.crt -CAkey ca.key
     -CAcreateserial -out broker.crt -days 100
```

The created broker certificate, key and CA certificate are stored on the broker and can be used in the TLS communication between clients and broker. The same CA certificate can be used to sign the client certificates too.

Client

In contrast to the broker, the client performs several tasks. It needs to communicate with the broker using MQTT, it communicates with other machines using WebSockets and it provides data that is sent to other communication partners or was received from them. Therefore, the client's implementation uses threads. Threads enable the program to perform different task separately. However, in this case, the threads need to share data between each other with is enabled by using **Queues**. A queue is a thread-safe method to distribute data between threads. The client needs three queues which can be seen in the following code snippet.

```
q_mqtt = Queue.Queue()  # queue MQTT data
q_ws = Queue.Queue()  # queue WebSocket data
q_m = Queue.Queue()  # queue machine data
```

They are initialized globally and passed to every thread when it is initialized. After that the threads need to be started which can be seen in the following code snippet.

```
wscomm = WSThread(q_ws, q_m)
datacomm = MachineThread(q_ws, q_mqtt, q_m)
mqcomm = MQThread(sys.argv[1], q_mqtt, q_m)

print("Start Machine")
datacomm.start()
print("Start MQTT")
mqcomm.start()
print("Start WS")
wscomm.start()
```

As it can be seen, the WebSocket thread receives the queue containing WebSocket data (**WS queue**) and the one containing machine data (**M queue**). The **WS queue** is just read by the WebSocket thread as it contains data that was generated by the machine and should be sent to another machine. The **M queue** is filled by the WS thread as it contains data that was received via the WebSocket connection and should be passed to the machine.

For the MQTT thread the behavior is similar. The queue containing MQTT data (**MQTT queue**) and the **M queue** are passed to the thread. The received data is pushed into the M queue by the MQTT thread and the data that should be published is pushed to the **MQTT queue** from the data thread.

The MQTT thread, in contrast to the other threads, also receives a parameter `sys.argv[1]`. This parameter contains the unique machine name that is needed for the MQTT connection.

The `init` method of the MQTT thread can be seen in the following code snippet:

```
1       def __init__(self, name, qps, qsock):
2           threading.Thread.__init__(self)
3           self.client = mqtt.Client(client_id="Machine" +
                 str(name))
4           self.mqttQueue = qps
5           self.mQueue = qsock
6           self.client.on_message = self.on_message
7           self.client.on_connect = self.on_connect
8           self.client.on_publish = self.on_publish
9           self.client.on_subscribe = self.on_subscribe
10          self.client.on_disconnect = self.on_disconnect
11          if (str(name).endswith('1')):
12              cert = "./cert/machine1.crt"
13              key = "./cert/machine1.key"
14          else:
15              cert = "./cert/machine2.crt"
16              key = "./cert/machine2.key"
17          self.client.tls_set(ca_certs="./cert/ca.crt",
                 certfile=cert, keyfile=key,
18                                cert_reqs=ssl.CERT_REQUIRED,
19                                tls_version=ssl.PROTOCOL_TLSv1_2,
                                       ciphers=None)
20          self.client.connect("10.0.0.50", 8883, 60)
```

As it can be seen in line 3, the machine name is set according to the passed parameter. Between line 6 and 10 the corresponding methods are registered to the callbacks. Between line 11 and 18 the current machine is identified according to the passed name and the corresponding certificates and keys needed for the TLS connection are specified. In line 19 the TLS parameters are set. It can be seen that the certificates are passed to the method. Furthermore, it is specified that the broker must offer a certificate and that the used TLS version will be 1.2. The parameter `cipher=None` indicates that all possible cipher suites available on the client are offered to the broker. Finally, in line 20, the connection to the broker is established. Usually, MQTT uses port 1883 but when using TLS port 8883 must be used.

After the connection was established the thread runs in an endless loop to publish data which can be seen in the following code snippet:

```
1               if not self.mqttQueue.empty():
2                   element = self.mqttQueue.get()
3                   if element.mode == 'Pub':
4                       print ('Publish data')
5                       self.publish(element.topic,
                            element.value)
6                   if element.mode == 'Sub':
7                       print ('Subscribe to topic')
8                       self.subscribe(element.topic,
                            'on_message_client')
```

It can be seen in line 2 that actions are performed when the **MQTT queue** is not empty. The **MQTT queue** contains both data that should be published and topics the machine should subscribe to. The queue entry is distinguished using the parameter `mode`. Line 4 to 6 show the case for mode `Pub` where data should be published. The data contained in `value` is published with the topic `topic`.

Between line 7 and 9 the scenario for mode `Sub` when subscribing to a new topic can be seen. In line 9, the subscription on a special topic can be seen. A message regarding this special topic will be processed by the callback `on_message_client`. This could be used for every possible topic to distinguish between them without checking the topic on each reception. All message that do not have such a special callback are processed in the `on_message` method where the topic needs to be checked to distinguish the messages.

When a message is received due to a previous subscription, the corresponding message callback is activated and the data is stored in the **M queue** as it can be seen in the following code snippet for the callback `on_message_client`:

```
1       def on_message_client(mqttc, obj, msg):
2           mqttc.mQueue.put(msg)
```

The WebSocket thread just takes the **WS queue** and the **M queue** as input. The `init` method can be seen in the following code snippet:

```
1       def __init__(self, qws, qsock):
2           self.factory =
                WebSocketServerFactory(u"ws://127.0.0.1:9000")
3           self.factory.protocol = WebSockProt
4           self.factory.wsQueue = qws
5           self.factory.mQueue = qsock
```

In line 2 the factory for creating a WebSocket can be seen. It defines the IP address and the port the server should open and listen too. The client uses the same information to connect to the server. The defined protocol in line 3 references to the class `WebSockProt` which contains the actual functionality of the WebSocket.

WebSockets are based on a client-server architecture which means that one machine needs to be the client and the other one the server. In this implementation it was fixed which machine is the server and which one is the client. Another possibility would have been that the broker decides which one should be the server and which one the client. In principle, the server and client are very similar. The `run` method of the server can be seen in the following code snippet:

```
    def run(self):
        reactor.listenTCP(9000, self.factory)
        reactor.run()
```

The client's `run` method, in contrast, looks like this:

```
    def run(self):
        reactor.connectTCP("127.0.0.1", 9000, self.factory)
        reactor.run()
```

It can be seen that the server opens port `9000` and listens to it and the client connects to the IP address and port using TCP.

Sending data is equal for both sides but it is done in different methods. For the client, sending data is performed in the `onOpen` method. This method is called after calling `connectTCP(...)` was successful. For the server, sending is done in the `onConnect` method as this one is called when a client connects to the server. The code for sending data with the client can be seen in the following code snippet:

```
        def sendData():
            if not self.factory.q_ser.empty():
                element = self.factory.mQueue.get()
                signature = sk.sign(element)
                element.signature = signature
                self.sendMessage(element.encode('utf8'))
            self.factory.reactor.callLater(0.1, sendData)

        sendData()
```

As it can be seen, the client periodically tries to send data if something is available.

Furthermore, in line 7 the signature of the message is generated and added to the data in line 8.

When receiving a message, the callback `onMessage` is called. It can be seen in the following code snippet:

```
1       def onMessage(self, payload, isBinary):
2           data = payload.decode('utf8')
3           if vk.verify(data.signature, data.value):
4               self.factory.mQueue.put(data.value)
```

As it can be seen in line 2, the data is decoded and in line 3 the signature is verified with the public key of the sender. If the signature is valid, the data can be used by the machine. To generate a private and public key for signatures, the two commands shown in the next code snippet need to be executed. The verification key must be sent to the receiver of signed messages. The elliptic curve, in this case, is a NIST curve with a key length of 192 bit.

5 Evaluation

In this chapter, the risk and security analysis as well as the evaluation regarding package overhead due to the security mechanisms will be discussed. As the data transferred between the machines and the broker needs to be secured to prevent eavesdropping or manipulation of the data, the used security mechanisms need to be evaluated regarding their ability to secure the data.

Security mechanisms enlarge the size of a package due to signatures or encrypted data which might cause problems when transmitting data as large packages might get dropped easier in a lossy network. Therefore, the proposed approach is evaluated regarding the enlargement of the package due to the used security mechanism.

5.1 Security Analysis

Due to the fact, that two different communication scenarios exist it is necessary to analyze the risks for each scenario. First, the principles of a security analysis will be explained. After that the security analysis for the proposed hybrid communication approach will be discussed.

5.1.1 Principles of Security Analysis

A security analysis is performed before starting to design and implement any concept that aims to secure some objective and afterwards to evaluate if the chosen security mechanisms protect the objectives. In their paper, Myagmar et al. [61] show how to analyze a hard- or software concept regarding security risks. They propose a three step model for security engineering. The first step is threat modeling where each possible threat is identified. The second step is to specify the security requirements which specify what should be protected. The final step is to implement the mechanisms that fulfill the security requirements and therefore, solve every threat.

Reimair et al. [62] show the security analysis of a management system for cryptographic keys on a mobile devices. They state that a security analysis for data transmission contains the following steps:

1. Understand the analyzed system and domain.
2. Identify the assets and active entities that could cause a threat.

3. Identify the security requirements (confidentiality, integrity, ...) of the system and choose appropriate security mechanisms.
4. Identify the threats caused by each entity and the asset they threaten.
5. After implementation of the security mechanisms, check the identified threat and find countermeasures or residual risks.
6. Check if all threats could be solved appropriately by the implemented security mechanisms. If not, choose other/more security mechanisms and continue from 5.

5.1.2 Threat Analysis of the Hybrid Communication Approach

In this section, the security analysis of the proposed communication concept will be discussed. It is divided into several parts. First, the definition of the entities and assumptions will be given. Second, the assets will be described and finally, the threats, their countermeasures or residual risks will be discussed.

Entities

As already discussed in the previous section, the first step is to identify the **entities (E)** interacting with each other. These entities contain the devices that could possibly be attacked and also possible attackers.

(E1) Broker: The broker is used to distribute non-time critical but confidential data between machines or production lines, and also stores maintenance data of the machines in the data cloud or computation cloud to enable the cloud to provide useful information to employees. The broker uses a secure element to store the cryptographic keys in a hardware security module(HSM) providing temper resistance.

(E2) Machine: The machines communicate to each other in cases where fast data transmission is necessary to enable a fast, fluent production process or react to failures. The data sent between machines is non-confidential. Furthermore, they send confidential information such as maintenance data or non-time critical data to the broker.

(E3) Security Controller: The security controller is used by the machines to sign the data sent between the machines. It also contains the cryptographic keys in a HSM to ensure that no one can access the key.

(E4) Security Controller Vendor: The vendor of the security controller ensures that the controller provides all necessary security mechanisms and that the keys are stored in a HSM. He also provides the initial key.

(E5) **Machine Vendor:** The machine vendor adds the security controller to the machines and ensures that the services provided by the controller are used properly. He might also change the key on the security controller.

(E6) **Factory:** The factory and its owner buy machines from the machine vendor and try to protect their data and production secrets from trespassers and other malicious influences. He might accidentally cause a threat.

(E7) **Trespasser:** The trespasser tries to access confidential data sent between the machines and broker or insert failures or his own into any of the devices. He might also try to attack the wireless communication module to destroy the communication between the devices or try to gain physical access to the machines.

(E8) **Maintenance Worker:** The maintenance worker has physical access to the machines on the production floor and can change the key of the security controller. He could also manipulate the machine such as trying to sent data to the machine or read data from the machine. He might perform similar actions as a **trespasser**.

Assumptions

Before analyzing the communication concept regarding security threats, some assumptions need to be made. The reason is that it would exceed the scope of this thesis to analyze each and every possible scenario. The following list contains all assumptions regarding, for example, trustworthiness that will be used in the security analysis.

- The security controller vendor is rated as trustworthy as his reputation would get damaged if his controller would not guarantee the specified level of security.
- The machine vendor is not seen as trustworthy as he might try to add backdoors to the machine to retrieve confidential information.
- The factory and their owners are assumed to be trustworthy as they want to secure confidential data and will, therefore, not try to attack or weaken the security concepts in the factory.
- The maintenance worker is not assumed to be trustworthy as he is not a member of the factory but of the machine vendor and might want to steal confidential data. He is considered as a possible attacker. The maintenance worker is assumed to attack both wired and wireless communication scenarios as he can easily gain physical access.
- The trespasser is the attacker in this scenario and tries everything to steal confidential data, destroy the communication possibility or retrieve the key

from any security controller. It is also assumed that trespasser mainly attack the wireless data transfer as gaining physical access is a hard task.

- It is assumed that each machine contains a HSM and a security controller capable of creating digital signatures.
- The scenarios regarding how to store new cryptographic keys in the security controllers will not be discussed here as this would exceed the scope of this thesis and therefore, no security analysis will be done for these scenarios. It is assumed that the keys were already present on the machines and broker.
- The certificates of brokers and machines are stored in a HSM and are assumed to be trustworthy and signed by a trusted certificate authority. A company might sign the certificates used in the factory by themselves to prevent misuse of their certificates.

Assets

The next step in the risk analysis is to identify the **assets (A)**. These are the objectives that should be protected from possible misuse by an adversary and can be seen in the following list:

(A1) Wired Machine-to-Machine Data: The data transmitted between machines is necessary to ensure a fast, fluent production process such as informing the next machine about the state of the currently manipulated product. The transmission needs to ensure integrity and authenticity.

(A2) Wireless Machine-to-Broker/Broker-to-Machine Data: The data transmitted between machines and brokers is non-time critical and contains maintenance data or production flow data used by employees to predict throughput, date of machine maintenance or machine performances. Furthermore, data from other production lines or from employees might be transmitted to the machines using the broker. The data transmission must provide confidentiality, integrity and authentication.

(A3) Cryptographic Keys: The keys stored on the machines and also on the broker to en- and decrypt data. They can only be changed by authorized staff and must not be revealed with any kind of attack or threat.

Threats, Countermeasures and Residual Risks

Finally, the **threats (T)** of each active **entity (E)**, their **countermeasures (C)**, and their **residual risks (R)** regarding the **assets (A)** are discussed. The **threats (T)** contain possible threats and also attacks on different **entities(E)**.

The following list gives the threats for each entity as well as possible countermeasures or residual risks. Furthermore, the asset compromised by each threat will be given. Each entity has one or more threat and each threat has either a countermeasure, a residual risk or both. The cohesiveness of a threat, countermeasure and/or residual risk are indicated by different abbreviations and equal numbers. The compromised assets are listed for each threat.

(E4) Security Controller Vendor.

(T1) The security controller vendor could unintentionally leave backdoors on the controller such as debugging interfaces (software) or special debugging pins (hardware) on the security controller open. This would give trespasser the opportunity to break the controller and reveal the cryptographic keys.

Assets compromised: **(A1)**, **(A2)**, **(A3)**

(C1) Security controllers can be certified by a trusted third party authority that ensures that no such backdoors are available.

(T2) The security controller vendor could provide a wrong or faulty implementation of the cyptographic algorithm. This might give trespasser an possibility to retrieve the cryptographic keys.

Assets compromised: **(A1)**, **(A2)**, **(A3)**

(C2) Best practice for algorithms must be ensured as well as secure cryptographic algorithms should be used. Furthermore, the software should be certified by a trusted third party authority.

(T3) The security controller vendor might specify a too short or weak key (influenced by e.g. weak random number generators) which might give trespasser the chance to retrieve the key due to his short length or weakness.

Assets compromised: **(A1)**, **(A2)**, **(A3)**

(C3) Specify a key length that is long enough. The length of a key depends strongly on the used algorithm.

(T4) The security controller vendor could implement a security routine or parts of it with an algorithm that can be broken such as MD5. Trespasser might be able to break the algorithm and reveal the key.

Assets compromised: **(A1)**, **(A2)**, **(A3)**

(C4) State-of-the-art security concepts need to be used regarding possibility to be broken as well as minimum key length.

(E5) Machine Vendor.

(T5) The machine vendor could fit the security controller wrongly into the machine. This might cause wrong or faulty functionality of the security controller and might give trespassers the possibility to reveal the key.

Assets compromised: **(A1)**, **(A2)**, **(A3)**

(C5) Trusted third party authorities exist that certify the machine of the machine vendor for correctly fitting the security controller into the machine.

(T6) The machine vendor might use the API to interact with the security controller wrongly or just parts of it, so that only parts of the functionality are used such as signing a message but not encrypting it which would lead to a lack of confidentiality. Trespassers could have the chance to intercept the data traffic and extract confidential information.

Assets compromised: **(A1)**, **(A2)**

(C6) The security controllers API must not allow to execute only parts of an algorithm. Furthermore, the security controller vendor must ensure that if a controller is able to perform several algorithms, the algorithm used for a specific machine is fixed and cannot be changed by a customer.

(T7) The machine vendor might change the key on the security controller to a weak but long enough one that could be computed by using, for example, the machine ID. If the factory that bought machines with such security controller would not change the cryptographic key, a trespasser might be able to retrieve the key.

Assets compromised: **(A1)**, **(A2)**, **(A3)**

(R7) There is no countermeasure for this threat. The purchasers of the machines must ensure that all keys are changed to their own ones.

(T8) The machine vendor might try to send non-encrypted confidential data using WLAN to some hidden access point. As the machines use WLAN to communicate to the broker, this would enable a trespasser to read confidential data.

Assets compromised: **(A2)**

(C8) Factories should only buy machines certified by trusted third party authorities as they ensure that the required level of security is achieved with the given machine.

(E6) Factory.

(T9) Several or all machines in the factory might use the same key set by the factory. If the key is weak, it might happen that an attacker reveals the key and could read the data from several machines and also send malicious data to several machines.

Assets compromised: **(A1)**, **(A2)**

(R9) Nothing can be done against this threat. Factories are responsible themselves to ensure the strength of the key and the usage of different keys for different machines.

(T10) The cryptographic keys of one or more devices might be available on a non-HSM device. This could allow attackers to get the keys and reveal confidential information or inject failures into the system.

Assets compromised: **(A1)**, **(A2)**, **(A3)**

(R10) No countermeasure exist to prevent this threat. The factory needs to ensure that the keys are exclusively stored on HSMs.

(T11) The physical access to different areas of the factory might not be permitted for unauthorized persons. This could enable trespassers or attackers to access the factory physically and destroy devices or perform other unwanted actions.

Assets compromised: **(A1)**, **(A2)**

(R11) No countermeasure in form of a security mechanism exists. The factory must prevent unauthorized persons from entering the factory by using, for example, access control systems. The keys can be secured by storing them exclusively on HSMs.

(E7) Trespasser. As the trespasser is assumed to be mainly able to attack the wireless communication, the list threats affect the wireless communication approach.

(T12) The trespasser could perform a Denial of Service (DoS) attack [45] on the wireless interfaces of the broker or the machines. Machines and broker would not be able to communicate using WLAN anymore.

Assets compromised: **(A2)**

(R12) There is nothing that can be done against DoS attacks.

(T13) Trespassers could try to gain physical access to the production floor or other parts of the factory. This could cause destruction of devices (Dos).

Assets compromised: **(A1)**, **(A2)**

(R13) There exists no countermeasure in form of a security mechanism. Factory owner must ensure that no unauthorized persons can enter the factory by using, for example, access control systems.

(T14) A trespasser might eavesdrop on the wireless communication between machines and brokers and steal confidential data.

Assets compromised: **(A2)**

(C14) As the wireless communication is encrypted using TLS, the attacker is not able to steal sensitive data.

(T15) Trespasser might try to perform a Man-In-The-Middle attack on the communication between broker and machine when the communication is set up. This would enable the attacker to read all traffic sent between the machine and broker.

Assets compromised: **(A2)**

(C15) This is prevented by using TLS where the machine needs to send a certificate to the broker when connecting. The attacker has no access to the private key of the machine and the certificate signed by the attacker would not be valid. Therefore, no communication would be set up.

(T16) An attacker might try to send malicious or corrupted data using WLAN to the machines or brokers. This might insert malicious software into the factory's network and reveal confidential data or cause a break down of machines or brokers.

Assets compromised: **(A1)**, **(A2)**

(C16) All wireless communications between machines and broker are encrypted using TLS, therefore, the attacker is not able to inject any data .

(T17) The attacker might try to act as a machine and publish/subscribe data. This would enable the attacker to read confidential data and inject malicious data. The malicious data would be distributed by the broker to many machines.

Assets compromised: **(A1)**, **(A2)**

(C17) Any machine trying to connect to a broker via the wireless TLS secured connection needs to provide a certificate to the broker when establishing the connection. The broker would reject the attacker's certificate as it is not signed by the company and the connection is not set up.

(T18) The attacker might try to act as a broker. Machines and other brokers would connect to the broker and send confidential data. The attacker would be able to steal any kind of data or send malicious data to the machines.

Assets compromised: **(A1)**, **(A2)**

(C18) As TLS is used to secure the connection, the machine requests a certificate from the broker. The attacker's certificate is rejected by the machine and it will not set up a connection to the attacker's broker.

(E8) Maintenance Worker. The maintenance worker could cause all threats a **trespasser** could. The countermeasures and residual risks are also the same as for the **trespasser**, therefore, they are not listed again. The list just contains the additional threats that could be caused by a maintenance worker which affect mainly the wired communication.

(T19) The maintenance worker might perform a DoS attack on the machines or brokers such as destroying them or removing the security controller. Assets compromised: **(A1)**, **(A2)**

(R19) There is no countermeasure against this.

(T20) The maintenance worker might try to change the keys or certificates on the machines and brokers. This would enable him to communicate with machines or brokers and steal confidential data.

Assets compromised: **(A1)**, **(A2)**, **(A3)**

(C20) As an authentication is needed to change the keys or certificates on the security controller, only trustworthy factory employees can authenticate themselves on the security controller and change the keys and certificates.

(T21) Maintenance workers might connect themselves physically to the machine via Ethernet and can act like they where another machine. They could then try to send data to a connected machine and induce faulty or malicious data into the machine. This might cause errors in the production flow.

Assets compromised: **(A1)**, **(A2)**, **(A3)**

(C21) Each data packet is digitally signed by the sending machine. The receiving machine checks if the signature is valid. A maintenance worker has no access to the private keys on the machines and can, therefore, not generate a valid signature.

(T22) The maintenance worker might connect themselves physically to the machine via Ethernet and read data sent by the machine.

Assets compromised: **(A1)**

(R22) Nothing can be done against this problem. This is just a minor threat as no confidential data is sent between machines but mainly information regarding the production flow or currently manipulated product.

(T23) A maintenance worker might try to connect physically via any interface and change the configuration of the machine or inject malicious data.

Assets compromised: **(A1)**, **(A2)**, **(A3)**

(C23) All debug and other interfaces on machines need to be secured by authentication mechanisms. Only authorized persons with access to the authentication data are able to change or add configurations or any other data on the machine.

(T24) A maintenance worker might perform a Man-In-The-Middle attack on the traffic between machines. Machines would communicate with the attacker without knowing it. The attacker could manipulate the data before forwarding it to the receiving machine.

Assets compromised: **(A1)**

(C24) As the data is digitally signed, the attacker is not able to manipulate the data. Furthermore, he cannot create a valid signature of the data as he can not generate valid keys of a machine. Reading the data is not a problem as no confidential data is sent between machines.

The threat analysis shows that all assets of this hybrid communication approach can be protected with the proposed techniques. For the wireless communication all three requirements confidentiality, integrity and authentication are fulfilled. For the wired communication the two required security measures integrity and authentication are fulfilled. This means that the used security mechanisms meet the security requirements for the proposed communication architecture. The remaining residual risks do not provide a significant security risk or can be prevented by the factory owner.

5.2 Evaluation of Communication Overheads

In this section, the overhead of using the selected security mechanisms is compared to sending packets plain. First, the difference for the wireless communication where TLS is used, will be discussed. After that the overhead due to using the ECDSA for the wired communication will be shown.

5.2.1 Wireless Communication

TLS is used to secure the transfer of data using WLAN. To establish a TLS connection it is necessary to perform a handshake. The following list shows the steps for server and client and the overhead generated by the handshake:

- **ClientHello:** The size of this message is around 160 bytes. It depends on the number of cipher suits available on the client.
- **ServerHello:** ServerHello messages have a size of around 70 bytes depending on the TLS extension.
- **Certificate:** The size of certificates varies greatly. In the case of this thesis, the generated certificate of the server has a size of 1230 bytes. The certificates of the machines have 1230 bytes. The CA certificate has a size of 1340 bytes. In total, the exchanged certificates have a size of 3800 bytes.
- **ClientKeyExchange:** This message has a size of 130 bytes and is used to generate the pre-master secret.
- **ChangeCipherSpec(Client and Server):** This message has a size of exactly 1 byte. This results in a total overhead of 2 bytes.
- **Finish(Client and Server):** The Finish message has size of 12 bytes for TLS. If SSL would have been used, the overhead would have been 36 bytes. The use of TLS results in a total overhead of 24 bytes.
- 4 bytes per sent message for the handshake header also need to be considered. As 9 messages are sent between client and server, 36 bytes need to be added to the overhead.
- Additionally, each transmission results in a overhead of 5 bytes for the TLS record header. As 4 transmissions where done, the total overhead is 20 bytes

$$\begin{array}{r} 36 \\ +\ 20 \\ +\ 160 \\ +3800 \\ +\ 130 \\ +\ 2 \\ +\ 24 \\ \hline 4172 \end{array}$$

When summing up the values, the total overhead of establishing a TLS connection is **4172** bytes. After the connection was established, the overhead for sending the AES encrypted data is at most **40 bytes** due to the MAC (20 bytes), the AES encryption (max. 15 bytes), and the TLS record header (5 bytes).

5.2.2 Wired Communication

For the WebSocket communication the overhead due to the elliptic curve signature is rather small. For a message, the overhead due to the self-signed signature used in the implementation is **40 bytes**. This can be computed by doubling the key size of the elliptic curve plus adding up to **16 bytes** ASN.1 encoding. The key size is 192 bit which results in **24 bytes**.

6 Conclusion and Future Work

In this chapter, the limitations, future work and conclusion of this thesis will be discussed. First, the work will be concluded with a recapitulation of the proposed approach. After that, the limitations regarding the design, implementation and threat analysis will be explained. Finally, that possible future work will be depicted.

6.1 Conclusion

In this thesis, a hybrid communication approach for smart factories was proposed. It contains wired and wireless communication technologies to deal with the requirements of a smart factory environment. Smart factories are designed in a way that smart objects in the factory interact in a context-aware matter and organize the cooperation to solve tasks without the need for central government. Therefore, a combination of wired and wireless communication technologies is used in this approach. Wireless communication provides a much higher flexibility when adding or removing devices than wired technologies. However, wireless technologies suffer from issues that do not arise when communicating with cables. Due to the massive number of devices such as smart sensors, machines, or controllers that operate in a smart factory, wireless communication is unavoidable.

Wireless communication has one serious disadvantage compared to wired communication regarding the used medium. Wireless communication uses air as medium which means that every person can see the sent data whereas for wired communication this is only necessary when a person is physically connected to the cable. This issue results in the need for security mechanisms to protect the sent data from trespassers. Especially in a smart factory environment, the security of data is very important to prevent leeching of confidential data or injecting malicious data into the network.

As a hybrid communication approach was developed in this thesis, different security mechanisms need to be selected to protect the data from being leeched or to protect machines from malicious software. The wireless communication was secured by using TLS and the wired communication was secured with digital signatures. These mechanisms ensure confidentiality, integrity and authentication for the wireless communication, and integrity and authentication for the wired technology. The selected mechanisms where evaluated regarding their ability to fulfill the security requirements of both communication technologies. The security analysis showed that the selected security mechanisms are able to protect the proposed hybrid communication approach. Therefore, the proposed communication approach with the selected security mechanisms could be used in a smart factory.

6.1.1 Limitations

In this section, the limitations of this thesis will be shown. First, the limitations of the hybrid architecture will be explained. After that, the limitations the implementation will be discussed and finally, the ones of the threat analysis will be shown.

The limitations of the hybrid communication approach will be discussed in the following list:

- The used wireless technology is WLAN. This might cause problems in an industrial environment when many devices try to communicate to each other as the transmission would take very long and is not deterministic.
- As wired communication technology, Ethernet was chosen. This might be a problem as more than two machines might want to communicate to each other and therefore, each machine would need many Ethernet ports.

The following list gives limitations of the implementation:

- In the architecture, security controllers and HSMs on every machine and broker are designated. For the implementation, no security controllers or HSMs are used. Furthermore, the implementation is a prototypical proof-of-concept implementation which means that it should just be proven that such an approach actually works.
- The keys for the signatures were generated once and stored manually on the machines and are not exchanged during connection establishment. They can only be updated manually.
- The certificates were self-signed and manually stored on each machine and broker. They can only be changed manually.
- As Python was used for the implementation the behavior of the threads is not deterministic which means that no statement about the timing behavior can be made.

The following list states the limitations of the threat analysis:

- In the thread analysis several assumptions were made. One of them was that each machine and broker uses a security controller and a HSMs to secure the key. This was not implemented but assumed in the threat analysis because when using this approach in a real factory, the machines and brokers would definitely contain security controllers and HSMs.
- The thread analysis did not take security issues when distributing keys and certificates between devices into account. This would have exceeded the scope of the work and was omitted.

6.2 Future Work

This section contains possible future work to extend or improve the currently proposed approach. It is divided into several subsections where future work for the hybrid communication approach, the implementation, and the threat analysis is discussed.

In the following list, the future work for the communication approach will be discussed:

- Another wireless communication technology that is more robust and provides a more deterministic behavior to be able to fulfill timing constraints could be used. WLAN might not meet the requirements when connecting hundreds of devices as the transmission behavior is non-deterministic and the the packet loss rate is higher with more communicating devices. A possibility would be one of the other wireless technologies listed in Section 2.2.1.
- A different wired communication technology could be used as it would need a lot of Ethernet ports to connect every machine with every other. A field bus could be used as some provide fast, deterministic data transmission and can connect several devices with less needed ports. Possible technologies were already discussed in Section 2.2.1.
- A different security mechanism for protecting the data sent with a wireless technology could be used. TLS has a very high overhead when the connection is established. Due to the huge number of devices in a smart factory such a process would be too time consuming. Furthermore, the package loss is high when many devices communicate over the same medium which is also a disadvantage with TLS as it sends many messages when establishing the connection.
- Security mechanisms to update the configuration of a machine or change the keys and certificates should be invented.

The following list shows the possible future work for the current implementation:

- Instead of the Raspberry Pi, a platform that enables real-time processing could be used to evaluate the hybrid communication approach and the security mechanisms regarding timing constraints.
- A different programming language should be used to perform real-time processing and enable the previously mentioned evaluation.
- A real security controller and HSM could be used to secure the key and encrypt the data.

The future work for the threat analysis is shown in the following list:

- Analyze threats of possible security mechanisms to update the keys and certificates on the devices.
- Compare different security mechanisms for the proposed communication approach to find better suiting techniques.

Bibliography

[1] Zuehlke, Detlef, "SmartFactory—Towards a factory-of-things," *Annual Reviews in Control*, vol. 34, no. 1, pp. 129–138, 2010.

[2] Bundesministerium für Bildung und Forschung Referat, *Zukunftsbild Industrie 4.0.* Bundesministerium für Bildung und Forschung Referat, 2013.

[3] Weiser, Mark, "The computer for the 21st century," *Scientific american*, vol. 265, no. 3, pp. 94–104, 1991.

[4] "The Internet of Things: A survey ," *Computer Networks* , vol. 54, no. 15, pp. 2787 – 2805, 2010, .

[5] Lasi, Heiner and Fettke, Peter and Kemper, Hans-Georg and Feld, Thomas and Hoffmann, Michael, "Industry 4.0," *Business & Information Systems Engineering*, vol. 6, no. 4, p. 239, 2014.

[6] Kang, Hyoung Seok and Lee, Ju Yeon and Choi, SangSu and Kim, Hyun and Park, Jun Hee and Son, Ji Yeon and Kim, Bo Hyun and Do Noh, Sang, "Smart manufacturing: Past research, present findings, and future directions," *International Journal of Precision Engineering and Manufacturing-Green Technology*, vol. 3, no. 1, pp. 111–128, 2016.

[7] Jay Lee, "Smart Factory Systems," *Informatik Spektrum*, vol. 38, pp. 230–235, 2015.

[8] Hertel, Michael, "Risiken der Industrie 4.0–Eine Strukturierung von Bedrohungsszenarien der Smart Factory," *HMD Praxis der Wirtschaftsinformatik*, vol. 52, no. 5, pp. 724–738, 2015.

[9] TS102689, ETSI, "Machine-to-Machine Communications (M2M): M2M Service Requirements."

[10] Weyrich, Michael and Schmidt, Jan-Philipp and Ebert, Christof, "Machine-to-Machine communication," *IEEE Software*, vol. 31, no. 4, pp. 19–23, 2014.

[11] Haryantho, Krisna, "3GPP Long Term Evolution (LTE)," 2008.

[12] Rech, Jörg, *Wireless LANs: 802.11-WLAN-Technologie und praktische Umsetzung im Detail.* Heise Verlag, 2012.

[13] G. Cena and I. C. Bertolotti and A. Valenzano and C. Zunino, "Evaluation of Response Times in Industrial WLANs," *IEEE Transactions on Industrial Informatics*, vol. 3, no. 3, pp. 191–201, Aug 2007.

[14] E. Ferro and F. Potorti, "Bluetooth and Wi-Fi wireless protocols: a survey and a comparison," *IEEE Wireless Communications*, vol. 12, no. 1, pp. 12–26, Feb 2005.

[15] Gomez, Carles and Oller, Joaquim and Paradells, Josep, "Overview and evaluation of bluetooth low energy: An emerging low-power wireless technology," *Sensors*, vol. 12, no. 9, pp. 11 734–11 753, 2012.

[16] Baronti, Paolo and Pillai, Prashant and Chook, Vince WC and Chessa, Stefano and Gotta, Alberto and Hu, Y Fun, "Wireless sensor networks: A survey on the state of the art and the 802.15. 4 and ZigBee standards," *Computer communications*, vol. 30, no. 7, pp. 1655–1695, 2007.

[17] Song, Jianping and Han, Song and Mok, Al and Chen, Deji and Lucas, Mike and Nixon, Mark and Pratt, Wally, "WirelessHART: Applying wireless technology in real-time industrial process control," in *Real-Time and Embedded Technology and Applications Symposium, 2008. RTAS'08. IEEE.* IEEE, 2008, pp. 377–386.

[18] Want, Roy, "An introduction to RFID technology," *IEEE Pervasive Computing*, vol. 5, no. 1, pp. 25–33, 2006.

[19] *Fundamental Operating Principles.*

[20] Stenumgaard, Peter and Chilo, José, J. Ferrer-Coll, and P. Angskog, "Challenges and conditions for wireless machine-to-machine communications in industrial environments."

[21] Small, Robert W and van den Enden, John P, "D-sub connector," Apr. 23 1996, US Patent 5,509,821.

[22] Tovar, Eduardo and Vasques, Francisco, "Real-time fieldbus communications using Profibus networks," *IEEE transactions on Industrial Electronics*, vol. 46, no. 6, pp. 1241–1251, 1999.

[23] Feld, Joachim, "PROFINET-scalable factory communication for all applications," in *Factory Communication Systems, 2004. Proceedings. 2004 IEEE International Workshop on.* IEEE, 2004, pp. 33–38.

[24] Engels, Horst, "CAN-bus," *Franzis, Poing*, pp. 978–3 772 351 464, 2000.

[25] Lian, Feng-Li and Moyne, James R and Tilbury, Dawn M, "Performance evaluation of control networks: Ethernet, ControlNet, and DeviceNet," *IEEE control systems*, vol. 21, no. 1, pp. 66–83, 2001.

[26] ODVA, "EtherNet/IP," 2015.

[27] Cena, Gianluca and Valenzano, Adriao and Vitturi, Stefano, “Hybrid wired/wireless networks for real-time communications,” *IEEE industrial electronics magazine*, vol. 2, no. 1, pp. 8–20, 2008.

[28] Edited by Andrew Banks and Rahul Gupta, “MQTT Version 3.1.1 Plus Errata 01,” 2015.

[29] Stanford-Clark, Andy and Truong, Hong Linh, “MQTT for sensor networks (MQTT-SN) protocol specification,” *International Business Machines Corporation version*, vol. 1, 2008.

[30] Shelby, Zach and Hartke, Klaus and Bormann, Carsten, “The constrained application protocol (CoAP),” Tech. Rep., 2014.

[31] Saint-Andre, Peter and Smith, Kevin and Tronçon, Remko and Troncon, Remko, *XMPP: the definitive guide.* ” O’Reilly Media, Inc.”, 2009.

[32] Bray, Tim and Paoli, Jean and Sperberg-McQueen, C Michael and Maler, Eve and Yergeau, François, “Extensible markup language (XML),” *World Wide Web Consortium Recommendation REC-xml-19980210. http://www. w3. org/TR/1998/REC-xml-19980210*, vol. 16, p. 16, 1998.

[33] Pardo-Castellote, Gerardo, “Omg data-distribution service: Architectural overview,” in *Distributed Computing Systems Workshops, 2003. Proceedings. 23rd International Conference on.* IEEE, 2003, pp. 200–206.

[34] S. Vinoski, “Advanced Message Queuing Protocol,” *IEEE Internet Computing*, vol. 10, no. 6, pp. 87–89, Nov 2006.

[35] O’Hara, John, “Toward a commodity enterprise middleware,” *Queue*, vol. 5, no. 4, pp. 48–55, 2007.

[36] Bahga, A. and Madisetti, V., *Internet of Things: A Hands-On Approach:* . Arsheep Bahga & Vijay Madisetti, 2014.

[37] Fette, Ian, “The websocket protocol,” 2011.

[38] Herfs, Werner and Lohse, Wolfram and Özdemir, Denis and Malik, Adam, “Komponentenkapselung und -selbstbeschreibung,” *atp edition*, vol. 55, no. 12, pp. 54–60, 2013.

[39] Lass, Sander and Kotarski, David, “IT-Sicherheit als besondere Herausforderung von Industrie 4.0,” *Industrie*, vol. 4, pp. 397–419, 2014.

[40] Junker, Holger, “IT-Sicherheit für Industrie 4.0 und IoT,” *Datenschutz und Datensicherheit-DuD*, vol. 39, no. 10, pp. 647–651, 2015.

[41] Wallner, Sebastian and Mayer, Karlheinz, “Ein ganzheitliches IT-Sicherheitskonzept für Industrie 4.0 Infrastrukturen,” *Datenschutz und Datensicherheit-DuD*, vol. 40, no. 1, pp. 43–46, 2015.

[42] Sadeghi, Ahmad-Reza and Wachsmann, Christian and Waidner, Michael, “Security and privacy challenges in industrial internet of things,” in *Proceedings of the 52nd Annual Design Automation Conference.* ACM, 2015, p. 54.

[43] Barki, Amira and Bouabdallah, Abdelmadjid and Gharout, Saïd and Traoré, Jacques, “M2M Security: Challenges and Solutions,” *IEEE Communications Surveys & Tutorials*, vol. 18, no. 2, pp. 1241–1254, 2015.

[44] Lawson, Nate, “Side-channel attacks on cryptographic software,” *IEEE Security & Privacy*, vol. 7, no. 6, pp. 65–68, 2009.

[45] Needham, Roger M, “Denial of service,” in *Proceedings of the 1st ACM Conference on Computer and Communications Security.* ACM, 1993, pp. 151–153.

[46] Hancke, Gerhard P, “A practical relay attack on ISO 14443 proximity cards,” *Technical report, University of Cambridge Computer Laboratory*, vol. 59, pp. 382–385, 2005.

[47] Jawurek, Marek and Kerschbaum, Florian and Danezis, George, “SoK: Privacy Technologies for Smart Grids—A Survey of Options,” *Microsoft Res., Cambridge, UK*, 2012.

[48] Ahmadzadegan, M Hossein and Fabritius, Tapio, “A multi-purpose triangular framework for M2M communication security,” in *Computer Applications and Information Systems (WCCAIS), 2014 World Congress on.* IEEE, 2014, pp. 1–5.

[49] Mahkonen, Heikki and Rinta-aho, Teemu and Kauppinen, Tero and Sethi, Mohit and Kjällman, Jimmy and Salmela, Patrik and Jokikyyny, Tony, “Secure m2m cloud testbed,” in *Proceedings of the 19th annual international conference on Mobile computing & networking.* ACM, 2013, pp. 135–138.

[50] Bellare, Mihir and Desai, Anand and Jokipii, Eron and Rogaway, Phillip, “A concrete security treatment of symmetric encryption,” in *Foundations of Computer Science, 1997. Proceedings., 38th Annual Symposium on.* IEEE, 1997, pp. 394–403.

[51] Daemen, Joan and Rijmen, Vincent, *The design of Rijndael: AES-the advanced encryption standard.* Springer Science & Business Media, 2013.

[52] W. Diffie and M. Hellman, “New directions in cryptography,” *IEEE Transactions on Information Theory*, vol. 22, no. 6, pp. 644–654, Nov 1976.

[53] Dierks, Tim, “The transport layer security (TLS) protocol version 1.2,” 2008.

[54] Johnson, Don and Menezes, Alfred and Vanstone, Scott, “The Elliptic Curve Digital Signature Algorithm (ECDSA),” *International Journal of Information Security*, vol. 1, no. 1, pp. 36–63, 2001.

[55] Kravitz, David W, “Digital signature algorithm,” Jul. 27 1993, US Patent 5,231,668.

[56] Smart, Nigel P, “The discrete logarithm problem on elliptic curves of trace one,” *Journal of cryptology*, vol. 12, no. 3, pp. 193–196, 1999.

[57] Eugster, Patrick Th and Felber, Pascal A and Guerraoui, Rachid and Kermarrec, Anne-Marie, “The many faces of publish/subscribe,” *ACM Computing Surveys (CSUR)*, vol. 35, no. 2, pp. 114–131, 2003.

[58] Goldreich, Oded, *Foundations of cryptography: volume 2, basic applications.* Cambridge university press, 2009.

[59] “Raspberry Pi 3,” https://www.raspberrypi.org/products/raspberry-pi-3-model-b/, [Online; accessed September 2016].

[60] “Raspbian Jessie,” https://www.raspberrypi.org/blog/raspbian-jessie-is-here/, [Online; accessed September 2016].

[61] Myagmar, Suvda and Lee, Adam J and Yurcik, William, “Threat modeling as a basis for security requirements,” in *Symposium on requirements engineering for information security (SREIS)*, vol. 2005. Citeseer, 2005, pp. 1–8.

[62] F. Reimair and P. Teufl and C. Kollmann and C. Thaller, “MoCrySIL - carry your cryptographic keys in your pocket,” in *2015 12th International Joint Conference on e-Business and Telecommunications (ICETE)*, vol. 04, July 2015, pp. 285–292.